## ABB

## MANUAL

## PLC Automation

## Automation Builder, programmable logic controllers, control panels

Automation Builder 2.7.0, AC500 V3, AC500-eCo V3, AC500-XC V3, AC500-S, CP600, CP600-eCo, CP600-Pro


## Table of contents

1 Preface ..... 6
1.1 Documentation guide ..... 6
1.2 Do I use AC500 V2 or AC500 V3?. ..... 6
1.3 Your tasks - documentation from the user's point of view ..... 7
1.4 Regulations. ..... 9
1.5 Older revisions of this document ..... 10
1.6 Structure of safety notices ..... 10
2 Safety instructions ..... 12
3 Getting started with example projects ..... 16
3.1 Introduction. ..... 16
3.2 Engineering software Automation Builder ..... 16
3.2.1 Purpose ..... 16
3.2.2 Installation of the Automation Builder ..... 16
3.2.3 Licensing procedure ..... 17
3.2.4 Setting up of communication parameters in Windows. ..... 19
3.3 Hardware AC500 V3. ..... 21
3.3.1 Configuration for example projects ..... 21
3.3.2 System assembly, construction and connection. ..... 23
3.4 Example project for central I/O expansion. ..... 24
3.4.1 Purpose ..... 24
3.4.2 Preconditions ..... 24
3.4.3 Creation, setting up and saving of your AC500 V3 project. ..... 24
3.4.4 Configuration of the I/O module ..... 29
3.4.5 Programming and compilation ..... 33
3.4.6 Setting up the communication gateway ..... 42
3.4.7 Installation and update of the AC500 V3 firmware ..... 48
3.4.8 Logging in to CPU and downloading the program ..... 50
3.4.9 Testing the program. ..... 51
3.4.10 Setting up a visualization ..... 53
3.4.11 Creation of a visualization ..... 57
3.4.12 Enabling a web visualization ..... 65
3.4.13 Reset the CPU ..... 70
3.5 Example project for remote I/O expansion with PROFINET ..... 71
3.5.1 Purpose ..... 71
3.5.2 Preconditions ..... 71
3.5.3 Set-up PROFINET controller. ..... 72
3.5.4 Set-up PROFINET device ..... 74
3.5.5 Add remote I/O expansion to project. ..... 78
3.5.6 Test the program. ..... 82
3.5.7 Reset the CPU ..... 83
4 System overview, planning and operation ..... 85
4.1 System overview ..... 85
4.1.1 AC500 PLC product family ..... 85
4.1.2 AC500/S500 system structure ..... 86
4.1.3 AC500-eCo/S500-eCo system structure ..... 87
4.1.4 AC500/S500: Short description hardware ..... 89
4.1.5 AC500-eCo/S500-eCo: Short description hardware ..... 93
4.1.6 AC500-S ..... 94
4.1.7 CP600 control panels ..... 95
4.1.8 Automation Builder: Short description engineering software ..... 95
4.2 Application planning ..... 97
4.2.1 Safety instructions. ..... 97
4.2.2 Processor module and I/O selection. ..... 97
4.2.3 I/O bus - Communication within the PLC ..... 97
4.2.4 Fieldbus connectivity options. ..... 98
4.2.5 Power supply dimensioning. ..... 100
4.2.6 Libraries, software packages and licensed features ..... 103
4.3 Mechanical planning and installation ..... 104
4.3.1 Control cabinet assembly (indoor use). ..... 104
4.3.2 Mounting and demounting - general information ..... 106
4.3.3 Mounting and demounting the terminal base ..... 107
4.3.4 Mounting and demounting the AC500 processor module. ..... 109
4.3.5 Mounting and demounting the communication module ..... 109
4.3.6 Mounting and demounting the terminal unit. ..... 111
4.3.7 Mounting and demounting the I/O module. ..... 113
4.3.8 Mounting and demounting the AC500-eCo processor module ..... 114
4.3.9 Mounting and demounting the S500-eCo I/O module ..... 123
4.3.10 Accessories for AC500 (Standard) ..... 126
4.3.11 Accessories for AC500-eCo. ..... 127
4.3.12 Control panels CP600 ..... 130
4.4 Wiring. ..... 131
4.4.1 Grounding concept ..... 131
4.4.2 EMC-conforming assembly and construction ..... 132
4.4.3 Connection and wiring of the PLC platform. ..... 137
4.4.4 Connection and wiring of the modules ..... 145
4.5 Configuration and programming ..... 145
4.6 Commissioning ..... 146
4.7 Operation. ..... 147
4.7.1 Operating modes. ..... 147
4.7.2 Diagnosis system ..... 147
4.7.3 Status LEDs, display and control elements. ..... 148
4.7.4 PLC firmware/application update ..... 148
4.7.5 Control panel firmware/application update. ..... 148
4.7.6 Reset Warm ..... 149
4.8 Troubleshooting ..... 149
4.8.1 General. ..... 149
4.8.2 Possible malfunctions. ..... 149
4.8.3 Execution of a hotfix. ..... 151
4.9 Maintenance ..... 152
4.9.1 Maintenance intervals ..... 152
4.9.2 Replace an AC500 module ..... 152
4.9.3 Replace an I/O module with hot swap ..... 153
4.9.4 Replace a CP600 control panel ..... 157
4.10 Decommissioning ..... 157
4.11 Recycling. ..... 158
5 Hardware descriptions. ..... 159
5.1 Technical data of the system. ..... 159
5.1.1 System data AC500-eCo ..... 159
5.1.2 System data AC500. ..... 166
5.1.3 System data AC500-XC ..... 169
5.1.4 AC500-S ..... 174
5.1.5 CP600 ..... 174
5.2 Device specifications ..... 175
5.2.1 Processor modules ..... 175
5.2.2 Communication modules for AC500(-XC) processor modules ..... 327
5.2.3 Terminal bases for AC500(-XC) processor modules and communication modules ..... 377
5.2.4 I/O modules ..... 387
5.2.5 Terminal units for S500(-XC) I/O modules ..... 938
5.2.6 Communication interface modules ..... 951
5.2.7 Terminal units for communication interface modules ..... 1270
5.2.8 Accessories ..... 1286
5.2.9 Storage devices ..... 1332
5.3 Status LEDs, display and control elements ..... 1338
6 Configuration and programming. ..... 1339
6.1 Cyber security ..... 1340
6.1.1 General ..... 1340
6.1.2 Defense in depth ..... 1341
6.1.3 Secure operation ..... 1342
6.1.4 Hardening ..... 1345
6.1.5 Certificates factory default - no encryption ..... 1346
6.1.6 Open Ports and Services ..... 1349
6.2 Engineering software Automation Builder ..... 1350
6.2.1 Introduction ..... 1350
6.2.2 Automation Builder updates ..... 1350
6.2.3 Use of open source components ..... 1350
6.2.4 Managing your licenses ..... 1350
6.2.5 Setting up of communication parameters in Windows. ..... 1382
6.2.6 Version information ..... 1384
6.2.7 PLC runtime and demo licensing ..... 1385
6.2.8 Create log files for support ..... 1385
6.2.9 Menues, views, windows ..... 1386
6.2.10 Device repository ..... 1387
6.2.11 Creating and configuring projects. ..... 1390
6.2.12 Handling of AC500 projects ..... 1391
6.2.13 Connection of devices ..... 1392
6.2.14 Connection of serial interfaces ..... 1394
6.2.15 Converting an AC500 V2 project to an AC500 V3 project ..... 1394
6.2.16 Automation Builder installation manager ..... 1395
6.3 Configuration in Automation Builder for AC500 V3 products ..... 1398
6.3.1 General settings ..... 1398
6.3.2 PLC devices and components ..... 1443
6.3.3 Windows server. ..... 1647
6.3.4 Protocols and special servers ..... 1647
6.3.5 Data transfer and programming ..... 1750
6.3.6 Server installation ..... 1758
6.3.7 Converting an AC500 V2 project to an AC500 V3 project ..... 1799
6.4 Programming with CODESYS ..... 1800
6.4.1 CODESYS Development System ..... 1800
6.4.2 Fieldbus Support. ..... 2812
6.4.3 Runtime systems, OPC UA server ..... 2834
6.4.4 Libraries ..... 2852
6.4.5 CODESYS Visualization ..... 2852
6.4.6 CODESYS Visualization Support ..... 3753
6.5 Libraries and solutions ..... 3773
6.5.1 Information on libraries. ..... 3773
6.5.2 Reference to CODESYS (V3) ..... 3773
6.5.3 Library Manager functionality ..... 3773
6.5.4 ACS/DCS drives libraries ..... 3778
6.5.5 Application libraries ..... 3833
6.5.6 BACnet-BC ..... 3833
6.5.7 CAA library guidelines ..... 3849
6.5.8 Data Logger Library. ..... 3849
6.5.9 High Availability Modbus TCP ..... 3859
6.5.10 Motion Control ..... 3904
6.5.11 MQTT client library. ..... 4048
6.5.12 PLCopen libraries ..... 4051
6.5.13 Pumping library V3 ..... 4056
6.5.14 Reference, function blocks ..... 4086
6.6 Engineering interfaces and tools ..... 4191
6.6.1 Export and import interfaces ..... 4191
6.6.2 CODESYS Security Agent ..... 4201
6.6.3 CODESYS Static Analysis ..... 4208
6.6.4 Multi download tool ..... 4306
6.6.5 Tool Calling Interface (TCI) implementation ..... 4309
6.6.6 Drive composer pro integration ..... 4311
6.6.7 Professional Version Control ..... 4315
6.6.8 Subversion ..... 4356
6.6.9 Mail service with SMTP/SMTPS ..... 4361
6.6.10 SNMP ..... 4362
6.6.11 Scripting with Python ..... 4362
6.7 Human machine interface ..... 4397
6.7.1 Control panels ..... 4397
6.7.2 Panel Builder interface ..... 4398
6.7.3 SCADA Integration ..... 4406
6.8 System technology ..... 4411
6.8.1 General ..... 4411
6.8.2 System technology of CPU and overall system ..... 4412
6.8.3 System technology of the AC500 communication modules ..... 4565
6.8.4 System technology of the communication interface modules ..... 4568
6.9 Diagnosis and debugging ..... 4605
6.9.1 The diagnosis system ..... 4605
6.9.2 Online diagnosis in Automation Builder ..... 4640
6.9.3 Diagnosis messages ..... 4655
7 Glossary ..... 4710
8 Index ..... 4713

## 1 Preface

### 1.1 Documentation guide



Where to find information about:

- PLC system
- $y^{2}$ Chapter 4.1 "System overview" on page 85
- $y^{\Perp}$ Chapter 3 "Getting started with example projects" on page 16
- Starter kit: Introduction to PLC programming and visualization, including all needed hardware components
- Safety user manual for safety PLCs
- Devices
- Data sheets
- Installation instructions
- ${ }^{3}$ Chapter 5.2 "Device specifications" on page 175
- Safety user manual for safety PLCs
- Control panels
- Software and programming
- ${ }^{\text {® }}$ Chapter 6 "Configuration and programming" on page 1339
- Programming examples available for download
- Programming examples available in the engineering suite: Open Automation Builder menu "Help $\rightarrow$ Project examples"
- Application descriptions for specific features and use cases
- Release notes for the latest version of the engineering suite Automation Builder


### 1.2 Do I use AC500 V2 or AC500 V3?

This document describes AC500 V3 products.
We offer different ranges of the processor modules (V2, V3). You can easily find out which ones are used in your system. Either have a look at your used modules or your configuration.


Fig. 1: Hardware: Processor modules

| Written type on the module | Example | Range | Relevant manual |
| :--- | :--- | :--- | :--- |
| Processor module type PM + <br> $\mathbf{3}$ digits | PM595 | V2 | AC500 V2 |
| Processor module type PM + <br> 4 digits | PM5670 | V3 | AC500 V3 (this manual) |



Fig. 2: Software: Configuration in engineering software Automation Builder

| Written type in brackets | Example | Range | Relevant manual |
| :--- | :--- | :--- | :--- |
| Processor module type PM + <br> $\mathbf{3}$ digits | PM595 | V2 | AC500 V2 |
| Processor module type PM + <br> $\mathbf{4}$ digits | PM5670 | V3 | AC500 V3 (this manual) |

### 1.3 Your tasks - documentation from the user's point of view

All information about AC500, AC500-XC and AC500-eCo is available in this manual.
All information about AC500-S and AC500-S-XC is available online in the safety user manual.

As a specialist for PLCs, new to AC500 PLC

| PLC related videos | youtube.com/user/abbplc |
| :---: | :---: |
| Getting started with example projects | * Chapter 3 "Getting started with example projects" on page 16 |
| PLC system description | を Chapter 4.1 "System overview" on page 85 |
| Hardware descriptions | « Chapter 5.2 "Device specifications" on page 175 |
| System technology: System behavior, interaction between PLC behavior (firmware), configuration, programming and use cases. | を Chapter 6.8 "System technology" on page 4411 |

[^0]| AC500 V3 CPU specifications | ② Chapter 5.2 "Device specifications" on page 175 |
| :---: | :---: |
| Comparison of product features | - ${ }^{\circ}$ Chapter 5.2.1.2.1 "Product overview and comparison" on page 303 <br> - via product catalog |
| Convert an AC500 V2 project to an AC500 V3 project | ② Chapter 6.2.15 "Converting an AC500 V2 project to an AC500 V3 project" on page 1394 |
| Compatible modules with AC500 CPUs | « Chapter 5.2 "Device specifications" on page 175 |
| Documentation for AC500 V2 | AC500 V2 |

As a mechanical/electrical designer

| PLC system description | ${ }^{\text {s }}$ Chapter 4.1 "System overview" on page 85 |
| :---: | :---: |
| Hardware descriptions | « Chapter 5.2 "Device specifications" on page 175 |
| Comparison of product features | - ${ }^{\circ}$ Chapter 5.2.1.2.1 "Product overview and comparison" on page 303 <br> - via product catalog (online) |

## As a control cabinet manufacturer

| Assembly of modules | \& Chapter 4.3 "Mechanical planning and <br> installation" on page 104 |
| :--- | :--- |
| Connection of modules | In the device specifications, select the desired <br> product to access the connection for this <br> device. <br> 』 Chapter 5.2 "Device specifications" <br> on page 175 <br> "Device specifications $\rightarrow$ Product group <br> $\rightarrow$ Product type $\rightarrow$ Electrical connection" |
| Installation instructions | $\underline{\text { All AC500 modules }}$ |

As a program-
ming engineer

| Example projects | * Chapter 3 "Getting started with example projects" on page 16 <br> Programming examples for download <br> Programming examples available in engineering suite: Open Automation Builder menu "Help $\rightarrow$ Project examples" <br> Application descriptions for specific features and use cases |
| :---: | :---: |
| Installation of Automation Builder | « Chapter 3.2.2 "Installation of the Automation Builder" on page 16 |
| License management for Automation Builder | を Chapter 6.2.4 "Managing your licenses" on page 1350 |
| Configuration of PLC hardware in Automation Builder | ⓨ Chapter 6.3 "Configuration in Automation Builder for AC500 V3 products" on page 1398 |
| Firmware update | ⓨ Chapter 6.3.1.4 "Firmware identification and update" on page 1419 |
| Programming | * Chapter 6.4 "Programming with CODESYS" on page 1800 |
| Function block libraries | Libraries by ABB \& Chapter 6.5 "Libraries and solutions" on page 3773 <br> CODESYS libraries by 3S $\leadsto$ Chapter 6.4.4 "Libraries" on page 2852 |
| System technology: System behavior, interaction between PLC behavior (firmware), configuration, programming and use cases. | $\stackrel{\Perp}{ }{ }^{\star}$ Chapter 6.8 "System technology" on page 4411 |
| Visualization and web visualization: Example projects | (4) Chapter 3.4 "Example project for central I/O expansion" on page 24 |
| Visualization and web visualization | « Chapter 6.4.5 "CODESYS Visualization" on page 2852 |


| Add，remove or modify software packages in Automation Builder via installation manager | （3）Chapter 6．2．16＂Automation Builder installa－ tion manager＂on page 1395 |
| :---: | :---: |
| Add－on software packages | を Chapter 6.6 ＂Engineering interfaces and tools＂on page 4191 |
| HMI，e．g．，interface to control panels and SCADA systems | を Chapter 6.7 ＂Human machine interface＂ on page 4397 |

As a commis－ sioning engi－ neer

| Function block libraries | Libraries by ABB $\&$ Chapter 6.5 ＂Libraries and solutions＂on page 3773 <br> CODESYS libraries by 3S $\stackrel{y}{4}$ Chapter 6．4．4 ＂Libraries＂on page 2852 |
| :---: | :---: |
| Hardware descriptions | （7）Chapter 5.2 ＂Device specifications＂ on page 175 |
| Troubleshooting | « Chapter 4.8 ＂Troubleshooting＂on page 149 <br> ② Chapter 6.9 ＂Diagnosis and debugging＂ on page 4605 |

As a service engineer

| Troubleshooting | \＆Chapter 4．8＂Troubleshooting＂on page 149 <br> 』 Chapter 6．9＂Diagnosis and debugging＂ <br> on page 4605 |
| :--- | :--- |
| List of diagnosis and error messages | \＆Chapter 6．9．3＂Diagnosis messages＂ <br> on page 4655 |
| Contact ABB support | $\underline{\text { new．abb．com／plc }}$ |

## 1．4 Regulations

Planning and installation of the electrical system

The planning and installation of the electrical system must be carried out in compliance with the applicable regulations and standards．Hazards due to malfunctions must be prevented by taking appropriate measures．
The suitability of the products for the respective application is proven by declarations of con－ formity and certificates．

The PLC Automation catalog contains an overview of the available declarations of conformity and certificates．

Qualified per－Both the AC500 control system and other components in the vicinity are operated with dan－ sonnel
gerous touch voltages．Touching live components can lead to serious health implications or even death．

To avoid such risks and the occurrence of property damage，persons involved in the installation， commissioning and maintenance must have relevant knowledge about：
－Automation technology
－Handling of hazardous voltages
－Application of relevant standards and regulations，accident prevention regulations and reg－ ulations on special environmental conditions（e．g．，hazardous areas due to explosive sub－ stances，heavy soiling or corrosive influences）．

### 1.5 Older revisions of this document

You can always find all revisions of our documents on our website.
AC500 V3 (online)


 Biock Lbranes


Revisions Select any of the revisions
Latest revision Get a link to the latest revision
This revision Get a direct link to the selected revision

### 1.6 Structure of safety notices

Throughout the documentation we use the following types of safety and information notices. They make you aware of safety considerations or give advice on AC500 products usage.


1 Safety alert symbol indicates the danger.
2 Signal word classifies the danger.
3 Type and source of the risk are mentioned and possible consequences are described.
4 Measures to avoid these consequences (enumerations).

Signal words

## DANGER!

DANGER indicates a hazardous situation which, if not avoided, will result in death or serious injury.
Ensure to take measures to prevent the described impending danger.

## WARNING!

WARNING indicates a hazardous situation which, if not avoided, could result in death or serious injury.
Ensure to take measures to prevent the described dangerous situation.

## CAUTION!

CAUTION indicates a hazardous situation which, if not avoided, could result in minor or moderate injury.
Ensure to take measures to prevent the described dangerous situation.

## NOTICE!

NOTICE is used to address practices not related to physical injury but might lead to property damage for example damage of the product.
Ensure to take measures to prevent the described dangerous situation.

NOTE provides additional information on the product, e.g., advices for configuration or best practice scenarios.

## 2 Safety instructions

Relevant standards and regulations, accident prevention regulations and regulations on special environmental conditions must be observed (e.g., hazardous areas due to explosive substances, heavy soiling or corrosive influences).

The devices must be handled and operated within the specified technical data and system data.
The devices contain no serviceable parts and must not be opened.
Removable covers must be closed during operation unless otherwise specified.
Any liability for the consequences of incorrect use or unauthorized repairs is rejected.

Qualified per- Both the AC500 control system and other components in the vicinity are operated with dansonnel gerous touch voltages. Touching live components can lead to serious health implications or even death.

To avoid such risks and the occurrence of property damage, persons involved in the installation, commissioning and maintenance must have relevant knowledge about:

- Automation technology
- Handling of hazardous voltages
- Application of relevant standards and regulations, accident prevention regulations and regulations on special environmental conditions (e.g., hazardous areas due to explosive substances, heavy soiling or corrosive influences).

Functional The AC500-S safety user manual must be read and understood before using the safety configusafety ration and programming tools of Automation Builder/PS501 Control Builder Plus. Only qualified personnel are permitted to work with AC500-S safety PLCs.

General infor- The examples and diagrams in this manual are included solely for illustrative purposes. mation Because of the many variants and requirements associated with any particular installation, ABB cannot assume responsibility or liability for actual use based on the examples and diagrams.
The PLC was developed according to the relevant standards. Any module-specific measures are described in the individual descriptions of the modules.

## PLC-specific safety notices

The product family AC500 control system is designed according to the EN 61131-2 and IEC 61131-2 standards. Any data that differs from IEC 61131-2, is due to the higher requirements of Maritime Services. Other differences are described in the technical data description of the devices.

## NOTICE!

## Avoidance of electrostatic charging

PLC devices and equipment are sensitive to electrostatic discharge, which can cause internal damage and affect normal operation. Observe the following rules when handling the system:

- Touch a grounded object to discharge potential static.
- Wear an approved grounding wrist strap.
- Do not touch connectors or pins on component boards.
- Do not touch circuit components inside the equipment.
- If available, use a static-safe workstation.
- When not in use, store the equipment in appropriate static-safe packaging.


## NOTICE!

## Use of suitable enclosure

The devices must be mounted in a control cabinet that ensures compliance with the specified environmental conditions.

## Cleaning instructions

Do not use cleaning agent for cleaning the device.
Use a damp cloth instead.

Connection plans and a user program must be created so that no dangerous situations can occur during normal operation or failure.

The application must be tested to ensure that no dangerous situations can occur during operation.

Do not operate devices outside of the specified, technical data!
Trouble-free functioning cannot be ensured outside of the specified data.

## NOTICE!

PLC damage due to missing grounding

- Make sure to ground the devices.
- The grounding (switch cabinet grounding) is supplied both by the mains connection (or 24 V supply voltage) and via the DIN rail. The DIN rail must be connected to ground before power is supplied to the device. The grounding may be removed only if it is certain that no more power is being supplied to the control system.
- In case of screw mounting, use metal screws for grounding.


## CAUTION!

## Do not obstruct the ventilation for cooling!

The ventilation slots on the upper and lower sides of the devices must not be covered.

## Run signal and power wiring separately!

CAUTION!

Signal and supply lines (power cables) must be laid out so that no malfunctions due to capacitive and inductive interference can occur (EMC).

## WARNING!

Warning sign on the module!
This indicates that dangerous voltages may be present or that surfaces may have dangerous temperatures.

## WARNING!

## Splaying of strands can cause hazards!

Avoid splayed strands when wiring terminals with stranded conductors.

- Ferrules can be used to prevent splaying.


## WARNING!

## Removal/Insertion under power

Removal or insertion under power is permissible only if all conditions for hot swapping are fullfilled.

๕ Chapter 4.9.3 "Replace an I/O module with hot swap" on page 153
The devices are not designed for removal or insertion under power when the conditions for hot swap do not apply. Because of unforeseeable consequences, it is not allowed to plug in or unplug devices with the power being ON.
Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while they are energized in a hazardous location could result in an electric arc, which could create an ignition source resulting in fire or explosion.
Prior to proceeding, make sure that power is been disconnected and that the area has been thoroughly checked to ensure that flammable materials are not present.

The devices must not be opened when in operation. The same applies to the network interfaces.

Information on batteries

## CAUTION!

## Use only ABB approved lithium battery modules!

At the end of the battery's lifetime, always replace it only with a genuine battery module.

## CAUTION!

## Risk of explosion!

Do not open, re-charge or disassemble lithium batteries. Attempting to charge lithium batteries will lead to overheating and can cause explosions.

Protect them from heat and fire and store them in a dry place.
Never short-circuit or operate lithium batteries with the polarities reversed. The batteries are likely to overheat and explode. Avoid unintentional short circuiting do not store batteries in metal containers and do not place them on metallic surfaces. Escaping lithium is a health hazard.

## Environment considerations

Recycle exhausted batteries. Dispose of batteries in an environmentally conscious manner in accordance with regulations issued by the local authorities.

## 3 Getting started with example projects

### 3.1 Introduction

This document gives an overview of the steps for the first use of a PLC with AC500 V3 processor module and describes:

- installation of the engineering software ${ }^{4}$ Chapter 3.2 "Engineering software Automation Builder" on page 16
- hardware needed for example projects $\Longleftrightarrow$ Chapter 3.3 "Hardware AC500 V3" on page 21
- setting up a first, simple project for a stand-alone CPU with central I/O expansion, including visualization and web visualization ${ }^{\aleph}$ Chapter 3.4 "Example project for central I/O expansion" on page 24
- commissioning a project for remote I/O expansion with PROFINET $\Leftrightarrow$ Chapter 3.5 "Example project for remote I/O expansion with PROFINET" on page 71

Getting started with an AC500-eCo V3 processor module: Starter kit smart guide

### 3.2 Engineering software Automation Builder

### 3.2.1 Purpose

For configuring and programming of any AC500 CPU you need the engineering software suite Automation Builder. Automation Builder is available for download.

Further information on page 16

### 3.2.2 Installation of the Automation Builder

Preconditions You must have admininstrator rights on your PC to install Automation Builder. In case of an update installation:


Create a project archive before upgrading Automation Builder. Project archives contain all project data, including data that is not stored with a *.project file, e.g. device description files for third party devices.
ⓨ Chapter 6.3.1.1.8.2 "Creation of an archive " on page 1408

Installation 1. Go to abb.com/automationbuilder to access the homepage of Automation Builder.
2. In the "Downloads" section, select "Download Automation Builder".
3. In the "Latest Automation Builder" section, select "Automation Builder x.x. Download" (x.x $=$ latest version). This downloads the installer on your PC.
4. Open the downloaded installer and follow the instructions of the installation manager.

5. Keep the default type of installation to "Premium Edition" if available.
6. Select software packages to be installed:

Enable the check box "PLC - AC500 V3" to activate installation of all options for AC500 V3.
7. Click "Download and install" and follow the instructions of the setup.

### 3.2.3 Licensing procedure

When you start Automation Builder software for the first time, you will be asked to choose a license option.
See also:

- Details about the license model, the features of the editions and the latest license information
- Detailed description of the installation and the licensing possibilities of Automation Builder


## Activate a trial

 licenseAutomation Builder software must be installed successfully.PC is connected to the internet.1. Start Automation Builder. $\Rightarrow$ A licensing wizard starts and guides you through the licensing procedure.
2. Enter user information.

In case of future support requests, your registration details enable $A B B$ support team to handle your questions quickly.
3. Select "OK".

4. Enable the trial license.
5. Select "Next".

6. Enable the single PC license and select "Next".

7. Enable online activation and select "Next".
$\Rightarrow$ License activation procedure starts. A successfully ended licensing procedure ends with a success message.
8. Select "OK" to end the wizard.
$\Rightarrow$ Automation Builder license is activated and starts.

### 3.2.4 Setting up of communication parameters in Windows

Setting up of communication parameters

To set up the communication between the PC and the PLC, e.g., for downloading the compiled program, you have to set up the communication parameters.

The IP address of your PC must be in the same class as the IP address of the CPU.
The factory setting of the IP address of the CPU is 192.168.0.10.
The IP address of your PC should be 192.168.0.X. Avoid X = 10 in order to prevent an IP conflict with the CPU.

Subnet mask should be 255.255.255.0.

## Changing of the 1. Open Windows control panel. Click "Network and Internet IP address <br> $\rightarrow$ Network and Sharing Center".

2. Click "Change adapter settings".
$\Rightarrow$
If using existing network with several devices, please pay attention on given network rules or contact your system administrator.
3. Right-click "Local Area Connection (Ethernet)" and select [Properties].

4. Double-click "Internet Protocol Version 4 (TCP/IPv4)".

5. Enter your desired IP address and subnet mask.

### 3.3 Hardware AC500 V3

### 3.3.1 Configuration for example projects

The example projects require a small PLC configuration with I/O devices, e.g., as available in the training case TA5450-CASE.

Table 1: Modules for example projects to get started with AC500 V3 PLC

| Product name | Type | First project <br> (\% Chapter 3.4 <br> "Example project for <br> central I/O expan- <br> sion" on page 24 | Second project <br> 乡 Chapter 3.5 <br> "Example project <br> for remote <br> I/O expansion <br> with PROFINET" <br> on page 71 |
| :--- | :--- | :--- | :--- |
| PM5630-2ETH | AC500 V3 CPU | x | x |
| TB5620-2ETH | terminal base for CPU | x | x |
| DA501 | analog/digital mixed <br> input/output (I/O) <br> module | x | x |
| TU516-H | terminal unit for I/O <br> module | x | x |


| Product name | Type | First project <br> ( Chapter 3.4 <br> "Example project for <br> central I/O expan- <br> sion" on page 24 | Second project <br> ( Chapter 3.5 <br> "Example project <br> for remote <br> I/O expansion <br> with PROFINET" <br> on page 71 |
| :--- | :--- | :--- | :--- |
| CM579-PNIO | PROFINET communi- <br> cation module | -- | x |
| CI502-PNIO | PROFINET commu- <br> nication interface <br> module | -- | x |
| TU508-ETH | terminal unit for com- <br> munication interface <br> module | -- | x |
| TA524 | blind cap for terminal <br> base | x | x |

## Connections



Fig. 3: Training case TA5450

In the training case, the control panel CP6607 is included. A control panel is not needed for the example projects.

For testing the example project some inputs require to be connected as follows:


Fig. 4: Wiring of training case

For the example projects, not all input switches and none of the potentiometers included in training case are necessary.
You will need switch I1 for the example project for central I/O expansion.
You will need switch 15 for the example project for remote I/O expansion.

### 3.3.2 System assembly, construction and connection

## - NOTICE! <br> Avoidance of electrostatic charging

PLC devices and equipment are sensitive to electrostatic discharge, which can cause internal damage and affect normal operation. Observe the following rules when handling the system:

- Touch a grounded object to discharge potential static.
- Wear an approved grounding wrist strap.
- Do not touch connectors or pins on component boards.
- Do not touch circuit components inside the equipment.
- If available, use a static-safe workstation.
- When not in use, store the equipment in appropriate static-safe packaging.

You can mount AC500 PLC either to DIN rail or to a metal plate. Here, we recommend to mount on DIN rail.

1. Snap the terminal base onto DIN rail.
2. Snap the additional terminal units for I/O modules onto DIN rail.
3. Make the sensor/actuator wire connections according to the dedicated electronic module you want to use. Provide external process power supply as required.
4. If required, make the fieldbus connections according to the dedicated master communication module you want to use.
5. Plug the appropriate electronic and I/O modules in the correct locations (processor module, communication modules on terminal base, and eventually also communication interface modules and I/O modules onto dedicated terminal units).
6. Connect a programming cable (Ethernet cable between ETH port of CPU and PC with engineering software).

### 3.4 Example project for central I/O expansion

### 3.4.1 Purpose

The following steps show how to set-up an application project and configure the hardware. A simple logic is used as example to introduce in programming and commissioning of the PLC. The workflow for creation of a visualization is explained, as well as how to set-up a web server for visualization.

### 3.4.2 Preconditions

- Automation Builder is installed and licensed as, at least, basic edition ${ }^{\wedge}$ Chapter 6.2.4 "Managing your licenses" on page 1350.
- AC500 V3 CPU is assembled and connected to the PC $\Leftrightarrow$ Chapter 3.3 "Hardware AC500 V3" on page 21.


### 3.4.3 Creation, setting up and saving of your AC500 V3 project

### 3.4.3.1 Creation of a project

1. Launch Automation Builder either out of the desktop icon or out of the Windows menu.

2. Select "New Project" or go to menu "File $\rightarrow$ New Project".

3. Select "Projects".
4. Select "AC500 project".
5. Fill in project name.
6. Choose a location to save the project to.
7. Select [OK].
8. Select "PLC - AC500 V3".
9. Select the CPU according to your hardware set-up.

10. Select [Add PLC] to add the CPU to your application.

## 3．4．3．2 Configuration of your processor module

|  | 30-2ETH - TB5620-2ETH) |  |  |
| :---: | :---: | :---: | :---: |
| Devices | －$\square^{\text {x }}$ | ［1］PLC＿AC500＿V3 $\times$ |  |
| －-9 ACSOO V3 Basic Project <br> －国 PLCLogic <br> $=0$ Application值 Libracy Manager目 PLCPRG（PRG） <br> －Task Configuration $=$ es Task <br> （1）PLCPRG <br> 10 Bus <br> －Interfaces <br> $-\mathrm{COM}_{2} 1\left(\mathrm{COM}_{1}\right)$ <br>  1．CAN（\＆Emptr＞） <br> $=$ 㗊 Ethernet <br> － 0 EाH1（PP Settings） （10）NetConfg（Vet <br> － 0 ETH2（IP Settings） （11）NetConfg＿1 10 （13）Protocols（Clent Pro <br> －㽧 Extension＿bus K．Slot＿1（＜Empty＞） <br> K． $\operatorname{slot} 2$（＜Enotr＞） | 620－2ETH） <br> Serial（Com） <br> 9） <br> nfig） <br> （s） | Communication Settings <br> PLC Settings <br> Version information <br> Statistic <br> Files <br> Log <br> PLC Shell <br> Users and Groups <br> Access Rights <br> Symbol Rights <br> PM5630－2ETH Hardware <br> CPU Parameters |  |
| Parameter <br> Error LED | Type <br> Enumeration of BYTE | Value On | Default Value On |
| －Check battery | Enumeration of BYTE | On | On |
| －Stop on error class | Enumeration of BYTE | Diagnosis of at least error class 2 | Diagnosis of at least error dass 2 |

1．Double－click＂PLC＿AC500＿V3＂．
$\Rightarrow A$ tab opens in the editor view．
2．Select＂CPU Parameters＂．
3．Under parameter＂Check battery＂，choose the value＂Off＂since there is no battery present inside the CPU module．

4．Keep the default values for all other parameters．

## 3．4．3．3 Creation of folders in the device tree

To optimize the project readability，you will create different folders to group similar objects．The folder names are exemplary．Because the device tree view follows an alphabetical order，we use number prefixes to determine the order．


1. Right-click "Application".
2. Select "Add Folder".

3. Type in "10 POUs". This is a name example. Here, the intention is to see this folder as a last one.

The folder "10 POUs" is for program organization units (POU). POUs are objects of type program, function or function block that are used to create a user program.

### 3.4.3.4 Saving the project

de Project.project** - Automation Builder

| File | Edit View | Project | Build | Online | Debug |
| :---: | :---: | :---: | :---: | :---: | :---: |
| , | New Project... |  |  | Ctr |  |
| 2 | Open Project... |  |  | Ctrl +O |  |
|  | Close Project |  |  |  |  |
| - | Save Project |  |  |  |  |
|  | Save Project As... |  |  |  |  |
|  | Project Archive |  |  |  | * |
|  | Source Upload... |  |  |  |  |
|  | Print... |  |  |  |  |
|  | Page Setup... |  |  |  |  |
|  | Recent Projects |  |  |  | * |
|  | Exit |  |  | Alt + F4 |  |

$\triangleright \quad$ Select menu "File $\rightarrow$ Save Project".
Alternatively, select the save icon 回 in the tool bar.
Alternatively, press [Ctrl] + [S].

### 3.4.4 Configuration of the I/O module

### 3.4.4.1 General

- The types and order of modules in the Automation Builder project must match the real hardware configuration.
- The position of the modules in the device tree can be changed by drag and drop.


### 3.4.4.2 Adding an I/O bus module



1. Right-click "IO_Bus" in the device tree.
2. Select "Add object".

| Add object below ：IO＿Bus |  |  |  | $\times$ |
| :---: | :---: | :---: | :---: | :---: |
| Object path： |  |  |  |  |
| PLC＿AC500＿V310＿Bus |  |  |  |  |
| Object name：DA501 |  |  |  |  |
| Categories | Search object name．．． |  |  |  |
| $\$ 500 \mathrm{eCo}$／／O modules <br> S500 I／O modules <br> Scripting <br> Uncategorized | Name Short Description |  | Version | Order $\mathrm{N}^{\wedge}$ |
|  | 䉰AC522 | 8AC．U／I／RTD． 12 Bit＋sign | 3．2．0．0 | 1SAP2! |
|  | 圌A1523 | 16AI，U／／／RTD．12Bit＋sign | 3．2．1．0 | 1SAP2! |
|  | 圌A1531 | 8AI，U／I／RTD／Thermo，16Bit＋sign | 3．2．1．0 | 1SAPx 5 |
|  | 管A0523 | 16AO，U／I，12bit＋sign | 3．2．1．0 | 1SAPx5 |
|  | ［17．1） C 521 | 4AI／4AO，U／／RTD，12bit＋sign | 3．2．1．0 | 1SAPxE |
|  | 而AX522 | 8AI／8AO，U／I／RTD， 12 bit＋sign | 3．2．1．0 | 1SAPx： |
|  | － 1 AX523 | 8AI／8AO，U／／／RTD， 12 bit＋sign | 3．2．0．0 | ？ |
|  | －${ }^{1}$ CD522 | 2 2xencoder．2xPWM | 3.2 .10 | 1SAPXE |
|  | DA501 | 16DI／8DC／4AI／2AO．24VDC | 3．2．1．0 | 1SAPx 5 |
|  | －DA502 | 16DO／8DC／4AI／2AO，24VDC | 3．2．1．0 | 1SAPx5 |
|  | － 1 DC522 | 16DC，24VDC | 3．2．1．0 | 1SAPx4 |
|  | 閘DC523 | 24DC，24VDC | 3．2．1．0 | 1SAPx 4 |
|  | －DC532 | 16DI／16DC．24VDC | 3．2．1．0 | 1SAPx4 |
|  | 面D1524 | 32DI，24VDC | 3．2．1．0 | 1SAPx4 |
|  | 四D0524 | 32D0．24VDC | 3．2．1．0 | 1SAPx4V |
|  | $<$ |  |  | ＞ |
| $\square$ Close this dialog after each transaction $\square$ Display all versions |  |  |  |  |
| Reset filter |  | Add object |  | Close |

3．Select＂S500 I／O modules＂．
4．Select＂DA501＂module．
5．Select［Add object］to add the module to the I／O bus．

### 3.4.4.3 Variable mapping of the DA501



| (6) DAS01 $\times$ |  |  |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Diagnosis <br> DA501 Parameters | Find | Filter Show all |  | - + Add F | r 10 chann | ${ }^{+}$Go | Stance |  |  |
|  | $\begin{aligned} & \text { Variable } \\ & +* p \end{aligned}$ |  | Mapping | Channel <br> Digital inputs DI0 - D17 | Address \%IBO | Type BYTE | Default Value | Unit | Description |
| DA501 I/O Mapping | $\begin{aligned} & +x \\ & +x \end{aligned}$ |  |  | Digital inputs DI8 - DI15 | \%181 | bite |  |  |  |
|  |  |  |  | Analog input Al0+ | \%IW1 | $\mathbb{I N T}^{\text {I }}$ |  |  |  |
| DA501 IEC Objects | + * |  |  | Analog input AI1+ | \%IW2 | $\mathrm{INT}^{\text {N }}$ |  |  |  |
|  | $\pm *$ |  |  | Analog input AI2+ | \%w3 | $\mathbb{I N T}^{\text {d }}$ |  |  |  |
| I/O mapping list | + \% |  |  | Analog input AI3+ | \%1w4 | INT |  |  |  |
| Information | 曾 |  |  | Analog output AOO + | \%Qwo | $\mathbb{I N T}^{\text {N }}$ |  |  |  |
|  |  |  |  | Anslog output AO1+ | \%QW1 | INT |  |  |  |
|  | $\begin{aligned} & +x_{0} \\ & +x_{0} \end{aligned}$ |  |  | Digital inputs DC16-DC23 | \%1810 | bite |  |  |  |
|  |  |  |  | Digital outputs DC16-DC23 | \%Q86 | BMTE |  |  |  |
|  | - Fast counter |  |  |  |  |  |  |  |  |

1. Double-click "DA501" in the device tree.
$\Rightarrow$ A tab opens in the editor view.
2. Select "DA501 I/O Mapping"
$\Rightarrow$ Here, you will map variable names (symbols) for the channels you will need in the program.

The suggested name convention is based on "Hungarian notation". A name prefix is describing variable type: e.g., "x" = variable of type BOOL, "w" = WORD, "i" = INT (integer) etc. This increases the code readability and is helpful for program analysis.

### 3.4.4.4 Handling the digital input variables



1. Open the list of the digital inputs.
2. Fill in the variable names:

| Channel | Type | Variable |
| :--- | :--- | :--- |
| Digital input DI8 | BOOL | xDI_08_DA501_I1 |

### 3.4.4.5 Handling the digital output variables

| (0) DA501 $\times$ |  |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Diagnosis <br> DA501 Parameters | Find | Filter Show all |  |  | - \# Add FB for 10 channel... + Go to instance |  |  |  |
|  | Variable $\pm *$ | Mapping | Channel <br> Digital inputs DIO - DI7 | Address <br> \%IB0 | Type <br> BYTE | Default Value | Unit | Description |
| DA501 I/O Mapping | $\begin{aligned} & f=x \\ & t \quad x, \end{aligned}$ |  | Digital inputs DI8 - DI15 | \%IB1 | brte |  |  |  |
|  |  |  | Analog input AIO+ | \%W1 | INT |  |  |  |
| DA501 IEC Objects | + $x^{0}$ |  | Analog input Al1+ | \%w2 | INT |  |  |  |
|  | + ${ }^{\circ}$ |  | Analog input A12+ | \%rw3 | INT |  |  |  |
| I/O mapping list | + |  | Analog input A13+ | \%IW4 | INT |  |  |  |
|  | $*$ |  | Analog output AOO+ | \%QW0 | INT |  |  |  |
| Information |  |  |  | \%QW1 | INT |  |  |  |
|  | $\begin{array}{r} k \\ +\quad x_{p} \\ \hline \end{array}$ |  | Digital inputs DC16-DC23 | \%IB10 | BYTE |  |  |  |
|  | $\frac{+x}{0 *}$ |  | Digital outputs DC16 - DC23 | \%QB6 | BYTE |  |  |  |
|  | * xStartDrillinal | - | Digital output DC16 | \%Qx6.0 | BOOL |  |  |  |

1. Open the list of the digital outputs.
2. Fill in the variable names:

| Channel | Type | Variable |
| :--- | :--- | :--- |
| Digital output DC16 | BOOL | xStartDrilling1 |

### 3.4.5 Programming and compilation

### 3.4.5.1 Task configuration

A task is a time unit in the processing of a user program (IEC application), which defines by parameters the way and the speed the CPU is executing the user program.
For this project you will use only one cycling task.

In the device tree, you see the objects "Task configuration" and "Task". Both created automatically with the project.
For this project you will use only one cycling task.

- Double-click "Task" in the device tree.
$\Rightarrow A$ tab opens in the editor view.
For this project you will use only one cyclic task. Keep the default settings for the task.


Priority This is how the CPU prioritizes the task, when more than one task is defined. Priority $0 \ldots 15=$ real time tasks, priority $16=$ non-real time task.
Type In the CPU you can run tasks dependent on the demands of the process
Interval For cyclic tasks you can set the cyclical execution time. It is usually set in milliseconds with IEC time syntax
Watchdog To keep track of the time it takes to complete the task
Calls You can call in one or more program POUs in one single task

### 3.4.5.2 Main program PLC_PRG

In the default task configuration, there is one call of a POU (program organization unit) i.e. "PLC_PRG" $\Downarrow y$ (shown in chapter 3.4.5.1 Task configuration on page 33).
In your project the "PLC_PRG" will become a main program containing calls to other programs (POUs) which you will create one by one.


The PLC_PRG POU has been defined by default in ST (Structured Text) editor. Keep this setting because of good visibility of the instructions at a glance and good handling for troubleshooting.

To optimize the project readability, you will work with the previously created folder "10 POUs" and add the created subroutines (POUs) to this folder. The subroutines will be created in FBD (Function Block Diagram) editor.

### 3.4.5.3 Boolean logic "NOT"

### 3.4.5.3.1 Application example "driller"

Recognizing of a driller by a photo sensor. "TRUE" input signal from sensor indicates that a driller is broken. If driller has been found correct, then start drilling.


Table 2: Required behavior

| Signal from photo sensor | Required signal of motor ON |
| :--- | :--- |
| FALSE | TRUE |
| TRUE | FALSE |

Table 3: Hardware set-up

| Element | HW channel | Symbol | Description |
| :--- | :--- | :--- | :--- |
| Switch I1 | DA501 DI8 | xDI_08_DA501_I1 | Photo sensor |
| LED output DC16 | DA501 DC16 | xStartDrilling1 | Motor on |

### 3.4.5.3.2 Implementation

## Creation of a new program POU in the project



1. Right-click " 10 POUs".
2. Select "Add object".
3. Select "POU".
4. Select [Add object].

5. Enter "_01_Assignment_NOT".
6. Select "Program".
7. Select "Function Block Diagram (FBD)".
8. Select [Add].
$\Rightarrow$ POU has been added.

## Assigning the hardware DI signals to local variables

1. Double-click POU"_01_Assignement_NOT" in the device tree.

2. Select "Assignment" from the ToolBox.
3. Drag and drop "Assignment" into the "Start here" field in network "1".

4. Select "???" on the left side of the assignment, then select "...".
5. Open the "lo Config_Globals_Mapping" mapping list and select "xDI_08_DA501_I1".
6. Select $[O K]$ to add this variable to the left side of the assignment connector.

7. Select "???" on the right side of the assignment connector and mark the "???".
8. Create a new local variable by typing in "xDrillerBroken1" which will replace the "???".
9. Press [Enter].
$\Rightarrow$ "Auto Declare" opens.
You see the written variable name and the data type BOOL. The scope is "VAR". It means it is a local variable within this POU.
10. Select $[O K]$ to accept the entries.

11. Drag and drop "Network" from the ToolBox to the down-arrow of network 1.
$\Rightarrow$ You added a network "2" below network 1.

## Adding assignments and a Boolean NOT to the DO signals

1. Add an assingment from the ToolBox.
2. Type in or copy \& paste "xDrillerBroken1" to the left side of the instruction line.
3. Select "???" on the right side of the instruction line, then select "...".
$\Rightarrow$ "Input Assistant" opens.

4. In the "IoConfig_Globals_ Mapping" variable list, select "xStartDrilling1".
5. Select $[O K]$ to close the dialog.

```
xDrillerBroken1 __xStartDrilling1
```


6. Right-click the center of assignment PIN.
7. Select "Negation" to add a negation to the assignment.


```
xDrillerBroken1 -- xStartDrilling1
```


## Calling the POU in the PLC_PRG



1. Double-click "PLC_PRG".
2. Select the first line in "PLC_PRG" and press [F2].
$\Rightarrow$ "Input Assistant" opens.

3. Select "Module Calls".
4. Open "Application".
5. Open "10 POUs" and select "_01_Assignment_NOT".
6. Select $[O K]$ to close the dialog.


## 3．4．5．3．3 Compilation of the project

Before logging－in to the CPU，you need to compile the complete code without any errors．

$\triangleright$ Select menu＂Build $\rightarrow$ Generate code＂．
$\Rightarrow$ The result of the compiling is shown in the＂Messages＂field at the bottom of the screen．

If you skip the compiling and select＂Login＂，the Automation Builder will automatically trigger compiling in advance to logging－in．

## 3．4．5．3．4 Saving the project

能 Project．project＊－Automation Builder

| File | Edit View | Project | Build | Online | Debug |
| :---: | :---: | :---: | :---: | :---: | :---: |
| 淮 | New Project．．． |  |  |  |  |
| － | Open Project．．． |  |  | Ctrl＋ 0 |  |
|  | Close Project |  |  |  |  |
| ［ | Save Project |  |  | Ctrl + S |  |
|  | Save Project As．．． |  |  | － |  |
|  | Project Archive |  |  |  |  |
|  | Source Upload．．． |  |  |  |  |
|  | Print．．． |  |  |  |  |
|  | Page Setup．．． |  |  |  |  |
|  | Recent Projects |  |  |  | － |
|  | Exit |  |  | Alt＋F4 |  |

$\triangleright \quad$ Select menu＂File $\rightarrow$ Save Project＂．
Alternatively，select the save icon 回 in the tool bar．
Alternatively，press［Ctrl］＋［S］．

## 3．4．6 Setting up the communication gateway

Setting up of To set up the communication between the PC and the PLC，e．g．，for downloading the compiled communication parameters program，you have to set up the communication parameters．

The IP address of your PC must be in the same class as the IP address of the CPU． The factory setting of the IP address of the CPU is 192．168．0．10．

The IP address of your PC should be 192.168.0.X. Avoid $X=10$ in order to prevent an IP conflict with the CPU.
Subnet mask should be 255.255.255.0.

## Changing of the

 IP address1. Open Windows control panel. Click "Network and Internet
$\rightarrow$ Network and Sharing Center".
2. Click "Change adapter settings".

3. Right-click "Local Area Connection (Ethernet)" and select [Properties].

4. Double-click "Internet Protocol Version 4 (TCP/IPv4)".

5. Enter your desired IP address and subnet mask.

## Setting up the communication gateway

CPU and PC are connected with an Ethernet cable.

1. In the Automation Builder device tree right-click "PLC_AC500_V3".
2. Select "Communication Settings".

3. Keep the default value in the IP address of the CPU or type in the current IP address, if differs.

The standard (default) IP address of the port ETH1 is: 192.168.0.10
4. Select $[O K]$ to implement the IP address.

Network scan If you need to scan the network for the CPU or if you have multiple CPUs on the same network.

1. Right-click "PLC_AC500_V3" in the device tree.
2. Select "Communication Settings".

3. Select "..."
$\Rightarrow$ "Pick IP Address for PLC_AC500_V3" opens.

| Pick IP Address for 'PLC_AC500_V3'.... | - |  |
| :--- | :--- | :--- | :--- | :--- |
| Abort scan $\square$ Extended Scan     <br> MAC address Device name Port Serial number Device ID IP Address <br>  Config. IP Address Device Type    <br>       <br>       |  |  |
| Scarning |  |  |

The automatic scan runs.
The results will appear in this field.
4. Select the CPU in the field and select $[O K]$ to implement the needed communications gateway.

Checking the communication settings

If you need to check the communications settings or if you want to see more information about the current selected CPU.


1. Double-click "PLC_AC500_V3" in the device tree.
2. Select "Communication Settings".
$\Rightarrow$ The selected IP address is shown.
3. If the IP address is not visible, enter the IP address manually.
4. To test the connection and/or to see the CPU information press [Enter] or click on the black dot next to the PLC figure.

### 3.4.7 Installation and update of the AC500 V3 firmware

The multi download tool can be used to update multiple PLCs with firmware or an application at the same time.
« Chapter 6.6.4 "Multi download tool" on page 4306

Firmware The PLC user management will remain also after a firmware update or downgrade.
Due to a structural change in the PLC user management in firmware version 3.5 there are some limitations when updating the PLC firmware from a version 3.4 or lower to 3.5 or higher with activated user management.

If you update the PLC firmware from version 3.4 or lower to version 3.6 or higher this must be done in two steps. First an update to 3.5 must be done via Automation Builder. Afterwards the firmware can be updated to any later firmware version - either via Automation Builder or via memory card.

The PLC firmware can be updated via Automation Builder.

This is also necessary for commissioning V3 CPUs.

A very new CPU has no pre-installed firmware. To guarantee the authenticity of delivered AC500 firmware, V3 CPUs are delivered with a boot loader only. You need to download a valid firmware to the CPU. After download, the functionality of the CPU is given.


An Automation Builder project with an AC500 V3 CPU is open.
$\checkmark \quad$ CPU is in "stop" mode or shows uPdAtE (update) on the display.
$\checkmark \quad$ After update the CPU shows either donE or StoP on the display
$\checkmark \quad$ For new modules: IP address is set. (The default IP address is 192.168.0.10)

1. Double-click CPU "PLC_AC500_V3".
2. Select "Version information".


## 3. Select "Update Firmware".

$\Rightarrow$ While the update process is running, the RUN and ERR LEDs are toggling, i.e., they are flashing alternating.
4. Wait for the PLC to finish the update.


A completed update is indicated by a message on the display. Either donE, or StoP.

## NOTICE!

Do not disconnect the power supply during the update process! The PLC could be damaged.
$\Rightarrow$ StoP indicates a restart has been performed by the CPU. When donE is displayed sometimes it is necessary to re-boot the CPU manually, e.g., by powering-off. Manual re-boot might be, e.g., for some older CPU versions or if downgrading to an older firmware version according to application settings.


The CPU display shows StoP after re-boot. The update process is finished.
5. If necessary, refresh the version information by switching to another tab and back. $\Rightarrow$ Successful firmware update:


Behavior of LEDs during firmware update


| LED | LED flashes | Status |
| :--- | :--- | :--- |
| RUN and ERR | Toggling | Update pending |
| RUN | Flashing slow | Done successful |
| ERR | Flashing slow | Done failed |

### 3.4.8 Logging in to CPU and downloading the program

Logging-in to the CPU will load the project into the AC500 V3 CPU. The first log-in will also load the hardware set-up.


1. In the Automation Builder menu select "Online $\rightarrow$ Login [PLC_AC500_V3]". $\Rightarrow$ A pop-up will appear.
2. Select [Yes] to download the application to the AC500V3 CPU.

$\Rightarrow$ PLC is in "stop" mode.
3. Start the PLC $\triangleq$ Chapter 3.4.9.1 "Starting the program execution" on page 51.

Generally, if the CPU is in RUN mode, i.e. in program execution mode, a download will always cause the mode change to "stop". In stop mode the CPU is not controlling the system!
Always, after selecting the "Login" command, read carefully the dialog box text to ensure that you are aware of the CPU's behavior after the command confirmation.

By default, a download generates following actions in the CPU:

- The project is stored in the RAM memory.
- The project is stored in the flash EEPROM, if boot application was created.


### 3.4.9 Testing the program

### 3.4.9.1 Starting the program execution

$\checkmark \quad$ You are logged in the CPU.An executable project is loaded to the CPU.The CPU is in "stop" mode.

| Debug Iools Window Help |  |
| :--- | :--- | :--- |
| 1 Start [PLC_AC500_V3] | F5 |


$\triangleright \quad$ Select menu "Debug $\rightarrow$ Start [PLC_AC500_V3]".
Alternatively, select the "start" icon in the tool bar.
Alternatively, press [F5].

### 3.4.9.2 Testing the function

$\triangleright \quad$ Operate the switch 11 and observe:

- The LEDs of the relevant DA501 inputs and outputs.
- The online status of inputs and outputs within the POU.



### 3.4.9.3 Stopping the program execution

You are logged in the CPU.An executable project is loaded to the CPU.$v$
The CPU is in RUN mode.


Help

$\triangleright \quad$ Select menu "Debug $\rightarrow$ Stop [PLC_AC500_V3]"
Alternatively, select the "stop" icon in the tool bar.
Alternatively, press [Shift] + [F8].

### 3.4.10 Setting up a visualization

### 3.4.10.1 General

If you are not yet familiar with CODESYS visualization, we recommend you to start with the application example First steps with CODESYS visualization. The example demonstrates the main features of visualization and provides insights into possible use cases.

To use the Edge browser for CODESYS visualization of AC500 PLCs, follow the steps described in the application note Usage in the Edge browser.

### 3.4.10.2 Adding the VisualizationManager



1. Right-click "Application" in the device tree.
2. Select "Add object".
3. Select "VisualizationManager".
4. Select [Add object] to add the VisualizationManager to the project.
$\Rightarrow$ Dialog "Add Visualization Manager" opens.

| Add Visualization Manager |
| :--- |
| Name: |
| VisualizationManager |
| Symbol libraries $\quad$ Active |
| VisualizationManager |
| A visualization symbol library is a CODESYS library with <br> graphics and graphical objects. If the visualization symbol <br> library is assigned the library is added into the POUs library <br> manager. The graphics and graphical objects are shown in the <br> toolboxwhen a visualization editor is the active editor. |

5. Select [Add].

$\Rightarrow$ You added the objects "VisualizationManager" and "VISU-TASK" to the device tree.

### 3.4.10.3 Settin up the VisualizationManager



1. Double-click VisualizationManager in the device tree.
$\Rightarrow A$ tab opens in the editor view.



2．Select＂Settings＂．
3．Open the drop－down menu＂Selected style＂．
4．Select＂Default，x．x．x＂（exemplary）．
5．Open the drop－down menu＂Selected language＂．
6．Select＂en＂for English language in the visualization．
7．Enable＂Visible＂for advanced settings．
8．Keep the file transfer to enable the visualization on the PLC（mandatory for web server function）« Chapter 3．4．12＂Enabling a web visualization＂on page 65.

## 3．4．10．4 Saving the project

能 Project．project＊－Automation Builder

| File | Edit View | Project | Build | Online | Debug |
| :---: | :---: | :---: | :---: | :---: | :---: |
| 洤 | New Project．．． |  |  | $\mathrm{Ctrl}+\mathrm{N}$ |  |
| 年 | Open Project．．． |  |  | $\mathrm{Ctrl}+\mathrm{O}$ |  |
|  | Close Project |  |  |  |  |
| ［ | Save Project |  |  | Ctrl + S |  |
|  | Save Project As．．． |  |  |  |  |
|  | Project Archive |  |  |  | ＊ |
|  | Source Upload．．． |  |  |  |  |
|  | Print．．． |  |  |  |  |
|  | Page Setup．．． |  |  |  |  |
|  | Recent Projects |  |  |  | － |
|  | Exit |  |  | Alt＋F4 |  |

$\triangleright \quad$ Select menu＂File $\rightarrow$ Save Project＂．
Alternatively，select the save icon 回 in the tool bar．
Alternatively，press［CtrI］＋［S］．

### 3.4.11 Creation of a visualization

### 3.4.11.1 General

If you are not yet familiar with CODESYS visualization, we recommend you to start with the application example First steps with CODESYS visualization. The example demonstrates the main features of visualization and provides insights into possible use cases.

To use the Edge browser for CODESYS visualization of AC500 PLCs, follow the steps described in the application note Usage in the Edge browser.

### 3.4.11.2 Adding a folder for visualization screens



1. Right-click "Application" in the device tree.
2. Select "Add Folder".

3. Type in "02 VISUs".
4. Select $[O K]$ to add the folder.

### 3.4.11.3 Adding a screen for "_01_Assignment_NOT" POU



1. Right-click "02 VISUs".
2. Select "Add object".
3. Select object "Visualization".
4. Select [OK].

5. Type in "PLC_VISU".
6. Select [Add].
$\Rightarrow A$ tab opens in the editor view.


Fig. 5: PLC_VISU_tab

The name "PLC_VISU" has been chosen, because it is the default name for a home screen in a web visualization.
If you have more than one visualization object in your project, it will be useful to choose another name, e.g. "_01_Assignment_NOT_v". And to choose "PLC_VISU" as a home screen to access all available visualization screens.
The name of a visualization object can be modified afterwards.

### 3.4.11.4 Creation and configuration of a visualization

### 3.4.11.4.1 Changing the background color

1. Double-click "PLC_VISU" in the device tree.
$\Rightarrow A$ tab opens in the editor view.

2. Right-click anywhere on the "PLC_VISU" editor page.
3. Select "Background".

4. Enable the check box "Use Color".
$\Rightarrow$ This enables the drop-down menu.
5. Select a color, e.g., "Lightgray".
6. Select $[O K]$ to add the color to "PLC_VISU".

### 3.4.11.4.2 Adding a screen title

1. Double-click on "PLC_VISU" in the device tree.

2. Select "ToolBox".

3. Select "Common controls".
4. Drag and drop "Label" to the page.

5. Type in "Start drilling condition".

### 3.4.11.4.3 Further lines and labels

1. Double-click on "PLC_VISU" in the device tree.

2. Select "ToolBox".
3. Select "Basic".
4. Drag and drop the line. Then drag the line to the needed length.

5. Follow the same procedure to create the other shapes and labels.

### 3.4.11.4.4 Lamp element for signal indication

1. Double-click on "PLC_VISU" in the device tree.

2. Select "ToolBox".
3. Select "Lamps/Switches/Bitmaps".
4. Drag and drop "Lamp" to the screen.
5. Adapt the size, if required.

6. Under "Image", select "Gray".

| Property <br> Element name | Value <br> GenElemInst_2 |
| :--- | :--- |
| Type of element | Lamp |

7. Double-click on "Variable" and select "..." to select a variable from the list.

8. Under "loConfig_Globals_Mapping", select "xStartDrilling1".
9. Select [OK].

### 3.4.11.4.5 Compilation of the project

Before logging-in to the CPU, you need to compile the complete code without any errors.

$\triangleright$ Select menu＂Build $\rightarrow$ Generate code＂．
$\Rightarrow$ The result of the compiling is shown in the＂Messages＂field at the bottom of the screen．

If you skip the compiling and select＂Login＂，the Automation Builder will automatically trigger compiling in advance to logging－in．

## 3．4．11．4．6 Saving the project

能 Project．project＊－Automation Builder

| File | Edit View | Project | Build | Online | Debug |
| :---: | :---: | :---: | :---: | :---: | :---: |
| 脜 | New Project．．． |  |  |  |  |
| － | Open Project．．． |  |  | Ctrl＋ 0 |  |
|  | Close Project |  |  |  |  |
| ［ | Save Project |  |  |  |  |
|  | Save Project As．．． |  |  |  |  |
|  | Project Archive |  |  |  | － |
|  | Source Upload．．． |  |  |  |  |
|  | Print．．． |  |  |  |  |
|  | Page Setup．．． |  |  |  |  |
|  | Recent Projects |  |  |  | ＊ |
|  | Exit |  | Alt＋F4 |  |  |

$\triangleright \quad$ Select menu＂File $\rightarrow$ Save Project＂．
Alternatively，select the save icon 回 in the tool bar．
Alternatively，press［Ctrl］＋［S］．

## 3．4．11．5 Loading the project to the CPU

1．Download the project to the CPU $\Longleftrightarrow$ ，as described in Chapter 3．4．8，on page 50.
2．Check the notification window at the end of the download．In case of message＂Boot parameters were changed．These changes will be applied after reboot＂，a reboot of the CPU is required after creation of the boot project．


### 3.4.11.6 Testing the program

$\triangleright \quad$ Operate the switches and observe the visualization screen.


### 3.4.12 Enabling a web visualization

### 3.4.12.1 Adding a web server object to the device tree

Ethernet ports can be configured for web server protocol. This description deals with ETH1 configuration for the web server


1. Right-click "ETH1" in the device tree.
2. Select "Add object".
3. Select "Web Server".
4. Select [Add object].
$\Rightarrow$ You added and activated a web server on Ethernet port 1 on the AC500 V3 CPU.

### 3.4.12.2 Setting up the web server



1. Double-click "WebVisu" in the device tree.

2. Under "Start Visualization", select "...".
$\Rightarrow$ A list opens.
3. Select the "PLC_VISU" screen from the list.
4. Keep all further settings with default values.

5. Select the link "Show used visualizations".

$\Rightarrow$ The VisualizationManager editor and there the tab "Visualizations" opens. All screens and dialog elements created in the project are visible.
Here, you can select which screens are enabled or disabled for web visualization.

If you want to select another screen as a start visualization, you must modify the adequate parameter in the webvisu.htm file: <param name="STARTVISU" value="PLC_VISU">


### 3.4.12.3 Compilation of the project

Before logging-in to the CPU, you need to compile the complete code without any errors.

$\triangleright$ Select menu＂Build $\rightarrow$ Generate code＂．
$\Rightarrow$ The result of the compiling is shown in the＂Messages＂field at the bottom of the screen．

If you skip the compiling and select＂Login＂，the Automation Builder will automatically trigger compiling in advance to logging－in．

## 3．4．12．4 Saving the project

能 Project．project＊－Automation Builder

| File | Edit View | Project | Build | Online | Debug |
| :---: | :---: | :---: | :---: | :---: | :---: |
| 脜 | New Project．．． |  |  |  |  |
| － | Open Project．．． |  |  | Ctrl＋ 0 |  |
|  | Close Project |  |  |  |  |
| ［ | Save Project |  |  |  |  |
|  | Save Project As．．． |  |  |  |  |
|  | Project Archive |  |  |  | － |
|  | Source Upload．．． |  |  |  |  |
|  | Print．．． |  |  |  |  |
|  | Page Setup．．． |  |  |  |  |
|  | Recent Projects |  |  |  | ＊ |
|  | Exit |  | Alt＋F4 |  |  |

－Select menu＂File $\rightarrow$ Save Project＂．
Alternatively，select the save icon 回 in the tool bar．
Alternatively，press［Ctrl］＋［S］．

## 3．4．12．5 Loading the project to the CPU

1．Download the project to the CPU $\leftrightarrows$ as described in Chapter 3.4 .8 ，on page 50.
2．Check the notification window at the end of the download．In case of message＂Boot parameters were changed．These changes will be applied after reboot＂，a reboot of the CPU is required after creation of the boot project．

## Automation Builder

Notification from target:
Notification from target:
"Boot parameters were changed. These changes will be applied after reboot. CAUTION: Please create boot project before !!!"

### 3.4.12.6 Creation of a boot project

By default, after project download, the boot project is created automatically.

### 3.4.12.7 Reboot of the CPU

$\triangleright \quad$ Reboot the CPU by switching OFF and ON the power supply. (The parameter for web server activation is a boot pamater which is loaded during boot of the CPU)

### 3.4.12.8 Testing the web visualization

$\checkmark$ You have downloaded the project and created the boot project.
$\checkmark$ The CPU has been rebooted.You are logged in.CPU is in "stop" mode.

1. Start the project execution, e.g., from the tool bar.
2. Launch an internet browser.
3. Type in the URL field: http://192.168.0.10/webvisu.htm. 192.168.0.10 is the IP address of CPU's ETH1 port.
/webvisu.htm is the default htm file.
$\Rightarrow$ Web visualization will be loaded.


The start screen "PLC_VISU" is displayed in a responsive view.

4. Test the function by operating switch I1.
5. Test the results for responsive view by changing the web browser window size.

### 3.4.13 Reset the CPU

Reset values and parameters

In some cases, it could be required to do a CPU reset, e.g., for resetting of counter values, parameters etc.

| Online | e Debug Iools Window | Help |
| :---: | :---: | :---: |
| 08 | Login [PLC_AC500_V3] | Alt+F8 |
| ¢ | Logout [PLC_AC500_V3] | Ctri+F8 |
|  | Create boot application [PLC_AC500_V3] |  |
| $\otimes$ | Logoff current device user |  |
|  | Download |  |
|  | Online Change |  |
|  | Source download to connected device |  |
| ***ㅜํ) | Download Manager... |  |
|  | Reset warm [PLC_AC500_V3] |  |
|  | Reset cold [PLC_AC500_V3] |  |
|  | Reset origin [PLC_AC500_V3] |  |

Fig. 6: Reset commands in "Online" menu
Reset All variables are reset, except RETAIN PERSISTENT variables.
warm
Reset cold Causes initialization of all variables, except PERSISTENT variables. By recommended creation of remanent variables always with both properties: PERSISITENT and RETAIN, this command resets all variables, except PERSISTENT RETAIN variables.
Reset All variables and the application project are reset.
origin

Table 4: Behavior of variables of type VAR (local or global) and variables of type PERSISTENT RETAIN

|  | VAR | VAR PERSISTENT RETAIN |
| :--- | :--- | :--- |
| After online command 'Online change' | no change | no change |
| After online command 'Download' | initialization | no change |
| After online command 'Reset warm' | initialization | no change |
| After online command 'Reset cold' | initialization | no change |
| After online command 'Reset origin' | initialization | initialization |
| After power supply off | initialization | no change |

[^1]

1. Right-click the station object "PLC_AC500_V3" in the device tree.
2. Select "Reset origin device [station name]".
$\Rightarrow$ The application is completely erased from the CPU (complete project from all memory areas).

### 3.5 Example project for remote I/O expansion with PROFINET

### 3.5.1 Purpose

This example introduces the configuration of the PLC with remote I/O. The use of I/O channels in a program and commissioning of the configuration is shown.

### 3.5.2 Preconditions

- Automation Builder is installed and licensed as, at least, standard edition $\Leftrightarrow$ Chapter 6.2.4 "Managing your licenses" on page 1350.
- AC500 V3 CPU is assembled and connected to the PC $\Leftrightarrow$ Chapter 3.3 "Hardware AC500 V3" on page 21.
- Configuration and programming of this example project will be made in the existing example project for central I/O expansion $\stackrel{y}{ }$ Chapter 3.4 "Example project for central I/O expansion" on page 24.
- CM579-PNIO communication module is inserted in terminal base and connected to the PLC (7) Chapter 3.3 "Hardware AC500 V3" on page 21.
- CI502-PNIO communication interface module is inserted in terminal unit and connected to the PLC ${ }^{\circledR}$ Chapter 3.3 "Hardware AC500 V3" on page 21.


## 3．5．3 Set－up PROFINET controller

## 3．5．3．1 Add the CM579－PNIO to the device tree

1．In the Automation Builder device tree under＂Extension＿Bus＂，right－click＂Slot＿1＂．
2．Select＂Add object＂．
3．Select＂CM579－PNIO＂．
4．Select［Replace object］to add the CM579－PNIO．


3．5．3．2 Set－up the general behavior

```
Devices
- 目 AC500 V3 Basic Project
    - -T. PLC_AC500_V3 (PM5630-2ETH - TB5620-2ETH)
        filf PLC Logic
    + [酉 IO_Bus
```



```
    邱拃 Extension_Bus
        - [5] CM579_PNIO (CM579-PNIO)
        &40] PNIO_Controller (PROFINET-IO-Controller)
    L Slot_2 (<Empty>)
```

1．Under＂Extension＿Bus＂，double－click＂CM579＿PNIO＂in the device tree．
$\Rightarrow A$ tab opens in the editor view．

## 2．Select＂CM579－PNIO Parameters＂．



Run on configuration This parameter will prohibit the PLC from running if the CM579－PNIO fault has a configuration error．
Bus behavior This parameter sets how the data from the bus flows in／out of the CM579－PNIO．


3．Select＂Status＂．
$\Rightarrow$ This opens the bus controller status and gives a basic status overview．
4．Select＂Information＂．
$\Rightarrow$ This page contains general information about the CM579－PNIO．
5．For the example project，you can keep the default settings．

## 3．5．3．3 Set－up the PROFINET IO controller

$\checkmark \quad$ To edit settings for the controller，you must not be logged－in to the PLC．

```
Devices
- FT AC500 v3 Basic Project
    - FI PLC_AC500_V3 (PM5630-2ETH - TB5620-2ETH)
        + 自! PLC Logic
        + 目 IO_Bus
        + 垍昷 Interfaces
        40] Extension_Bus
        - 朁 CM579_PNIO (CM579-PNIO)
```



```
            {. Slot_2 (<Empty>)
```

1．Under＂CM579＿PNIO＂，double－click＂PNIO＿Controller＂in the device tree．
$\Rightarrow A$ tab opens in the editor view．
2．Select＂PROFINET IO CONTROLLER＂

3. Select "General".
4. Here, you can set-up the way, IP addresses are distributed out to the industrial bus network. You can even set, what IP-address and DNS name (station name) the PROFINET controller has.
For the example project, keep the default settings.

### 3.5.4 Set-up PROFINET device

### 3.5.4.1 Hardware preparation

1. Switch off the power supply of your PLC.

2. Use a screw driver to set the Cl 502 module address to "02" by positioning of the upper rotary switch to "0" and lower switch to "2". Note, that the numbers have hexadecimal format.
3. Switch on the power supply.

### 3.5.4.2 Add the CI502-PNIO to the device tree

1. Right-click "PNIO_Controller" in the device tree.
2. Select "Add object".
3. Select "CI502-PNIO-Device".

4．Select［Add object］to add the device．


## 3．5．4．3 Configure the CI502－PNIO device

## 3．5．4．3．1 Configure the CI502－PNIO PROFINET IO device

## Devices

－－ 5 AC500 v3 Basic Project
－- II PLC＿AC500＿V3（PM5630－2ETH－TB5620－2ETH）

+ 目折 PLC Logic
＋ －i0＿Bus
$\pm$ Interfaces
－蚼 Extension＿Bus
－ 4 C0 0 CM579＿PNIO（CM579－PNIO）
- 铇 PNIO＿Controller（PROFINET－IO－Controller）
- 中D CI502＿PNIO＿Device（CI502－PNIO－Device）

CI502＿IO（CI502－IO）
K．Slot＿2（＜Empty＞）
1．Double－click＂Cl502＿PNIO＿Device＂．
$\Rightarrow A$ tab opens in the editor view．

## 2. Select "General".



## User-Defined Parameters

In Set all default values

| Parameters | Value | Datatype | Allowed values | Description |  |
| :--- | :--- | :--- | :--- | :--- | :--- |

Station name Default station name
IP Parameter IP-addressing parameters of the node. If modifications are required for "IP Parameter", they must be done also for CM579-PNIO and all other devices in this PROFINET line.
Communication Communication time set-up
VLAN Virtual local area network ID
RT Class PROFINET IO RT (real time) type settings

3. Set station name to e.g. "ci502-pn-0b" according to hardware settings.

The last two values of the node's "Station Name" in Automation Builder correspond to the position of module's rotary switches (hexadecimal values): e.g., "ci502-pn-0a" or "ci502-pn-10".
$\Rightarrow$
Use small letters for the station name and not large ones.
4. Leave the default settings for "IP Parameter".
5. Adjust the communication time settings to get a Watchdog (ms) 24:

- "Send clock (ms)": 4
- "Reduction ratio": 2
- "Phase": 1

6. Leave the default settings for "VLAN ID".
7. Leave the default settings for "RT Class".

If the node has the same device address (the last two digits of the device name) as set by means of the rotary switches on the module, all the node parameters will be loaded automatically upon initialization scan of the Cl50x module. This allows, e.g., the module exchange without an engineering tool.

### 3.5.4.3.2 Create CI502-PNIO I/O mapping to symbols



1. Double-click "CI502_IO".

| (11) CI502_IO $\times$ |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| General <br> PNIO Module I/O Mapping | Find |  | Filter Show all |  | $\checkmark$ |
|  | Variable $\pm *$ | Mapping | Channel <br> 8 -bit digital input | Address <br> \%IB24 | Type USINT |
| I/O mapping list | - |  | 8 -bit digital input | \%IB25 | USINT |
|  | * XDI_08_CI502_I5 | * | Channel 8 | \%IX25.0 | BOOL |
| Information | ${ }^{*}$ |  | Channel 9 | \%IX25.1 | BOOL |
|  | 4 |  | Channel 10 | \%IX25.2 | BOOL |
|  | * |  | Channel 11 | \%IX25.3 | BOOL |
|  | 4 |  | Channel 12 | \%IX25.4 | BOOL |
|  | ${ }^{4}$ |  | Channel 13 | \%IX25.5 | BOOL |
|  | * |  | Channel 14 | \%IX25.6 | BOOL |
|  | ${ }^{4}$ |  | Channel 15 | \%IX25.7 | BOOL |
|  | * |  | Fast Counter: Actual value 1 | \%ID7 | UDINT |
|  | * |  | Fast Counter: Actual value 2 | \%ID8 | UDINT |
|  | ${ }^{*}$ |  | Fast Counter: State byte 1 | \%IB36 | USINT |
|  | 企 |  | Fast Counter: State byte 2 | \%IB37 | USINT |
|  | $\pm{ }^{ \pm}$ |  | 8-bit digital output | \%QB28 | USINT |
|  | - ${ }^{*}$ |  | 8 -bit digital output | \%QB29 | USINT |
|  | * xDO_08_CI502 | * | Channel 8 | \%QX29.0 | BOOL |
|  |  |  | Channel 9 | \%QX29.1 | BOOL |
|  | * |  | Channel 10 | \%QX29.2 | BOOL |

2. Select "PNIO Module I/O Mapping".
3. Fill in the variable names:

| Element | Hardware channel | Symbol |
| :--- | :--- | :--- |
| Switch I5 | Cl502 DI8 | xDI_08_CI502_I5 |
| LED output DO8 | Cl502 DO 8 | xDO_08_CI502 |

### 3.5.5 Add remote I/O expansion to project

### 3.5.5.1 Add a program POU to the project



1. Right-click "01-POUs" in the device tree.
2. Select "Add object".
3. Select "POU".
4. Select [Add object].

5. Fill in "_30_PNIO_test".
6. Select "Program".
7. Select "Function Block Diagram".
8. Select [Add] to add the POU.

### 3.5.5.2 Create a POU logic

1. Double-click "30_PNIO_test" in the device tree.

2. In the ToolBox, select "Assignment".
3. Drag and drop "Assignment" into the "Start here" field in network "1".

4. Select "???" on the left side of the assignment, then select "...".
5. In "loConfig_Globals_Mapping" list, select "xDI_08_C/502_I5".
6. Select $[O K]$ to add this variable to the left side of the assignment connector.

7. Select "???" on the right side of the assignment, then select "..."
8. In "loConfig_Globals_Mapping" list, select "xDO_08_CI502".
9. Select [OK].


### 3.5.5.3 Call the POU in PLC_PRG

1. Double-click "PLC_PRG".
2. Select the next free line in "PLC_PRG" and press [F2].
$\Rightarrow$ "Input Assistent" opens.
3. Select "Module Calls".
4. Open "Application".
5. Open "10 POUs" and select "_30_PNIO test".
6. Select $[O K]$ to close the dialog.


### 3.5.5.4 Compilation of the project

Before logging-in to the CPU, you need to compile the complete code without any errors.

$\triangleright$ Select menu＂Build $\rightarrow$ Generate code＂．
$\Rightarrow$ The result of the compiling is shown in the＂Messages＂field at the bottom of the screen．

If you skip the compiling and select＂Login＂，the Automation Builder will automatically trigger compiling in advance to logging－in．

## 3．5．5．5 Saving the project

能 Project．project＊－Automation Builder

| File | Edit View | Project | Build | Online | Debug |
| :---: | :---: | :---: | :---: | :---: | :---: |
| 推 | New Project．．． |  |  | Ctrl |  |
| E） | Open Project．．． |  |  | $\mathrm{Ctrl}+\mathrm{O}$ |  |
|  | Close Project |  |  |  |  |
| 回 | Save Project |  |  | Ctrl |  |
|  | Save Project As．．． |  |  | ＊ |  |
|  | Project Archive |  |  |  |  |
|  | Source Upload．．． |  |  |  |  |
|  | Print．．． |  |  |  |  |
|  | Page Setup．．． |  |  |  |  |
|  | Recent Projects |  |  |  | ＊ |
|  | Exit |  |  | Alt＋F4 |  |

$\triangleright \quad$ Select menu＂File $\rightarrow$ Save Project＂．
Alternatively，select the save icon 回 in the tool bar．
Alternatively，press［Ctrl］＋［S］．

## 3．5．5．6 Loading the project to the CPU

1．Download the project to the CPU $\leftrightarrows$ as described in Chapter 3.4 .8 ，on page 50.
2．Check the notification window at the end of the download．In case of message＂Boot parameters were changed．These changes will be applied after reboot＂，a reboot of the CPU is required after creation of the boot project．


### 3.5.6 Test the program

### 3.5.6.1 Starting the program execution

$\square \quad$ You are logged in the CPU.An executable project is loaded to the CPU.The CPU is in "stop" mode.
Debug Iools Window Help

| Start [PLC_AC500_V3] | F5 |
| :--- | :--- | :--- |


$\triangleright \quad$ Select menu "Debug $\rightarrow$ Start [PLC_AC500_V3]".
Alternatively, select the "start" icon in the tool bar.
Alternatively, press [F5].

### 3.5.6.2 Test the function

$\triangleright \quad$ Operate the switch I5 and observe:

- The LEDs of the relevant CI 502 inputs and outputs.
- The online status of inputs and outputs within the POU.



### 3.5.7 Reset the CPU

Reset values and parameters

In some cases, it could be required to do a CPU reset, e.g., for resetting of counter values, parameters etc.

| Online | Debug Iools Window | Help |
| :---: | :---: | :---: |
| ¢ | Login [PLLC_AC500_V3] | Alt+F8 |
|  | Logout [PLC_AC500_V3] | Ctrl+F8 |
|  | Create boot application [PLC_A | C500_V3] |
| $\otimes$ | Logoff current device user |  |
|  | Download |  |
|  | Online Change |  |
|  | Source download to connected device |  |
| 눙) | Download Manager... |  |
|  | Reset warm [PLC_AC500_V3] <br> Resent cold [PLC_AC500_V3] <br> Reset origin [PLC_AC500_V3] |  |

Fig. 7: Reset commands in "Online" menu
Reset All variables are reset, except RETAIN PERSISTENT variables.
warm
Reset cold Causes initialization of all variables, except PERSISTENT variables. By recommended creation of remanent variables always with both properties: PERSISITENT and RETAIN, this command resets all variables, except PERSISTENT RETAIN variables.
Reset All variables and the application project are reset.
origin

Table 5: Behavior of variables of type VAR (local or global) and variables of type PERSISTENT RETAIN

|  | VAR | VAR PERSISTENT RETAIN |
| :--- | :--- | :--- |
| After online command 'Online change' | no change | no change |
| After online command 'Download' | initialization | no change |
| After online command 'Reset warm' | initialization | no change |
| After online command 'Reset cold' | initialization | no change |
| After online command 'Reset origin' | initialization | initialization |
| After power supply off | initialization | no change |

[^2]

1. Right-click the station object "PLC_AC500_V3" in the device tree.
2. Select "Reset origin device [station name]".
$\Rightarrow$ The application is completely erased from the CPU (complete project from all memory areas).

## 4 System overview, planning and operation

### 4.1 System overview

### 4.1.1 AC500 PLC product family

AC500 program- The AC500 (Standard), AC500-eCo, AC500-S and AC500-XC scalable PLC ranges provide mable logic con- solutions for small, medium and high-end applications. Our AC500 platform offers different trollers (PLCs) performance levels and is the ideal choice for high availability, extreme environments or safety solutions. Our AC500 PLC platform offers interoperability and compatibility in hardware and software from compact PLCs up to high-end and safety PLCs.

Due to the flexible combinations of AC500 devices and components, AC500 PLCs can be used for controlling a wide variety of applications to fulfill your automation needs.


## Features of AC500 PLCs

- Scalable and consistently expandable system
- Different performance classes of processor modules (CPUs) available
- Several fieldbusses available
- Parallel connection to several fieldbusses which can be combined arbitrarily

The AC500 product family consists of the product groups:

- AC500 (Standard):

AC500 standard PLCs offer a wide range of performance levels and scalability. The PLCs are highly capable of communication and extension for flexible application.

- AC500-eCo:

AC500-eCo PLCs are cost-effective, high-performance compact PLCs that offer total interoperability with the core AC500 range and provide battery-free buffering of remanent data. All I/O modules can be freely connected in a simple, stable and reliable manner.

- AC500-S:

AC500-S PLCs are designed for safety applications in factory, process or machine automation.

- AC500-XC:

AC500 (Standard) and AC500-S provide devices with -XC extension as a product variant. These variants operate according to their product group and can, in addition, be operated under extreme conditions. AC500-XC PLCs can be used at high altitudes, extended operating temperature and in humid conditions. The devices also provids a high level of resistance to vibration and corrosive gases. The AC500-XC series is consistent with standard devices concerning the overall dimensions, the control function and the software compatibility $\Leftrightarrow$ Chapter 5.1.3 "System data AC500-XC" on page 169.

The AC500 product family is characterized by functional modularity. As the complete AC500 product family shares the same hardware platform and programming software tool, the devices of the AC500 product groups can be flexibly combined.

S500 devices represent the I/O modules of the product group AC500 (Standard), whereas S500-eCo devices represent the I/O modules of the product group AC500-eCo. Both S500 and S500-eCo devices can be flexibly combined with devices of the AC500 product family.

### 4.1.2 AC500/S500 system structure

The AC500 product family provides a variety of modules and pluggable components for expanding the capabilities of the processor module with additional I/Os or other communication protocols. Depending on the features and functions of the processor module (CPU) compatible components can be added to a complete AC500 PLC system.

## Centralized I/O extension



01 Terminal base for processor module and 0 to 6 communication modules, with serial and Ethernet interfaces
02 Communication module for, e.g., PROFIBUS, EtherCAT, and for support of safety solution
03 Processor module with integrated communication
04 I/O module, up to 10 I/O modules can be connected
05 Terminal unit for I/O modules, also for decentralized extension

## Decentralized

I/O extension


1 Terminal unit for communication interface modules
2 Communication interface module with up to 24 integrated I/Os
3 I/O bus
4 Terminal unit for I/O modules, with screw or spring connection
5 Terminal unit for hot swap of I/O modules to exchange of I/O modules while the system is running
6 S500 I/O modules with up to 32 channels for analog and/or digital signals
7 Compact S500-eCo I/O modules with up to 16 channels for analog and/or digital signals
8 Removable terminal blocks

### 4.1.3 AC500-eCo/S500-eCo system structure

AC500-eCo/S500-eCo series is compatible with AC500/S500.
The compact AC500-eCo processor module can be used for small applications with the integrated I/O channels.
The functionality of the processor modules can be extended with option boards, e.g., for additional I/O channels.

For larger applications, I/O modules can be connected centralized or decentralized.


1 AC500-eCo processor module with option boards, different memory sizes, inputs and outputs available
2 S500-eCo I/O module, up to 10 I/O modules can be connected, also for decentralized extension
3 Terminal blocks in different variants available

Centralized I/O extension

Processor modules with an I/O bus interface on the right side can be expanded by up to 10 I/O modules to increase the number of the I/O channels.

---------.

Fig. 8: I/O modules (S500-eCo) directly connected to an AC500-eCo processor module

Up to 10 I/O modules can be connected remotely with the onboard Ethernet interface. Usually, a Modbus/TCP communication interface module or a processor module is used for the decentralized I/O extension.


Fig. 9: I/O modules (S500-eCo) connected to an AC500-eCo processor module via Modbus/TCP network, using a Modbus/TCP communication interface module or a processor module as client.

### 4.1.4 AC500/S500: Short description hardware

## Processor

 modules

AC500 processor modules contain the CPU with the core component of the PLC. The CPU is connected with the user memory, input and output module, communication port and other units via the system bus and performs tasks by means of system programs preset in the system memory. The CPU adopts the function preset by the application program to command the PLC for operation.

It has the following functions:

- To receive user programs and data entered
- To diagnose work faults of the power supply and PLC circuit as well as syntax errors in programming
- To receive the state or data of the site via the input interface and save it into the shadow register or data register
- To read the user programs in the memory one by one and execute them after interpretation
- To updating the state of the associated flag bits and the contents of the shadow register according to the execution results and providing output control using the output unit.

Processor modules are available in different performance classes. Only one processor module is required for a valid system architecture.
There are different types of processor module available that differ in the features and functions they provide, e.g. performance, LED display etc.
If required, processor modules are also available with an integrated Ethernet communication module (TCP/IP).

## Communication modules



AC500 communication modules are required for

- a connection to standard fieldbus systems and
- for integration into existing networks.

AC500 communication modules

- enable communication on different fieldbuses.
- are mounted on the left side of the processor module on the same terminal base.
- are directly powered via the internal communication module bus of the terminal base. A separate voltage source is not required.


## Terminal bases



The terminal base is needed for mounting and connecting the processor module and the communication modules. The modules are plugged on the terminal base.

On AC500-eCo processor modules and special AC500 (Standard) processor modules the terminal base cannot be removed.

## I/O modules



The I/O modules are the input/output unit which connects the PLC with the process. The PLC can detect controlled object data via the input interface and the data is taken as the basis for PLC control on the controlled object. In addition, the PLC sends processing results via the output interface to the controlled object to execute the control.
External input equipment and output equipment needs various signal levels whereas the information processed by the processor module in the PLC can only be the standard level. In order to perform this conversion, the I/O interface generally uses optical isolation and filtering to improve the interference immunity of the PLC. In addition, the I/O interface can generally indicate the working state to facilitate maintenance.

The PLC provides multiple I/O interfaces for operation level and drive capability to users for selection such as digital input, digital output, analog input, analog output, etc. I/O interfaces of the PLC interpret the number of input/output signals as the number of PLC I/O points. The number of I/O points is an important basis for PLC selection. If the system has insufficient I/O points, it can be expanded via the I/O extension interface of the PLC.

The I/O modules for digital and/or analog inputs and outputs are available in different versions and allow flexible use thanks to configurable channels.
The modules can be simply plugged onto a terminal unit for a centralized or decentralized I/O extension via communication interface modules.

## Terminal units



I/O modules and function modules are plugged on a terminal unit.
Terminal units enable simple prewiring without electronics and are available for 24 V DC and 120/230 V AC, optionally for spring or screw-type terminals.

## Communication interface modules

## Function modules

Function modules extend the PLC system to perform special task control. These modules often provide independent components such as a CPU, system programs, memory and interfaces connected with the PLC system bus

Function modules are connected with the PLC via the I/O bus to exchange data and independently work under cooperative management of the PLC

Memory In the PLC, the memory is mainly used for saving system programs, user programs and work data. There are two types of memory:

- Volatile memory

All saved data will be lost after power failure of the memory but the memory can provide a high access rate and unlimited programming cycles. Common volatile memories mainly include SRAM and DRAM (including common memories such as SDRAM).

- Nonvolatile memory:

All saved data will not be lost after power failure of the memory, but the memory is subject to a low read-write rate and limited rewrite cycles. Common nonvolatile memories mainly include NORflash, NANDflash, EEPROM, memory card, etc.

AC500 PLCs store all user programs in the nonvolatile memory to protect them from power failure. The programs are exported to the volatile memory during operation of the PLC to ensure high-speed and efficient operation. If user program debugging is finished, the programs can be fixed in the nonvolatile memory when they need no change. The work data is subject to frequent change and access during the PLC operation. It is saved in the volatile memory to meet the requirements for random access.

The work data memory of the PLC has the memory area for input and output relay, auxiliary relay, timer, counter and other logic devices. The state of these devices depends on initial setting and operation of the user programs. Some data maintains its existing state by using built-in supercapacitors or backup batteries in the event of a power failure. The memory area for data saving in the event of a power failure is called the data retention area.

Power supply The PLC is equipped with a switching power supply for the internal supply. In comparison with an ordinary power supply, the switching power supply has a higher stability and a higher noise immunity.

Some modules include a stabilized power supply for the supply of external sensors.

### 4.1.5 AC500-eCo/S500-eCo: Short description hardware

## Processor

 modules

AC500-eCo processor modules contains the CPU with the core microprocessor of the PLC. It is integrated with power supply, onboard I/Os and communication interface.
Functions:

- To download user programs
- To run the CPU
- To execute user programs in loops
- To monitor program input and output devices.

Processor modules are available in different performance classes and provide different numbers of onboard I/Os.

Only one processor module is required for a valid system architecture.

## Option boards



AC500-eCo processor modules can be extended with option boards. Option boards provide specific functionality. According to the processor module type, up to three option board slots are available for extension. Each option board slot supports most existing types of option board module. The option board modules provide the following functionality for processor module extension:

- Serial interface RS232 (isolated) or RS485 (isolated or not isolated)
- Digital I/O channels extension (digital inputs, digital outputs, or mixed type)
- Analog I/O channels extension (analog inputs, analog outputs) for standard signal voltage or current, but also temperature measurement with RTD/NTC or thermocouple sensors
- RTC real-time clock board for processor module PM5012 (other processor modules include RTC functionality)
- Configuration address switch board for KNX protocol, only for processor modules PM5072 and PM5082

If the number of onboard I/Os provided on the processor module is insufficient for a certain use case, the PLC can be expanded with I/O modules to meet the control requirements.

Memory In the PLC, the memory is mainly used for saving system programs, user programs and work data. There are two types of memory:

- Volatile memory:

All saved data will be lost after power failure of the memory but the memory can provide a high access rate and unlimited programming cycles. Common volatile memories mainly include SRAM and DRAM (including common memories such as SDRAM).

- Nonvolatile memory:

All saved data will not be lost after power failure of the memory, but the memory is subject to a low read-write rate and limited rewrite cycles. Common nonvolatile memories mainly include NORflash, NANDflash, EEPROM, memory card, etc.

AC500 PLCs store all user programs in the nonvolatile memory to protect them from power failure. The programs are exported to the volatile memory during PLC operation to ensure high-speed and efficient operation. If user program debugging is finished, the programs can be fixed in the nonvolatile memory when they need no change. The work data is subject to frequent change and access during PLC operation. It is saved in the volatile memory to meet the requirements for random access.

The work data memory of the PLC has the memory area for input and output relay, auxiliary relay, timer, counter and other logic devices. The state of these devices depends on initial setting and operation of the user programs. Some data maintains its existing state by using built-in supercapacitors or backup batteries in the event of a power failure. The memory area for data saving in the event of a power failure is called the data retention area.

Power supply The PLC is equipped with a switching power supply for the internal supply. In comparison with an ordinary power supply, the switching power supply has a higher stability and a higher noise immunity.

Some modules include a stabilized power supply for the supply of external sensors.

### 4.1.6 AC500-S

Functional safety

The AC500-S safety user manual must be read and understood before using the safety configuration and programming tools of Automation Builder/PS501 Control Builder Plus. Only qualified personnel are permitted to work with AC500-S safety PLCs.

The AC500-S safety PLC includes the following safety-relevant hardware components.

- SM560-S/SM560-S-FD-1/SM560-S-FD-4
- DI581-S
- DX581-S
- Al581-S
- TU582-S


### 4.1.7 CP600 control panels



CP600 control The CP600-eCo, CP600 and CP600-Pro control panels offer a wide range of features and panels platform functionalities for tailor-made visualization. Our CP600 platform offers different performance levels including control panels for economic applications, robust standard operator panels, high end multi-touch human machine interfaces (HMIs).

The easy to use engineering tool PB610 Panel Builder 600, part of Automation Builder, ensures easy scalability on the CP600 platform.

- CP600-eCo: The economical CP600-eCo control panel, with screen sizes from 4.3" to $10.1^{\prime \prime}$ widescreen, is aimed for standard functions and high usability for clear interaction with the operation process.
- CP600: The robust CP600 HMI, with screen sizes from 7" to 15 " provides up-to-date visualization performance, versatile communication and representative design for machines and systems.
- CP600-Pro: The CP600-Pro HMI, with screen sizes from 5" to 21.5 " widescreen, comes with high end visualization performance, multi-touch operation and versatile communication options
\& Further information on page 6


### 4.1.8 Automation Builder: Short description engineering software

Details about the Automation Builder license model, the features of the editions and the latest license information

## Configuration and programming

Configuration and programming of all AC500 control systems (processor modules) is done by using Automation Builder software.
Features:

- Standardized programming according to IEC 61131-3, five programming languages (Structured Text (ST), Function Block Diagram (FBD), Instruction List (IL), Ladder Diagram (LD), Sequential Function Chart (SFC)), Continuous Function Chart (CFC), debugging functions for program test
- Online diagnosis
- Debugging functions for the program test: Single step, single cycle, breakpoint
- Offline simulation - simulate commands without PLC being connected
- Sampling trace - timing diagrams for process variables
- Recipe management and watch lists
- Visualization
- Configuration of the communication interface modules (for PROFINET, PROFIBUS, EtherCAT, CANopen, Ethernet, Modbus)
- Programming - serial or via Ethernet networks
- Comprehensive libraries
- Export and import interfaces for devices, signals, applications, visualization, etc.
- Multi-user support and project compare
- Project scripting


Offline simula- IEC 61131-3 commands can be simulated without a PLC being connected, including the reletion vant malfunctions. After the program test, the application can be downloaded to the control system.

Sampling trace Timing diagrams for process variables and storage of data in a circular buffer with event trigger.

Recipe manage- Values of selected variables are displayed. Pre-defined values can be assigned to variables ment and watch which can then be downloaded to the control system all at once ("Write recipe"). Actual values lists from the control system can also be pre-assigned for reading into the Watch and Recipe Manager, and stored in memory there ("Read recipe"). These functions are also helpful, for example, for setting and entering control parameters.

Visualization Includes color change, moving elements, bitmaps, text display, allows input of setpoint values and display of process variables read from the PLC, dynamic bar diagrams, alarm and event management, function keys and ActiveX elements.

Programming The Ethernet interface of the processor modules is used to connect to the engineering software for programming, debugging and diagnosis.

[^3]
### 4.2 Application planning

### 4.2.1 Safety instructions

## NOTICE!

Wiring diagrams and user programs must be designed to prevent hazardous situations from occurring during normal operation or in the event of a malfunction.

### 4.2.2 Processor module and I/O selection

How to select modules for your automation task: In our main catalog we offer a CPU selector and several comparative tables to find the modules for your application.

### 4.2.3 I/O bus - Communication within the PLC

The synchronized I/O bus is the I/O data bus for the I/O modules connected with the processor modules or communication interface modules. Through this bus, I/O and diagnosis data are transferred.

With its fast data transmission, the I/O bus obtains very low reaction times.
Up to 10 I/O terminal units (for one I/O module each) can be added to one terminal base or to one AC500-eCo processor module. The I/O terminal units and the AC500-eCo I/O modules, have a bus input at the left side and a bus output at the right side. Thus the length of the I/O bus increases with the number of attached I/O modules ${ }^{\circ} \stackrel{\text { Table }}{ } 119$ "Maximum number of I/O devices which can be connected to the I/O bus" on page 389.


1 I/O bus connection
The connection of the I/O bus is performed automatically by telescoping the modules on the DIN rail. The I/O bus provides the following signals:

- Supply voltage of 3.3 V DC for feeding the electronic interface components
- 3 data lines for the synchronized serial data exchange
- several control signals


## NOTICE!

Except when using hot swap terminal units, the I/O bus is not designed for pulling and plugging modules during operation. If a module is pulled or plugged on a terminal unit that is not hot swap capable while the bus is running, the following consequences are possible

- reset of the station or of the processor module
- system lockup
- damage of the module


## WARNING!

## Removal/Insertion under power

Removal or insertion under power is permissible only if all conditions for hot swapping are fullfilled.

## « Chapter 4.9.3 "Replace an I/O module with hot swap" on page 153

The devices are not designed for removal or insertion under power when the conditions for hot swap do not apply. Because of unforeseeable consequences, it is not allowed to plug in or unplug devices with the power being ON.
Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while they are energized in a hazardous location could result in an electric arc, which could create an ignition source resulting in fire or explosion.
Prior to proceeding, make sure that power is been disconnected and that the area has been thoroughly checked to ensure that flammable materials are not present.

The devices must not be opened when in operation. The same applies to the network interfaces.

Profibus (master and slave) and CM589-PNIO are available since version 2.5.0 of the Automation Builder.

### 4.2.4 Fieldbus connectivity options

The PLC and control panel portfolio offers a wide range of scalable products, communication protocols and connectivity options, from the field level to the management and visualization levels.

An overview of the different connectivity options can be found below.


Fig. 10: Exemplary networking

## IT network/internet

- FTP(S)
${ }^{\text {r }}$ Chapter 6.3.4.5 "FTP server" on page 1721
- HTTP(S)

② Chapter 6.5.5 "Application libraries" on page 3833

- MQTT
\& Chapter 6.3.4.6 "MQTT client protocol" on page 1722
- NTP/SNTP
\& Chapter 6.3.4.4.1 "Introduction of the NTP/SNTP protocol" on page 1716
- OPC UA
\& Chapter 6.3.4.11 "OPC UA" on page 1750
- SMTP(S)

² Chapter 6.6.10 "SNMP" on page 4362

## Factory/side network

- BACnet

② Chapter 6.3.4.9 "BACnet-BC" on page 1733

- DNP3 (in preparation)
- FTP(S)
̌ Chapter 6.3.4.5 "FTP server" on page 1721
- HTTP(S)

② Chapter 6.5.5 "Application libraries" on page 3833

- IEC 60870-5-104

Chapter 6.3.4.1 "IEC60870-5-104 (Telecontrol)" on page 1647

- IEC 61850

² Chapter 6.3.4.2 "IEC 61850 Server" on page 1671

- KNX
„ Chapter 6.3.4.8 "KNX configurator" on page 1730
- MySQL/MSSQL ⓨ Chapter 6.5.5 "Application libraries" on page 3833
- OPC DA/AE
\& Chapter 6.3.4.11 "OPC UA" on page 1750
- OPC UA
\& Chapter 6.3.4.11 "OPC UA" on page 1750
- SNMP

ヶ Chapter 6.6.10 "SNMP" on page 4362

- SNTP
\& Chapter 6.3.4.4.1 "Introduction of the NTP/SNTP protocol" on page 1716
- TCP/IP
« Chapter 6.3.2.2.4.4 "Configuration of communication via Ethernet (TCP/IP)" on page 1469
- UDP

² Chapter 6.3.4.10 "UDP protocol" on page 1750

## Control network

- CANopen

๕ Chapter 6.3.2.11.1 "CANopen" on page 1521

- CAN 2A/2B

Chapter 6.3.2.11.1.1.5 "Configuration of the protocols CAN 2.0 A / CAN 2.0 B"
on page 1533

- EtherCAT
y Chapter 6.3.2.17 "EtherCAT configurator " on page 1609
- Ethernet/IP
«y Chapter 6.3.2.2.4.4 "Configuration of communication via Ethernet (TCP/IP)" on page 1469
- IEC 60870-5-104

Chapter 6.3.4.1 "IEC60870-5-104 (Telecontrol)" on page 1647

- IEC 61850

Chapter 6.3.4.2 "IEC 61850 Server" on page 1671

- Modbus RTU

Chapter 6.8.2.9 "Communication with Modbus RTU" on page 4506

- Modbus TCP

② Chapter 6.8.2.10 "Communication with Modbus TCP/IP" on page 4523

- PROFIBUS DP
\& Chapter 6.3.2.11.4 "PROFIBUS" on page 1553
- PROFINET
${ }^{\star}$ Chapter 6.3.2.11.2 "PROFINET" on page 1534
- PROFIsafe AC500-S
- SAE J1939

๕ Chapter 6.3.2.16.2.3.2 "Tab 'J1939 Manager - General"' on page 1604

### 4.2.5 Power supply dimensioning

### 4.2.5.1 General

The power consumption of a complete station consists of the sum of all individual consumptions.

The two supply voltages with 24 V DC are distinguished in the AC500 platform:

- Supply of the internal logic via terminals L+ and M on the CPU module, or an the AC500 terminal base for: CPU, communication mudule(s) and I/O bus.
- Supply of the process-side input/output circuits for analog signals and 24 V DC digital signals via the ZP and UP terminals of the S500 terminal units.
The two supply voltages can be provided by the same power supply unit. The CPU and the I/O modules should, however, be fused separately. Of course also separate power supplies are possible.


### 4.2.5.2 Calculation of the total current consumption

Example In the example, the AC500 control system consists of the following devices:

- AC500 CPU with Ethernet interface
- 4 communication modules
- 7 I/O modules (digital and analog)
- As well as the required terminal bases and terminal units


Because of the high total current consumption of the digital I/O modules (from $U P=24 V D C$ ), the supply is divided up into several electric circuits fused separately.
The maximum permitted total current over the supply terminals of the I/O terminal units is 8 A .

The total current can be calculated as follows:
$I_{\text {Total }}=I_{\text {L+ }}+I_{\text {UP }}$
with the assumptions
$I_{L+}=I_{C P U}+I_{I / O \text { bus }}+I_{C 1}+I_{C 2}+I_{C 3}+I_{C 4}$ (CPU + communication modules $+I / O$ bus $)$
$I_{I / O \text { bus }}=$ Number of expansion modules $\times$ Current consumption through the I/O bus per module
and
 $+I_{\text {LOAD7 }}$

If one assumes that all outputs are switched on and are operated with their maximum permitted load currents (under compliance with the maximum permitted currents at the supply terminals), then the following values are the result for an example shown above:

|  | $\mathrm{I}_{\text {CPU }}{ }^{*}$ ) | $\mathrm{I}_{\mathrm{Cx}}{ }^{*}$ ) | $\mathrm{I}_{\text {/O bus }}{ }^{*}$ ) | $\mathrm{I}_{\text {UPx }}{ }^{*}$ ) | $\mathrm{l}_{\text {LOADx }}{ }^{*}$ ) |
| :---: | :---: | :---: | :---: | :---: | :---: |
| CPU/communication module part |  |  |  |  |  |
| CPU | 0.110 A | - | - | - | - |
| C1 | - | 0.050 A | - | - | - |
| C2 | - | 0.085 A | - | - | - |
| C3 | - | 0.050 A | - | - | - |
| C4 | - | 0.050 A | - | - | - |
| I/O module part |  |  |  |  |  |
| Analog1 | - | - | 0.002 A | 0.150 A | - |
| Analog2 | - | - | 0.002 A | 0.150 A | 0.160 A |
| Analog3 | - | - | 0.002 A | 0.100 A | 0.080 A |
| Analog4 | - | - | 0.002 A | 0.100 A | 0.080 A |
| Digital1 | - | - | 0.002 A | 0.050 A | 8.000 A |
| Digital2 | - | - | 0.002 A | 0.050 A | 8.000 A |
| Digital3 | - | - | 0.002 A | 0.050 A | 8.000 A |
| $\Sigma$ columns | 0.110 A | 0.235 A | 0.014 A | 0.650 A | 24.320 A |
|  | $\Sigma \mathrm{I}_{\mathrm{L}+} \approx 0.4 \mathrm{~A}$ |  |  | $\Sigma \mathrm{I}_{\mathrm{UP}} \approx 25 \mathrm{~A}$ |  |
|  | $\mathrm{l}_{\text {Total }} \approx 25.4 \mathrm{~A}$ |  |  |  |  |
| *) All values in this column are exemplary values |  |  |  |  |  |

### 4.2.5.3 Dimensioning of the fuses

To be able to select the fuses for the station correctly, both the current consumption and the inrush currents (melting integral for the series-connected fuse) must be taken into consideration.

| Fuse | for | $\sum$ of the <br> melting <br> integrals in <br> $\mathbf{A}^{2} \mathbf{s}$ | $I_{\text {L+ }}$ | $\mathbf{I}_{\text {UPx }}$ | Recommended fuse |  |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| F1 | CPU logic | 1.000 | $\approx 0.4$ | - | Type | Value |
| F2 | Module Dig- <br> ital1 | 0.005 | - | 8.050 | Quick | 10 A |
| F3 | Module Dig- <br> ital2 | 0.008 | - | 8.050 | Quick | 10 A |
| F4 | Module Dig- <br> ital3 | 0.007 | - | 8.050 | Quick | 10 A |
| F5 | Modules <br> Analog1 + <br> Analog2 + <br> Analog3 + <br> Analog4 | 0.130 | - | 0.820 | Quick | 10 A |

### 4.2.6 Libraries, software packages and licensed features

Many applications are included in the Automation Builder. However, there are some that need to be purchased additionally. The applications are briefly introduced here. If applicable, links lead to further information depending on the version.

## HA-Modbus TCP library

Library package adds High Availability System functionality for redundant hot standby over Ethernet field network via HA-Modbus TCP.
«2 Chapter 6.5.9.1 "HA-Modbus TCP - System technology" on page 3859

## 61850 protocol

Adds engineering tool and library for 61850 Ed. 1 MMS server and GOOSE publish and subscribe functionalities.
*) Chapter 6.3.4.2 "IEC 61850 Server" on page 1671

## KNX protocol

Engineering and protocol package that seamlessly integrates ETS and Automation Builder.

* Chapter 6.3.4.8 "KNX configurator" on page 1730


## BACnet-BC protocol

BACnet is a standardized data communication protocol for building automation and control networks as defined in the ANSI/ASHRAE standard 135 and ISO 16484-5.

② Chapter 6.3.4.9 "BACnet-BC" on page 1733

## Data logger library

The data logging function block library ("DataLogger" and "DataLoggerEco") contains 5 function blocks for the purpose of advanced time-stamped data logging for different use cases.
y Chapter 6.5.8 "Data Logger Library" on page 3849

## Motion control library

Library package for decentralized, centralized and coordinated motion according to the PLCopen standard.
« Chapter 6.5.10.3 "Motion control library" on page 3958

## EtherNet/IP

EtherNet/IP is used to connect any number of different terminals in a network, using a scanneradapter relationship.
*) Chapter 6.4.2.5.1 "Introduction" on page 2816

Table 6: AC500 runtime licenses

| For | Description | Type | Order code |
| :--- | :--- | :--- | :--- |
| all AC500 CPUs | HA-Modbus TCP run- <br> time license | PS5601-HA-MTCP | 1SAP195400R0101 |
| all AC500 V3 CPUs <br> and AC500-eCo V3 <br> PM5072 and PM5082 | IEC 61850 protocol <br> runtime license | PS5602-61850 | 1SAP195600R0101 |
| all AC500 V3 CPUs <br> and AC500-eCo V3 <br> PM5072 and PM5082 | KNX IP protocol run- <br> time license | PS5604-KNX | 1SAP195800R0101 |
| all AC500 V3 CPUs <br> and AC500-eCo V3 <br> PM5072 and PM5082 | BACnet protocol B-BC <br> runtime license | PS5607-BACnet-BC | 1SAP195550R0101 |


| For | Description | Type | Order code |
| :---: | :---: | :---: | :---: |
| all AC500 V3 CPUs | Data logger runtime license for AC500 V3 | PS5609-Log | 1SAP195002R0101 |
| $\begin{array}{\|l\|} \hline \text { AC500-eCo V3 } \\ \text { PM5072 and PM } 5082 \end{array}$ | Data logger runtime license for PM5072 and PM5082 | PS5609-Log-e | 1SAP195004R0101 |
| all AC500 V3 CPUs and AC500-eCo V3 PM5032 and higher | Motion control library runtime license | PS5611-MC | 1SAP192150R0101 |
| AC500 V3/AC500- <br> eCo V3 <br> (for PM5032, PM5052, PM5072, PM5082 and PM56xx) | EtherNet/IP scanner runtime license for AC500 V3 | PS5613-EIP-S | 1SAP196101R0101 |
| AC500-eCo V3 <br> (for PM5032, PM5052, PM5072 and PM5082) | EtherNet/IP scanner runtime license for AC500-eCo V3 V3 | PS5613-EIP-S-e | 1SAP196103R0101 |
| AC500 V3/AC500- eCo V3 <br> (for PM5032, PM5052, PM5072, PM5082 and PM56xx) | EtherNet/IP adapter runtime license for AC500 V3 | PS5613-EIP-A | 1SAP196100R0101 |
| AC500-eCo V3 <br> (for PM5032, PM5052, PM5072 and PM5082) | EtherNet/IP adapter runtime license for AC500-eCo V3 | PS5613-EIP-A-e | 1SAP196102R0101 |
| For using runtime licensed features one license per CPU is required. The license has to be installed on the AC500 V3 CPU either by connecting it to Automation Builder or via memory card that has been prepared by Automation Builder for license activation. |  |  |  |

Activating a run- The linked application note provides step-by-step instructions on how to activate the runtime time license license for the PLC.

## AC500 V3 Activating a runtime license

Additional libraries and software packages for specific use cases are available to reduce the programming effort and implement quickly your application. In our main catalog we offer an overview of the available libraries and software packages.
\# Chapter 6.5 "Libraries and solutions" on page 3773

### 4.3 Mechanical planning and installation

### 4.3.1 Control cabinet assembly (indoor use)

## NOTICE!

PLC damage due to incorrect housing
Due to their construction (degree of protection IP 20 according to EN 60529) and their connection technology, the devices are only suitable for operation in closed control cabinets.

The cabinet must be suitable to protect the equipment from the following:

- unauthorized access,
- dusting and contamination,
- humidity and moisture and
- mechanical damage,

The equipment must be operated within the specified mechanical and enviromental conditions.

## Maintain spacing from:

- enclosure walls
- wireways
- adjacent equipment

Allow a minimum of 20 mm clearance on all sides. This provides ventilation and galvanic isolation.
It is recommended to mount the modules on an grounded mounting plate, or an grounded DIN rail, independent of the mounting location.


Fig. 11: Exemplary figure: left side AC500, right side AC500-eCO
1 Cable duct
2 Distance from cable duct $\geq 20 \mathrm{~mm}$
3 Mounting plate, grounded
4 Screw end-stop clamps recommended to avoid movement of the modules on the DIN rail

## NOTICE!

Horizontal mounting is highly recommended.
Vertical mounting is possible, then derating must be considered to avoid overheating due to poor air circulation. $\Leftrightarrow>$ Chapter 5.1.2.1 "Environmental conditions" on page 166.

When horizontal mounted, end-stop clamps are recommended to secure the modules in case of shock or vibration.

When vertically mounted, always place an end-stop clamps on the bottom and on the top of the modules to properly secure the modules.

### 4.3.2 Mounting and demounting - general information

The control system is designed to be mounted to a well-grounded mounting surface such as a metal panel. Additional grounding connections from the mounting tabs or DIN rail (if used), are not required unless the mounting surface cannot be grounded.

During panel or DIN rail mounting of all devices, be sure that all debris (metal chips, wire strands, etc.) is kept from falling into the controller. Debris that falls into the controller could cause damage while the controller is energized.

All devices are grounded through the DIN rail to chassis ground. Use zinc plated yellow-chromate steel DIN rail to assure proper grounding. The use of other DIN rail materials (e.g. aluminium, plastic, etc.) that can corrode, oxidize, or are poor conductors, can result in improper or intermittent grounding.

## Consider the safety instructions

In the description, special attention must be paid to designs using galvanic isolation, grounding and EMC measures for the reasons stated. Consider the safety instructions for AC500 product family.

### 4.3.3 Mounting and demounting the terminal base

Mounting and demounting on
DIN rail

1. Mount DIN rail 7.5 mm or 15 mm .
2. Mount the terminal base/function module terminal base:

$\Rightarrow$ The terminal base is put on the DIN rail above and then snapped-in below.
3. The demounting is carried out in a reversed order.


Mounting with screws

If the terminal base should be mounted with screws, wall mounting accessories TA526 must be inserted at the rear side first $\Leftrightarrow$ Chapter 5.2.8.2.5 "TA526 - Wall mounting accessory" on page 1324. These plastic parts prevent bending of the terminal base while screwing on. TB560x and TB561x need one TA526, TB562x, TB564x and TB566x need two TA526.


Fig. 13: Function module terminal bases, Fastening with screws

By wall mounting, the terminal base is grounded through the screws. It is necessary that

- the screws have a conductive surface (e.g. steel zinc-plated or brass nickelplated)
- the mounting plate is grounded
- the screws have a good electrical contact to the mounting plate

Practical tip The following procedure allows you to use the mounted modules as a template for drilling holes in the panel. Due to module mounting hole tolerance, it is important to follow these procedures:

1. On a clean work surface, mount no more than 3 modules (e.g. one terminal base and two terminal units).
2. Using the mounted modules as a template, carefully mark the center of all modulemounting holes on the panel.
3. Return the mounted modules to the clean work surface, including any previously mounted modules.
4. Drill and tap the mounting holes for the screws (M4 or \#8 recommended).
5. Place the modules back on the panel and check for proper hole alignment.
6. Attach the modules to the panel using the mounting screws.

7. Repeat the steps for all remaining modules.

### 4.3.4 Mounting and demounting the AC500 processor module

1. After mounting the terminal base on the DIN rail, mount the processor module.

2. Press the processor module into the terminal base until it locks in place.
3. The demounting is carried out in a reversed order. Press above and below, then remove the processor module.


### 4.3.5 Mounting and demounting the communication module

Communication modules are mounted on the left side of the processor module on the same terminal base. The connection is established automatically when mounting the communication module.

## NOTICE!

## Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

After mounting the terminal base, mount the communication modules.

1. First insert the bottom nose of the communication module into the dedicated holes of the terminal base. Then, rotate the communication module on the dedicated terminal base slot until it is locked in place.

$\Rightarrow$

## NOTICE!

## Risk of malfunctions!

Unused slots for communication modules are not protected against accidental physical contact.

- Unused slots for communication modules must be covered with dummy communication modules to achieve IP20 rating \& Chapter 5.2.8.2.4 "TA524 - Dummy communication module" on page 1323.
- I/O bus connectors must not be touched during operation.

2. The demounting is carried out in a reversed order.

Press above and below, then rotate the communication module and remove it.


### 4.3.6 Mounting and demounting the terminal unit <br> Mounting on <br> DIN rail

1. Mount DIN rail 7.5 mm or 15 mm .
2. Mount the terminal unit.

The terminal unit is snapped into the DIN rail in the same way as the Terminal Base. Once secured to the DIN rail, slide the terminal unit to the left until it fully locks into place creating a solid mechanical and connection.

When attaching the devices, make sure the bus connectors are securely locked together to ensure proper connection. Max. 10 terminal units can be attached.

3. Demounting: A screwdriver is inserted in the indicated place to separate the terminal units.


Mounting with screws

If the terminal unit should be mounted with screws, wall mounting accessories TA526 must be inserted at the rear side first ${ }^{*} \Rightarrow$ Chapter 5.2.8.2.5 "TA526 - Wall mounting accessory" on page 1324. These plastic parts prevent bending of the Terminal Base while screwing on.


Fig. 14: Fastening with screws

By wall mounting, the terminal unit is grounded through the screws. It is necessary that

- the screws have a conductive surface (e.g. steel zinc-plated or brass nickelplated)
- the mounting plate is grounded
- the screws have a good electrical contact to the mounting plate

Practical tip The following procedure allows you to use the mounted modules as a template for drilling holes in the panel. Due to module mounting hole tolerance, it is important to follow these procedures:

1. On a clean work surface, mount no more than 3 modules (e.g. one terminal base and two terminal units).
2. Using the mounted modules as a template, carefully mark the center of all modulemounting holes on the panel.
3. Return the mounted modules to the clean work surface, including any previously mounted modules.
4. Drill and tap the mounting holes for the screws (M4 or \#8 recommended).
5. Place the modules back on the panel and check for proper hole alignment.
6. Attach the modules to the panel using the mounting screws.

$$
\begin{aligned}
& \text { If mounting more modules, mount only the last one of this group and put } \\
& \text { the others aside. This reduces remounting time during drilling and tapping } \\
& \text { of the next group. }
\end{aligned}
$$

7. Repeat the steps for all remaining modules.

### 4.3.7 Mounting and demounting the I/O module

After mounting the terminal unit, mount the I/O modules.

1. Press the I/O module into the terminal unit until it locks in place.

2. The demounting is carried out in a reversed order. Press above and below, then remove the module.


### 4.3.8 Mounting and demounting the AC500-eCo processor module

4.3.8.1 Mounting and demounting the option boards

### 4.3.8.1.1 Optimized mounting of the option boards

The AC500-eCo processor modules have up to 3 slots for option boards.

Table 7: Option board slots


The best thermal circulation is given on slot 3 (bottom slot), followed by slot 2 (middle slot) and then slot 1 (top slot).
The best mounting position of the option board depends on its power dissipation.

## Rules for optimized mounting of the option boards

- The higher the power dissipation of the option board, the lower the mounting position should be selected.
- The TA5126-2AO-UI option board has the highest power dissipation and must always be mounted at the lowest option board slot.
The optimized mounting position can be easily determined with the help of this table.

Table 8: Power dissipation of the option boards

| Power dissipation of <br> the option boards | Digital | Analog | Serial interface | Accessory |
| :---: | :---: | :---: | :---: | :---: |
| very small |  |  | TA5141-RS232I(W) <br> TA5142-RS485I(W) <br> TA5142-RS485(W) | TA5130-KNXPB(W) <br> TA5131-RTC |
| small | TA5105-4DOT(W) | TA5123-2AI-RT(D/W) |  |  |
| medium | TA5110-2DI2DO(T/W) <br> TA5101-4DI(W) | TA5120-2AI-UI(W) |  |  |
| large |  | TA5126-2AO-UI(W) |  |  |

If the option boards to be mounted are in the same power dissipation level, then the slots can be freely selected.

### 4.3.8.1.2 Inserting the option board


$\triangleright$ Press the option board TA51xx (or TA5300-CVR) into the slot of the processor module PM50x2 until it locks in place.

The option board must click into the slot of the processor module.

### 4.3.8.1.3 Removing the option board



1. Push the option board on the side to release the lock.
2. At the same time, pull the option board out of the slot.

## CAUTION!

Risk of injury and damaging the product!
Always plug in the option board slot cover when the option board is not inserted.
If the option board slot cover is lost, please order the replacement TA5300-CVR (1SAP187500R0001).

Never power up the CPU with uncovered option board slot, otherwise it may cause serious injury and/or damage the product.

### 4.3.8.2 Mounting and demounting of the terminal blocks

Intended pur- Removable terminal blocks are used for power supply and for I/O connectors on AC500-eCo V3 pose processor modules PM50x2.

For option boards there are different removable terminal blocks in spring version.

For the AC500-eCo V3 Basic CPUs a 3-pin terminal block for power supply and a 13-pin terminal block for I/O connectors are used.

For the AC500-eCo V3 Standard CPUs and Pro CPUs a 3-pin terminal block for power supply, a 13-pin terminal block and a 12-pin terminal block for l/O connectors are used.

For all CPUs there is a screw and a spring variant available.

| Basic CPU |  | Standard and Pro CPUs |  |
| :---: | :---: | :---: | :---: |
| Spring terminal TA5211-TSPF-B | Screw terminal TA5211-TSCL-B | Spring terminal TA5212-TSPF | Screw terminal TA5212-TSCL |
|  |  |  |  |

Various removable spring terminal blocks are available for option boards.
The following spare parts are available (depending on the number of pins).

| Spring terminals |  |  |  |
| :---: | :---: | :---: | :---: |
| TA5220-SPF5 | TA5220-SPF6 | TA5220-SPF7 | TA5220-SPF8 |
|  |  |  |  |

## CAUTION!

## Risk of injury and damaging the product!

Improper installation and maintenance may result in injury and can damage the product!

- Installation and maintenance have to be performed according to the technical rules, codes and relevant standards, e.g. EN 60204-1.
- Read product documentation carefully before wiring. Improper wiring or wrong terminal block from other devices can damage the product!
- Only by qualified personnel.


## CAUTION!

Risk of injury and damaging the module when using unapproved terminal blocks!
Only use terminal blocks approved by ABB to avoid injury and damage to the module.

## Terminal block set for PM50x2

Processor modules PM50x2 CPU are not delivered with terminal blocks.
Screw terminal block set:

- TA5211-TSCL-B (1SAP187400R0001) for PM5012-x-ETH
- TA5212-TSCL (1SAP187400R0004) for PM5032-x-ETH, PM5052-x-ETH, PM5072-T-2ETH(W), PM5082-T-2ETH
Spring terminal block set:
- TA5211-TSPF-B (1SAP187400R0002) for PM5012-x-ETH
- TA5212-TSPF (1SAP187400R0005) for PM5032-x-ETH, PM5052-x-ETH, PM5072-T-2ETH(W), PM5082-T-2ETH


## Assembly



## Disassembly



### 4.3.8.3 Mounting a processor module on a DIN rail

## NOTICE!

Risk of function faults!
The processor module is grounded via DIN rail.
The DIN rail must be included into the grounding conception of the plant. Use only metal screws for grounding.


Mount the processor module at the top of the DIN rail, then snap it in below.

System overview, planning and operation
Mechanical planning and installation > Mounting and demounting the AC500-eCo processor module

See hardware description of PM50xx for connection.

### 4.3.8.4 Demounting a processor module mounted on a DIN rail

1. Remove I/O modules if connected.

2. While pressing down processor module pull it away from DIN rail.


### 4.3.8.5 Mounting a processor module on a metal plate

## NOTICE!

Risk of function faults!
Missing electrical contact by isolating screws or washers!
Use metal screws on the metal plate.
The metal plate must be included into the grounding concept of the plant.
Do NOT use insulating washers!

One TA543 wall mounting accessory is needed per processor module.

1. Snap in the TA543 at the back side of the processor module.

2. Fasten the processor module with two screws (diameter: 4 mm ) to the metal plate.


System overview, planning and operation
Mechanical planning and installation > Mounting and demounting the AC500-eCo processor module

③ See hardware description of PM50xx for connection.

### 4.3.8.6 Demounting a processor module mounted on metal plate

1. Remove I/O modules if connected.

2. Remove the 2 screws.


### 4.3.8.7 Mounting of the cable fixing accessory TA5301-CFA



1 TA5301-CFA cable fixing accessory
22 openings on the PM50x2 processor module
$\triangleright \quad$ Insert the TA5301-CFA cable fixing accessory into the two openings on the PM50x2 processor module marked white in the figure.

### 4.3.9 Mounting and demounting the S500-eCo I/O module

S500-eCo I/O modules can be mounted either on a DIN rail or with screws on a metal plate.

Mounting I/O modules on a DIN rail

## NOTICE!

## Risk of function faults!

The S500-eCo I/O modules are grounded via the DIN rail.
The DIN rail must be included into the grounding concept of the plant.
Use only metal screws.

1. Mount I/O module at the top of the DIN rail, then snap it in below.

2. Attach I/O module by hand to an other module. The I/O bus is connected automatically.


Demounting I/O
modules
mounted on a
DIN rail

1. Remove I/O module by hand if connected.

2. While pressing down I/O module pull it away from DIN rail.


Demounting I/O
modules
mounted on a metal plate

1. Remove the 2 screws.

2. Remove the I/O module from the connected module by hand.


### 4.3.10 Accessories for AC500 (Standard)

### 4.3.10.1 Inserting and removing a memory card



[^4]1. Insert the memory card into the memory card slot of the processor module until locked.

2. To remove the memory card, push on the memory card until it moves forward. By this, the memory card is unlocked and can be removed.

### 4.3.10.2 Further accessories

This section only describes accessories that are frequently used for system assembly, connection and construction.
All additional accessories that can be used to supplement the AC500 system

### 4.3.11 Accessories for AC500-eCo

### 4.3.11.1 Mounting and demounting the option boards

### 4.3.11.1.1 Optimized mounting of the option boards

The AC500-eCo processor modules have up to 3 slots for option boards.

Table 9: Option board slots


The best thermal circulation is given on slot 3 (bottom slot), followed by slot 2 (middle slot) and then slot 1 (top slot).
The best mounting position of the option board depends on its power dissipation.

## Rules for optimized mounting of the option boards

- The higher the power dissipation of the option board, the lower the mounting position should be selected.
- The TA5126-2AO-UI option board has the highest power dissipation and must always be mounted at the lowest option board slot.
The optimized mounting position can be easily determined with the help of this table.

Table 10: Power dissipation of the option boards

| Power dissipation of <br> the option boards | Digital | Analog | Serial interface | Accessory |
| :---: | :---: | :---: | :---: | :---: |
| very small |  |  | TA5141-RS232I(W) <br> TA5142-RS485I(W) <br> TA5142-RS485(W) | TA5130-KNXPB(W) <br> TA5131-RTC |
| small | TA5105-4DOT(W) | TA5123-2AI-RT(D/W) |  |  |
| medium | TA5110-2DI2DO(T/W) <br> TA5101-4DI(W) | TA5120-2AI-UI(W) |  |  |
| large |  | TA5126-2AO-UI(W) |  |  |

If the option boards to be mounted are in the same power dissipation level, then the slots can be freely selected.

### 4.3.11.1.2 Inserting the option board


$\triangleright$ Press the option board TA51xx (or TA5300-CVR) into the slot of the processor module PM50x2 until it locks in place.

The option board must click into the slot of the processor module.

### 4.3.11.1.3

Removing the option board


1. Push the option board on the side to release the lock.
2. At the same time, pull the option board out of the slot.

## CAUTION!

## Risk of injury and damaging the product!

Always plug in the option board slot cover when the option board is not inserted.
If the option board slot cover is lost, please order the replacement TA5300-CVR (1SAP187500R0001).
Never power up the CPU with uncovered option board slot, otherwise it may cause serious injury and/or damage the product.

### 4.3.11.2 Inserting and removing a memory card

AC500-eCo V3


1 Micro memory card slot cover
2 Micro memory card
3 Micro memory card slot

1. Open the micro memory card slot cover by turning it upwards.
2. Carefully insert the micro memory card into the micro memory card slot as far as it will go. Observe orientation of card.
3. Close the micro memory card slot cover by turning it downwards.

## NOTICE!

## Disturbed PLC operation

Do not remove the memory card when it is working!
Otherwise the memory card and/or files on it might get corrupted and/or normal PLC operation might be disturbed.

- Remove the memory card only when no black square ( $\square$ ) is shown next to MC in the display.


1 Micro memory card slot cover
2 Micro memory card
3 Micro memory card slot

1. Open the micro memory card slot cover by turning it upwards.
2. Micro memory card can be removed from the micro memory card slot by gripping and pulling with two fingers.
3. Close the micro memory card slot cover by turning it downwards.

### 4.3.11.3 Further accessories

This section only describes accessories that are frequently used for system assembly, connection and construction.
«4 All additional accessories that can be used to supplement the AC500 system

### 4.3.12 Control panels CP600

Refer to the installation instructions delivered with each control panel and to the operating instructions.

### 4.4 Wiring

### 4.4.1 Grounding concept

NOTICE!

## PLC damage due to missing grounding

- Make sure to ground the devices.
- The grounding (switch cabinet grounding) is supplied both by the mains connection (or 24 V supply voltage) and via the DIN rail. The DIN rail must be connected to ground before power is supplied to the device. The grounding may be removed only if it is certain that no more power is being supplied to the control system.
- In case of screw mounting, use metal screws for grounding.

Block diagram:
Digital I/O
modules


Block diagram:
Analog I/O modules


### 4.4.2 EMC-conforming assembly and construction

### 4.4.2.1 General principles

AC500 and AC500-eCo PLC devices are Class II/Class III devices and do not require a Protective Earth (PE) connection.
For proper EMC performance, all metal parts, DIN rails, mounting screws, and cable shield connection terminals are connected to a common ground and provide Functional Earth (FE). This is typically connected to a common reference potential, such as equipotential bonding rails.

Signal Grounds (SGND or GND) are used for signal reference and must not be connected to cable shields, FE or other signals unless otherwise specified in the specific device description.

General consid- Electric and electronical devices have to work correctly on site. This is also valid when electroerations magnetic influences affect them in defined and/or expected strength. The devices themselves must not emit electro-magnetic noises.
Advant controller components have a very high noise immunity.

When the wiring and grounding instructions are met, an error-free operation is given.
High electro-magnetic noises of nearby mounted applications must be taken in consideration during the planning phase.

An EMC compatible earthing concept will also guarantee an error-free operation here.

In order to prevent operating malfunctions, it is recommended, that the operating personnel discharge themselves prior to touching communication connectors or perform other suitable measures to reduce effects of electrostatic discharges.

## There are three important principles to be especially considered:

- Keep all connections as short as possible (in particular the grounding conductors)
- Use large conductor cross sections (in particular for the grounding conductors)
- Create low-impedance, i.e. good and large-sized contacts (in particular for the grounding conductors)

```
Pay attention to the following:
- Use vibration-resistant connections
- Clean metallic contact areas
- Use solid plug and screw-type connections
- Use earth cable shields with clips on a well-grounded metallic surface
- Do not use aluminium parts
- Do not use sheath wires
- Do not use toothed lock washers under screw connections
```



Fig. 15: Assembly: wrong


Fig. 16: Assembly: correct
Make a connection between the DIN rails and PE (Protective Earth). For this, use an grounding wire with a minimum conductor cross section of $10 \mathrm{~mm}^{2}$.

The wire is connected to the DIN rail with an M6 screw.
A large-area contact of the DIN rail with the metallic mounting plate improves the EMC behavior significantly, as the disturbances can be discharged more effective.

### 4.4.2.2 Cable routing

- Route cables meeting the standards.
- Sort the cables into cable groups:
- Power current cables
- Power supply cables
- Signal cables
- Data cables
- Route signal cables and data cables separately from the power cables.
- Separate cable ducts or cable bundles.
- The distance should be 20 cm or greater.
- Lay signal and data cables close to earthed surfaces.


### 4.4.2.3 Cable shields

- Use only shielded data cables. The shield should be grounded at both ends.

A cable shield only grounded at one end can only protect from capacitively coupled interference and low-frequency disturbances ( 50 Hz hum).

- Avoid parasitic currents flowing through the cable shields.

This can be done by installing current-carrying equipotential bondings.

- Use only cables with braided shields.

Foil shields are not robust enough, cannot be contacted well and have poor HF properties.

- Use only metallic or [metal]-plated plugs for shielded data cables.
- Use only shielded cables for analog signals.

For small signals ground the shield only at one end.

- Ground the cable shield directly with a clip when entering the control cabinet. Do not cut the shield until the cable reaches the module connected.

The connection between the PE bar and the shield bar must have a low impedance.

### 4.4.2.4 Control cabinet

> Installation of configurations in shielded cabinet can be required in shipping applications. Follow specific instructions in the applicable type approval certificate.

Available certificates can be found in the internet

1. Open landing page of the product e.g. DO526
2. In the middle of the page, click on [Downloads].
$\Rightarrow$ A two-column window opens on the page with all available documents.
3. In the left column of the downloads page click on [Certificate].
$\Rightarrow$ All available certificates are listed in the right column.

In the list of certificates, the maritime certificates can be identified by the title. The title consists of the abbreviation of the maritime classification society followed by the words: Type Approval Certificate. E.g. "DNV Type Approval Certificate: ...".

Connections The connections between the control cabinet, the mounting plates, the PE bar and the shield bar must have a low impedance.

Grounding Ground the control cabinet doors with short and highly flexible conductors.

Control cabinet Only use lighting with interference suppression.
lighting

For supplying Use the mains socket which is located inside the control cabinet. the PC

### 4.4.2.5 Reference potential

- Provide a uniform reference potential in the entire installation and ground all electrical appliances if possible.
- Route your grounding conductors in a star configuration so that no ground loops can occur.


### 4.4.2.6 Equipotential bonding

The Installation of equipotential bondings are necessary if there are present or expected potential differences between parts of your application.

- The impedance of equipotential bonding must be equal or lower than $10 \%$ of the shield impedance of the shielded signal cables between the same points.
- The conductor cross section of a equipotential bonding must be $16 \mathrm{~mm}^{2}$ to withstand the maximum possible compensating current.
- Equipotential bondings and shielded signal cables should be laid close to each other.
- Equipotential bondings must be connected to PE with low impedance.


Fig. 17: AC500, equipotential bonding
1 Cabinet 1
2 Cabinet 2
3 Power supply for the CPU
4 Fuse for the CPU power
5 Power supply for the I/Os
6 Fuse for the I/O power
7 For fuses for the contacts of the relay outputs
8 OV rail
9 Grounding of the 0 V rail
10 Cabinet grounding
11 Equipotential bonding between the cabinets min. $16 \mathrm{~mm}^{2}$
12 Cable shields grounding
13 Fieldbus connection (e.g. Ethernet)

### 4.4.3 Connection and wiring of the PLC platform

### 4.4.3.1 Safety instructions

## - NOTICE! <br> Attention: <br> The devices should be installed by experts who are trained in wiring electronic devices. In case of bad wiring, the following problems could occur: <br> - On the terminal base, the terminals $L+$ and $M$ are doubled. If the power supply is badly connected, a short circuit could happen and lead to a destruction of the power supply or its fuse. If no suitable fuse exists, the terminal base itself might be destroyed. <br> - The terminal bases and all electronic modules and terminal units are protected against reverse polarity. <br> - All necessary measures should be carried out to avoid damages to modules and wiring. Notice the wiring plans and connection examples.

## NOTICE!

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and temporary overvoltage up to 30 V DC.

## NOTICE!

## Attention:

Due to possible loss of communcation, the communication cables should be fixed with cable duct or bracket or clamp during application.

### 4.4.3.2 AC500

### 4.4.3.2.1 Power supply for AC500 system

The system is powered by two different power circuits:

- The power supply for the processor module is provided through the terminals $\mathrm{L}+/ \mathrm{M}$.
- The power supply for the I/O modules is provided through the terminals UP/ZP.

The power supply for the processor module is galvanic isolated from the power supply for the I/O modules.
As soon as the power supply for the processor module is present, the processor module starts. The power supply of the processor module and the I/O modules should be powered on the same time, otherwise the processor module will not switch to run mode after startup.
When during operation the power supply falls below the minimum process and supply voltage for more than 10 ms , the processor module switches to safe mode. A restart of the processor module only occurs by switching the power supply off and on again.
If an I/O module is disconnected during operation from the power supply while the processor module is still powered, the processor module will continue its normal operation on all other powered peripherals (other I/O modules, communication modules and communication interfaces), but freezes the input image. After recovery of the power supply of the affected I/O module it will continue normal operation and inputs and outputs will be updated.
«Chapter 5.1.2.1 "Environmental conditions" on page 166
As power supply for the AC500 system, the ABB power supplies series CP can be used
« Chapter 5.2.8.3.1 "CP-E - Economic range" on page 1325 \& Chapter 5.2.8.3.2 "CP-C. 1 - High performance range" on page 1327.

### 4.4.3.2.2 Power supply for processor modules

The supply voltage of 24 V DC is connected to a removable 5 -pin terminal block. L+/M exist twice. It is therefore possible to feed e.g. external sensors (up to 8 A max. with $1.5 \mathrm{~mm}^{2}$ conductor) via these terminals.

| Pin assignment | Pin Assignment |  | Label | Function | Description |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  | Terminal block removed |  | L+ | +24 V DC | Positive pin of the power supply voltage |
|  |  |  | L+ | +24 V DC | Positive pin of the power supply voltage |
|  |  |  | M | 0 V | Negative pin of the power supply voltage |
|  |  | inserted | M | 0 V | Negative pin of the power supply voltage |
|  |  |  | $\stackrel{1}{\square}$ | FE | Functional earth |

### 4.4.3.2.3 Connection of wires at the spring terminals

## Connection



Fig. 18: Connect the wire to the spring terminal (steps 1 ... 3)


Fig. 19: Connect the wire to the spring terminal (steps $4 \ldots 7$ )

1. Side view (open terminal drawn for illustration)
2. The top view shows the openings for wire and screwdriver
3. Insert screwdriver ( $2.5 \times 0.4$ to $3.5 \times 0.5 \mathrm{~mm}$ ) at an angle, screwdriver must be at least 15 mm free of insulation at the tip
4. While erecting the screwdriver, insert it until the stop (requires a little strength)
5. Screwdriver inserted - terminal open
6. Strip the wire for 7 mm (and put on wire-end ferrule)
7. Insert wire into the open terminal
8. Done

## Disconnection



Fig. 20: Disconnect wire from the spring terminal (steps 1 ... 3)


Fig. 21: Disconnect wire from the spring terminal (steps 4 ... 6)

1. Terminal with wire connected
2. Insert screwdriver ( $2.5 \times 0.4 \ldots 3.5 \times 0.5 \mathrm{~mm}$ ) at an angle, screwdriver must be at least 15 mm free of insulation at the tip
3. While erecting the screwdriver, insert it until the stop (requires a little strength) - terminal is now open
4. Remove wire from the open terminal
5. Done

### 4.4.3.2.4 Terminals for CANopen communication modules



Fig. 22: Combicon, 5-pole, female, removable plug with spring terminals


Fig. 23: Combicon, 5-pole, female, removable plug with spring terminals

Terminal type: Spring terminal

| Number of cores <br> per terminal | Conductor type | Cross section | Stripped conductor <br> end |
| :--- | :--- | :--- | :--- |
| 1 | solid | $0.2 \mathrm{~mm}^{2} \ldots 2.5 \mathrm{~mm}^{2}$ | 10 mm |
| 1 | flexible | $0.2 \mathrm{~mm}^{2} \ldots 2.5 \mathrm{~mm}^{2}$ | 10 mm |
| 1 with wire-end fer- <br> rule (without plastic <br> sleeve) | flexible | $0.25 \mathrm{~mm}^{2} \ldots 2.5 \mathrm{~mm}^{2}$ | 10 mm |
| 1 with wire-end fer- <br> rule (with plastic <br> sleeve) | flexible | $0.25 \mathrm{~mm}^{2} \ldots 2.5 \mathrm{~mm}^{2}$ | 10 mm |

### 4.4.3.2.5 Ethernet connection details

## Ethernet interface

Ethernet is also used for PROFINET, EtherCAT and Modbus TCP connection.

| Pin assignment | Interface | Pin | Signal | Description |
| :---: | :---: | :---: | :---: | :---: |
|  |  | 1 | TxD+ | Transmit data + |
|  |  | 2 | TxD- | Transmit data - |
|  |  | 3 | RxD+ | Receive data + |
|  |  | 4 | NU | Not used |
|  |  | 5 | NU | Not used |
|  |  | 6 | RxD- | Receive data - |
|  |  | 7 | NU | Not used |
|  |  | 8 | NU | Not used |
|  |  | Shield | Cable shield | Functional earth |

${ }^{\Downarrow}$ Supported protocols and used Ethernet ports
(4) Communication via Modbus TCP/IP
${ }^{〔}$ Communication via Modbus RTU

## Wiring

Cable length restrictions

For the maximum possible cable lengths within an Ethernet network, various factors have to be taken into account. Twisted pair cables (TP cables) are used as transmission medium for $10 \mathrm{Mbit} / \mathrm{s}$ Ethernet (10Base-T) as well as for $100 \mathrm{Mbit} / \mathrm{s}$ (Fast) Ethernet (100Base-TX). For a transmission rate of $10 \mathrm{Mbit} / \mathrm{s}$, cables of at least category 3 (IEA/TIA 568-A-5 Cat3) or class C (according to European standards) are allowed. For fast Ethernet with a transmission rate of $100 \mathrm{Mbit} / \mathrm{s}$, cables of category 5 (Cat5) or class D or higher have to be used. The maximum length of a segment, which is the maximum distance between two network components, is restricted to 100 m due to the electric properties of the cable.

Furthermore, the length restriction for one collision domain has to be observed. A collision domain is the area within a network which can be affected by a possibly occurring collision (i.e. the area the collision can propagate over). This, however, only applies if the components operate in half-duplex mode since the CSMA/CD access method is only used in this mode. If the components operate in full-duplex mode, no collisions can occur. Reliable operation of the collision detection method is important, which means that it has to be able to detect possible collisions even for the smallest possible frame size of 64 bytes ( 512 bits). But this is only guaranteed if the first bit of the frame arrives at the most distant subscriber within the collision domain before the last bit has left the transmitting station. Furthermore, the collision must be able to propagate to both directions at the same time. Therefore, the maximum distance between two ends must not be longer than the distance corresponding to the half signal propagation time of 512 bits. Thus, the resulting maximum possible length of the collision domain is 2000 m for a transmission rate of $10 \mathrm{Mbit} / \mathrm{s}$ and 200 m for $100 \mathrm{Mbit} / \mathrm{s}$. In addition, the bit delay times caused by the passed network components also have to be considered.

Table 11: Specified cable properties of the respective cable types per 100 m :

| Parameter | 10Base-T [10 MHz] | 100Base-TX [100 MHz] |
| :--- | :--- | :--- |
| Attenuation [dB/100m] | 10.7 | 23.2 |
| NEXT [dB/100m] | 23 | 24 |
| ACR [dB/100m] | N/A | 4 |
| Return loss [dB/100m] | 18 | 10 |
| Wave impedance $[\mathrm{Ohms}]$ | 100 | 100 |
| Category | 3 or higher | 5 |
| Class | C or higher | D or higher |

TP cable The TP cable has eight wires arranged in four pairs of twisted wires. Different color codes exist for the coding of the wires, the coding according to EIA/TIA 568, version 1, being the one most commonly used. In this code, the individual pairs are coded with blue, orange, green and brown color. One wire of a pair is unicolored and the corresponding second wire is striped, the respective color alternating with white. For shielded cables, a distinction is made between cables that have one single shield around all pairs of wires and cables that have an additional individual shield for each pair of wires. The following table shows the different color coding systems for TP cables:

Table 12: Color coding of TP cables:

| Pairs | EIA/TIA 568 <br> Version 1 |  | EIA/TIA 568 <br> Version 2 |  | DIN 47100 |  | IEC 189.2 |  |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Pair 1 | white/ <br> blue | blue | green | red | white | brown | white | blue |
| Pair 2 | white/ <br> orange | orange | black | yellow | green | yellow | white | orange |
| Pair 3 | white/ <br> green | green | blue | orange | grey | pink | white | green |
| Pair 4 | white/ <br> brown | brown | brown | slate | blue | red | white | brown |

Two general variants are distinguished for the pin assignment of the normally used RJ45 connectors: EIA/TIA 568 version A and version B. The wiring according to EIA/TIA 568 version $B$ is the one most commonly used.


Fig. 24: Pin assignment of RJ45 sockets

## Cable types

## Crossover cable

## Particular use

Crossover cables are needed only for a direct Ethernet connection without crossover functionality. In particular for AC500 modules in product life cycle phase "Classic".

Crossover cables are for a direct Ethernet connection of two terminal devices as the simplest variant of a network. From transmission lines of the first station to the reception lines of the second station.


Fig. 25: Wiring of a crossover cable

Straight-through For networks with more than two subscribers, hubs or switches have to be used additionally for cable distribution. These active devices already have the crossover functionality implemented which allows a direct connection of the terminal devices using straight-through cables.


Fig. 26: Wiring of a straight-through cable

## CAUTION!

## Risk of communication faults!

When using inappropriate cables, malfunctions in communication may occur.
Only use network cables of the categories 5 (Cat 5, Cat 5e, Cat 6 or Cat 7) or higher within PROFINET networks.

### 4.4.3.3 AC500-eCo

### 4.4.3.3.1 Power supply

The processor modules PM50x2 can be connected to the 24 V DC supply voltage via a removable 3-pin spring terminal block or a 3-pin screw terminal block.

Table 13: Removable terminal block for the supply voltage 24 V DC

| 3-pin spring terminal block | 3-pin screw terminal block |
| :---: | :---: |
|  |  |

The terminal block is available as a set for AC500-eCo V3 processor modules.

| Basic CPU (PM5012) |  | Standard CPUs (PM5032, PM5052) and <br> Pro CPUs (PM5072, PM5082) |  |
| :--- | :--- | :--- | :--- |
| Spring type | Screw type | Spring type | Screw type |
| TA5211-TSPF-B | TA5211-TSCL-B | TA5212-TSPF | TA5212-TSCL |

Further information on power supply and onboard inputs/outputs of the terminal blocks

| Pin assignment | Pin Assignment | Pin | Label | Function | Description |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | 1 | $\stackrel{1}{=}$ | FE | Functional earth |
|  | $123$ | 2 | L+ | +24 V DC | Positive pin of the power supply voltage |
|  | $๑ \bigcirc \bigcirc$ | 3 | M | 0 V | Negative pin of the power supply voltage |
|  | Terminal block inserted |  |  |  |  |

## NOTICE!

## Risk of damaging the PLC due to improper voltage levels!

- Never exceed the maximum tolerance values for process and supply voltages.
- Never fall below the minimum tolerance values for process and supply voltages.
Observe the system data and the technical data of the used module.
» Chapter 5.1.1 "System data AC500-eCo" on page 159


### 4.4.3.3.2 Ethernet

Ethernet interface

Ethernet is also used for Modbus TCP connection.

The Ethernet interface is carried out via a RJ45 jack. The pin assignment of the Ethernet interface:

| Interface | Pin | Description |  |
| :---: | :---: | :---: | :---: |
|  | 1 | Tx+ | Transmit Data + |
|  | 2 | Tx- | Transmit Data - |
|  | 3 | Rx+ | Receive data + |
|  | 4 | NC | Not connected |
|  | 5 | NC | Not connected |
|  | 6 | Rx- | Receive data - |
|  | 7 | NC | Not connected |
|  | 8 | NC | Not connected |
|  | Shield | Cable shield | Functional earth |

③ Supported protocols and used Ethernet ports
② Communication via Modbus TCP/IP
ⓨ Communication via Modbus RTU

### 4.4.3.4 Control panels CP600

Refer to the installation instructions delivered with each control panel and to the operating instructions.

### 4.4.4 Connection and wiring of the modules

Specific information about the connections of the modules and the wiring is available in the installation instructions.

- Installation instructions of AC500 PLC product family
- Installation instructions of CP600 control panels


### 4.5 Configuration and programming

| Advanced installation of engineering suite | Detailed information is available in the installation and licensing document. |
| :---: | :---: |
| Programming for PLC system | « Chapter 6.3 "Configuration in Automation Builder for AC500 V3 products" on page 1398 |

Activating soft- Detailed information is available in the installation and licensing document. ware licenses

### 4.6 Commissioning

Commissioning Preconditions: the PLC

- All PLC modules are installed and wired according to the configuration in Automation Builder.
- The PLC is connected to the power supply.
- The application and the necessary licenses are available.



## WARNING!

## Risk of injury or damage to equipment

To avoid endangering persons or equipment, check before commissioning:

- Mechanical and electrical installation
- Electrical safety of the installation
- Safety functions

1. Switch on the power supply.
$\Rightarrow$ The green PWR LED goes on.
$\Rightarrow$ Processor modules with display: Wait until the processor module displays "Update".

2. Install the firmware and the application on the processor module.

Installing the firmware and the application on a very new processor module is the same process than updating a firmware and an application on a CPU.
\& Chapter 4.7.4 "PLC firmware/application update" on page 148
$\Rightarrow$ The PLC starts with the new firmware and the new application.

Commissioning Preconditions:
the control panel

## WARNING!

## Risk of injury or damage to equipment

To avoid endangering persons or equipment, a functional test of the application must be performed before the final commissioning of the system.

- The control panel is installed and wired.
- The control panel is connected to the power supply.
- The project file and the necessary licenses are available.

1. Start the control panel.
2. Set the IP address of the control panel.
3. Download the project file to the control panel.
4. Connect the PLC and the control panel with an Ethernet cable.

## Commis- Step-by-step instructions using the example of the AC500-eCo starter kit with a control panel sioning example included

Protect your data, provide your projects with passwords and access restrictions.
« Chapter 6.4.1.6 "Protecting and Saving Projects" on page 1819

* Chapter 6.3.1.2.1 "User and access rights" on page 1409

The PLC has an integrated diagnosis system to output diagnosis messages, e.g., parameter setting is wrong or battery low. A diagnosis message is indicated by ERR LED $=$ on.
をy Chapter 4.8 "Troubleshooting" on page 149

### 4.7 Operation

### 4.7.1 Operating modes

The common operating mode is called run mode.

Table 14: Run mode

| LED | Display | Description |
| :---: | :---: | :---: |
| $\begin{aligned} & \text { PWR LED }=\text { on } \\ & \text { RUN LED }=\text { on } \\ & \text { ERR LED }=\text { off } \end{aligned}$ |  | Power supply is present. PLC is in run mode. |

If the PLC is connected to the power supply, but does not operate, it is in stop mode. In stop mode, the processor module does not control the system.

Table 15: Stop mode

| LED | Display | Description |
| :---: | :---: | :---: |
| $\begin{aligned} & \text { PWR LED = on } \\ & \text { RUN LED }=\text { off } \\ & \text { ERR LED }=\text { off/on } \end{aligned}$ |  | E.g., after the download of an application, after a firmware update, an error occured or the PLC was stopped by the user, etc. |

### 4.7.2 Diagnosis system

The PLC has an integrated diagnosis system to output diagnosis messages, e.g., parameter setting is wrong or battery low. A diagnosis message is indicated by ERR LED = on.
*) Chapter 4.8 "Troubleshooting" on page 149

### 4.7.3 Status LEDs, display and control elements

Depending on the device type, various operating elements provided on the front panel can be used to control the devices of the PLC system and/or to change the operating mode.
Operating elements:

- Status LEDs:

Indicates the availability of devices and components such as communication modules, communication interface modules or function modules. Functionality and diagnosis of the status LEDs depends on the specific module and is described in the device description of the appropriate module. Possible status: on/off/blinking

- I/O LEDs:

Displays the status of the inputs and outputs.

- Display:

Available for some processor modules. It can be used for simple configurations and for reading out diagnosis information.

* Chapter 6.8.2.5 "Display, LEDs and function keys on the front panel" on page 4452
\& Chapter 6.9.1.3 "Diagnosis in CPU display" on page 4607
\& Chapter 4.8 "Troubleshooting" on page 149
- Function keys and switches:

Allows to change the current operating modes/status manually Chapter 6.8.2.5.5
"Description of the function keys" on page 4457.

### 4.7.4 PLC firmware/application update

During runtime it might be necessary to update the firmware of the modules, e.g., for security reasons.
Updating the firmware and the application on a processor module is the same process than installing a firmware and an application on a very new processor module.
The installation can be performed either via a PC (recommended) or via a memory card.

* Chapter 6.3.1.4 "Firmware identification and update" on page 1419


Fig. 27: The "Version information" tab of a device in the Automation Builder

### 4.7.5 Control panel firmware/application update

The system components can be updated via Panel Builder software or via USB flash drive. Refer to the Panel Builder manual for a detailed description.

### 4.7.6 Reset Warm

## PLC

$\square$ PLC is in STOP mode.

1. Press and hold ( $>5 \mathrm{sec}$ ) $[R U N]$.

$$
\Rightarrow \text { RUN LED }=0 N .
$$

"Reset" is displayed.
2. Press $[O K]$ to reset ${ }^{+}$the PLC.

Control panel The factory restore of a control panel is described in the Panel Builder manual.

### 4.8 Troubleshooting

### 4.8.1 General

Diagnosis messages are shown in the engineering suite Automation Builder. The diagnosis information can be accessed in the IEC application and can be forwarded to, e.g., a web server or a control panel. $y^{\Perp}$ Chapter 6.9.1.5 "Diagnosis in IEC application" on page 4615
Diagnosis messages can also be read out from the display of the PLC. ${ }^{\circ}$ Chapter 6.9.3 "Diagnosis messages" on page 4655

Some diagnosis messages need to be acknowledged by the user (alarms ${ }^{+}$), others do not require acknowledgment (events ${ }^{+}$). The diagnosis system stores alarms and events in a diagnosis history.
Diagnosis messages include the severity of an error. Error severity can be used for defining system behavior, e.g., activating the error LED or stop the PLC.

### 4.8.2 Possible malfunctions

## Possible mal-

functions

| Fault description | Cause | Remedy |
| :---: | :---: | :---: |
| A processor module, a communication interface module or an I/O module can not be plugged on the corresponding terminal base or terminal unit. | Terminal bases and terminal units have a mechanical encoding. The encoding makes it impossible to insert a module to the wrong place to prevent dangerous voltages or destroyed modules. | Check the given design scheme of your control cabinet. <br> Get in touch with the planner of the design scheme. <br> Check the terminal base compatibility and the terminal unit compatibility. |
| An AC500-eCo V3 processor module does not start. | Configuration error due to wrong firmware of the option boards. | Update the option boards TA5141-RS232I, TA5142RS485I and TA5142-RS485 at least to system firmware 3.5.0_HF-7 and boot firmware 3.5.1 or higher. Use Automation Builder 2.5.2 or higher. <br> * Chapter 5.2.1.1.3.9 "TA5141-RS232I - Option board for COMx serial communication" on page 285 <br> « Chapter 5.2.1.1.3.10 "TA5142-RS485I - Option board for COMx serial communication" on page 289 <br> * Chapter 5.2.1.1.3.11 "TA5142-RS485- Option board for COMx serial communication" on page 296 |
| The module does not respond and is not available for online connection. | Module is not connected to the power supply. Or module is not functional any more. | Check power supply. <br> Replace the module or HMI . |


| Fault description | Cause | Remedy |
| :---: | :---: | :---: |
| The module does not respond and is not available for online connection. | Module is not connected to the power supply. Or module is not functional any more. | ② Chapter 4.9.2 "Replace an AC500 module" on page 152 |
| I/O module on a hot swap terminal unit does not respond and is not available for online connection. | I/O module is not plugged or damaged. | As soon as the correct and operational I/O module is plugged on the terminal unit, the module is configured and ready to start. <br> « Hot swap behavior of I/O modules <br> H Hot swap behavior of remote I/O modules |
| After hot swap of a (remote) I/O module, the system and/or I/O module does not start. | Wrong I/O module type plugged. The type must be the same as of the replaced module. | Plug the correct I/O module type. |
| The display of the control panel is black. | Control panel is not connected to the power supply. Or control panel is not functional any more. | Check power supply. <br> Replace the control panel. |
| ERR LED of the processor module flashes fast. The display shows "AC500". | A fatal system error occured. | Contact ABB technical support |
| ERR LED of the processor module flashes. RUN LED is off. The display shows "FAIL". | Installation or update of firmware, boot project or application failed. | Restart the device. Perform the installation or update again. <br> Chapter 4.7.4 "PLC firmware/application update" on page 148 |
| ERR LED is on. | PLC with event ${ }^{+}$or unacknowledged alarm ${ }^{+}$. | - Use Automation Builder, IEC application or other external access available in your project, e.g., a control panel, to read out and acknowledge the diagnosis messages. <br> In Automation Builder, double-click on a device and select the "Diagnosis" tab to access the diagnosis messages. <br> Or: <br> - Read out the diagnosis data on the display of the processor module. <br> ⓨ Chapter 6.9.3 "Diagnosis messages" on page 4655 |
| The display of the processor module shows "Demo" for 5 min at every license check. (For processor modules without display only visible when logged in with Automation Builder.) | PLC runs in demo mode because at least one feature license is missing. The demo license is valid for 10 days PLC in run. After this time, PLC will go to stop mode. | You can buy the missing license or extend the demo license via Automation Builder. The PLC will not switch back to run mode until a new license is activated or the licensed features are removed from the PLC application. <br> ²) Chapter 6.2.4 "Managing your licenses" on page 1350 <br> ²) Chapter 6.3.2.2.2 "PLC runtime licensing" on page 1446 |
| Every 10 min , the display of the processor module shows "Grace" for 5 min. | PLC runs in grace mode because at least one feature license which has been available disappeared. If the grace time expires, PLC will go to stop mode. | You need to buy or activate the missing. We recommend to use the Automation Builder for activation. After the PLC is in stop mode, it will not switch back to run mode until a new license is activated or the licensed features are removed from the PLC application. <br> « Chapter 6.2.4 "Managing your licenses" on page 1350 |


| Fault description | Cause | Remedy |
| :--- | :--- | :--- |
| Every 10 min, the display <br> of the processor module <br> shows "Grace" for 5 min. | PLC runs in grace mode <br> because at least one feature <br> license which has been avail- <br> able disappeared. II the grace <br> time expires, PLC will go to <br> stop mode. | on page 1446 6.3.2.2.2 "PLC runtime licensing" |
| The CPU displays <br> "noConn" (no connection). | No communication between <br> CPU and display is possible <br> Cue do very high CPU load <br> (e.g., endless loop in user <br> program and not activated <br> task watchdog). | Check the application for loops and operations <br> they need time. For example file handling. Wait for <br> response from other device who will block the exe- <br> cution from other code. |

If a problem persists, contact ABB technical support. In order to be able to solve your problem as soon as possible, please provide the affected Automation Builder project or Panel Builder project, the log files and the firmware versions. How to read out log files is described in the application note How to get log information.

### 4.8.3 Execution of a hotfix

Updates (also hotfixes) are displayed at every system start, if available. $\Leftrightarrow$ Chapter 6.2.2 "Automation Builder updates" on page 1350

Hotfixes can be in different files - *.exe, *.package or Linux (for communication interface modules).

Hotfix file via
A hotfix PLC firmware file (*.exe) can be installed via a memory card. ${ }^{*} \Rightarrow$ Chapter 6.3.1.4.5.2.3 memory card "Execution of update via memory card" on page 1430

Communication The firmware of AC500 communication interface modules can be updated with the IP configurainterface (CI) module firmware

Advanced: Manual installation in
Automation Builder
tion tool.

1. Download the current firmware hotfix from the PLC website.
2. Save and unpack the .zip file at any location of your hard disc.
3. Save and close open projects in Automation Builder.
4. In the menu bar of the Automation Builder open the "Package Manager". "Tools $\rightarrow$ Package Manager" $\Rightarrow$ The "Package Manager" opens.
5. Follow the instructions. ${ }^{4}$ Chapter 6.4.1.21.3.9.4 "Command 'Package Manager'" on page 2655

### 4.9 Maintenance

### 4.9.1 Maintenance intervals

| Interval | Maintenance work |
| :--- | :--- |
| After 3 years of utilisation or as soon as pos- <br> sible after the PLC diagnosis message "low <br> battery warning" | Replace battery of the CPU. |
| Regularly or when the faceplate of the control <br> panel is dusty. | Keep the faceplate of the control panel clean. <br> Refer to the panel's operating instruction. |

### 4.9.2 Replace an AC500 module

## CAUTION!

## Risk of injury and damaging the product!

Improper installation and maintenance may result in injury and can damage the product!

- Installation and maintenance have to be performed according to the technical rules, codes and relevant standards, e.g. EN 60204-1.
- Only by qualified personnel.



## WARNING!

## Removal/Insertion under power

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The devices must not be opened when in operation. The same applies to the network interfaces.

All modules of the AC500 range are maintenance free. Spare parts are not available and repair is not allowed. If a module is not functional any more, you have to replace the whole module.
If an I/O module is installed on a hot swap terminal unit, it is possible to replace it with the same type during runtime.

* Chapter 4.9.3 "Replace an I/O module with hot swap" on page 153

If an I/O module is installed on a terminal unit that does not support hot swap and for all other AC500 modules, proceed as follows.

1. Disconnect the power supply.
2. For processor modules, insert the memory card into the new module.
3. For communciation modules, adjust the position of the rotary switches on the new module.
4. For terminal units and terminal bases, disconnect the connectors from the old module.
5. Demount the module to be replaced as described in the installation instructions.
6. Mount the new module as described in the installation instructions.
7. For terminal units and terminal bases, connect the connectors.
8. Reconnect the power supply.
$\Rightarrow$ System boots up again.
9. If necessary, update module firmware ${ }^{\star y}$ Chapter 4.7.4 "PLC firmware/application update" on page 148.

If a processor module is replaced, you must install the firmware, the application, the licenses, passwords and the certificates on the new processor module.
\& Chapter 6.3.1.4.2.2 "CPU firmware" on page 1419
« Chapter 4.7.4 "PLC firmware/application update" on page 148
② Chapter 6.3.2.2.2 "PLC runtime licensing" on page 1446
ⓨ Chapter 6.6.2.2 "Encrypted Communication with Devices via Controller Certificates" on page 4201
«y Chapter 6.4.1.6 "Protecting and Saving Projects" on page 1819
« Chapter 6.4.1.11 "Downloading an Application to the PLC" on page 1965

If a problem persists, contact $A B B$ technical support.

### 4.9.3 Replace an I/O module with hot swap

### 4.9.3.1 Preconditions for using hot swap

## Conditions for hot swapping

- Digital outputs are not under load.
- Input/output voltages above safety extra low voltage/ protective extra low voltages (SELV/PELV) are switched off.
- Modules are completely plugged on the terminal unit with both snap fit engaged before switching on loads or input/output voltage.


## Hot swap

## WARNING!

Risk of explosion or fire in hazardous environments during hot swapping!
Hot swap must not be performed in flammable environments to avoid life-threatening injury and property damage resulting from fire or explosion.

## WARNING!

## Electric shock due to negligent behavior during hot swapping!

To avoid electric shock

- make sure the following conditions apply:
- Digital outputs are not under load.
- Input/output voltages above safety extra low voltage/ protective extra low voltage (SELV/PELV) are switched off.
- Modules are fully interlocked with the terminal unit with both snap-fits engaged before switching on loads or input/output voltage.
- Never touch exposed contacts (dangerous voltages).
- Stay away from electrical contacts to avoid arc discharge.
- Do not operate a mechanical installation improperly.


## NOTICE!

Risk of damage to I/O modules!
Hot swapping is only allowed for I/O modules.
Processor modules and communication interface modules must not be removed or inserted during operation.

H = Hot swap

## Hot swap

System requirements for hot swapping of I/O modules:

- Types of terminal units that support hot swapping of I/O modules have the appendix TU5xx-H.
- I/O modules as of index F0.

The following I/O bus masters support hot swapping of attached I/O modules:

- Communication interface modules CI5xx as of index F0.
- Processor modules PM56xx-2ETH with firmware version as of V3.2.0.

Hot swap is not supported by AC500-eCo V3 CPU!


The index of the module is in the right corner of the label.

## NOTICE!

Risk of damage to I/O modules!
Modules with index below F0 can be damaged when inserted or removed from the terminal unit in a powered system.

## NOTICE!

## Risk of damage to I/O modules!

Do not perform hot swapping if any I/O module with firmware version lower than 3.0.14 is part of the I/O configuration.

For min. required device index see table below.

| Device | Min. required device index for I/O module as of <br> FW Version 3.0.14 |
| :--- | :--- |
| AC522(-XC) | F0 |
| Al523 (-XC) | D2 |
| A1531 | D4 |
| A1531-XC | D2 |
| A1561 | B2 |
| Al562 | B2 |
| A1563 | B3 |
| AO523 (-XC) | D2 |
| AO561 | D2 |
| AX521 (-XC) | D2 |
| AX522 (-XC) | B2 |
| AX561 | D1 |
| CD522 (-XC) | D2 |
| DA501 (-XC) | F0 |
| DA502 (-XC) | D2 |
| DC522 (-XC) | D2 |
| DC523 (-XC) | D2 |
| DC532 (-XC) | A2 |
| DC562 | D2 |
| DI524 (-XC) | B2 |
| DI561 | B2 |
| D1562 | B2 |
| D1571 | A1 |
| D1572 |  |
|  |  |


| Device | Min. required device index for I/O module as of <br> FW Version 3.0.14 |
| :--- | :--- |
| DO524 (-XC) | A3 |
| DO526 | A2 |
| DO526-XC | A0 |
| DO561 | B2 |
| DO562 | A2 |
| DO571 | B3 |
| DO572 | B2 |
| DO573 | A1 |
| DX522 (-XC) | D2 |
| DX531 | D2 |
| DX561 | B2 |
| DX571 | B3 |
| FM562 | A1 |

### 4.9.3.2 Compatibility of hot swap

Hot swap is not supported by AC500-eCo V3 CPU!

|  | Central I/O on V3 CPU |
| :--- | :--- |
| I/O module on TU5xx-H connected to I/O bus <br> master | AC500 V3 CPU types: <br> PM56xx-2ETH |
| Required version of I/O bus master | Firmware as of V3.2.0 |
| Fieldbus master when used as remote I/O with <br> AC500 V3 | - |
| When used as remote I/O on third party <br> controller (PLC or DCS) | - |

### 4.9.3.3 Hot swap behavior

| Hot swap behavior | Central I/O on V3 CPU |
| :--- | :--- |
| Start-up behavior with unplugged or damaged <br> I/O module on hot swap terminal unit TU5xx-H | System and I/O modules attached to the CPU <br> are starting (except unplugged or damaged <br> module when plugged on hot swap terminal <br> unit). <br> As soon as the correct and operational I/O <br> module is plugged on the terminal unit, the <br> module is configured and ready to start. <br> No specific setting needed. |
| Start-up behavior with wrong I/O module type <br> on any terminal unit | System and I/O modules are not starting |
| Diagnosis of presence of hot swap terminal <br> unit | Diagnosis using PLC browser command "io- <br> bus desc" in Automation Builder V3. <br> The PLC browser then provides an overview <br> of the modules on the I/O bus including the <br> position of hot swap terminal units in the <br> I/O bus. <br> In the application program this |
| can be detected with a func- |  |
| tion block "IoModuleHotSwapInfo" |  |
| (Library: AC500_lo/Function Blocks/I/O-Bus). |  |
| One instance of function block is needed per |  |
| terminal unit on the I/O bus. The function |  |
| block provides five outputs delivering infor- |  |
| mation about slut number, hot swap capa- |  |
| bility and plugged/unplugged state of the I/O |  |
| module |  |$|$| If module is pulled then diagnosis Err 9480 |
| :--- |
| "Module removed from Hot Swap Terminal |
| Unit" is generated |

### 4.9.4 Replace a CP600 control panel

Control panels are maintenance free. Spare parts are not available and repair is not permitted. If a control panel becomes inoperable, it must be replaced. An operation manual is included with the new unit.

For instructions on how to transfer the runtime to a new control panel and how to load projects to a new control panel, refer to the Panel Builder manual.

The procedure for changing from a CP600 device to a CP600 device of the 2nd generation is described in the application example Converting a CP600 project. A description of the differences between both device variants can also be found in this application example.

### 4.10 Decommissioning

1. Delete the runtime licenses from the device by returning the licenses.
ⓨ Chapter 6.3.2.2.2.5 "Returning a license" on page 1452
2. Delete certificates available on the CPU.
※ Chapter 6.6.2.4.1 "View 'Security Screen' - 'Devices'" on page 4205
3. Delete applications.

* Chapter 6.4.1.21.3.7.12 "Command 'Reset Origin"' on page 2635
*s Chapter 6.4.1.21.3.7.13 "Command 'Reset Origin Device"' on page 2636

4. Delete applications from memory card, if available.

* Chapter 6.4.1.15 "Copying files to/from PLC" on page 2027

5. If available, remove memory card and battery from CPU.
6. Delete all user accounts and user data.

* "Tab 'User"' on page 2593

7. Demount and dispose the hardware modules.
\& Demount AC500, AC500-XC and AC500-S
« Chapter 4.11 "Recycling" on page 158

If you can not access the data stored in the CPU, e.g., because the CPU is not functional any more, then physically destroy the device.
This ensures that the credentials that are stored in the device, can not be misused.

### 4.11 Recycling



## Disposal and recycling information

This symbol on the product (and on its packaging) is in accordance with the European Union's Waste Electrical and Electronic Equipment (WEEE) Directive.

The symbol indicates that this product must be recycled/disposed of separately from other household waste.
It is the end user's responsibility to dispose of this product by taking it to a designated WEEE collection facility for the proper collection and recycling of the waste equipment.
The separate collection and recycling of waste equipment will help to conserve natural resources and protect human health and the environment.
For more information about recycling, please contact your local environmental office, an electrical/electronic waste disposal company or the store where you purchased the product.

## 5 Hardware descriptions

### 5.1 Technical data of the system

### 5.1.1 System data AC500-eCo

### 5.1.1.1 Environmental conditions

Table 16: Process and supply voltages

| Parameter | Value |
| :---: | :---: |
| 24 V DC |  |
| Voltage | 24 V (-15 \%, +20 \%) |
| Protection against reverse polarity | Yes |
| 24 V AC |  |
| Voltage | 24 V (-15 \%, +10 \%) |
| Frequency | 50/60 Hz (-6 \%, +4 \%) |
| 100 V AC ... 240 V AC wide-range supply |  |
| Voltage | 100 V .. 240 V (-15 \%, +10 \%) |
| Frequency | 50/60 Hz (-6 \%, +4 \%) |
| Allowed interruptions of power supply, according to EN 61131-2 |  |
| DC supply | Interruption < 10 ms , time between 2 interruptions $>1$ s, PS2 |
| DC supply (only for analog option boards TA512x) | Interruption < 1 ms , time between 2 interruptions $>1 \mathrm{~s}, \mathrm{PS} 1$ |

## NOTICE!

Risk of damaging the PLC due to improper voltage levels!

- Never exceed the maximum tolerance values for process and supply voltages.
- Never fall below the minimum tolerance values for process and supply voltages.
Observe the system data and the technical data of the used module.
Chapter 5.1.1 "System data AC500-eCo" on page 159


## NOTICE!

Improper voltage level or frequency range which cause damage of $A C$ inputs:

- AC voltage above 264 V
- Frenquency below 47 Hz or above 62.4 Hz


## NOTICE!

Improper connection leads cause overtemperature on terminals.
PLC modules may be destroyed by using wrong cable type, wire size and cable temperature classification.

CPUs

Table 17: Temperature ranges for processor modules revision 0


Table 18: Temperature ranges for processor modules revision 1


| Parameter | Value |  |  |
| :---: | :---: | :---: | :---: |
|  | PM5012-x-ETH | PM5032-x-ETH, PM5052-xETH, PM5072-T-2ETH, PM5082-T-2ETH | PM5072-T-2ETHW |
|  | - |  | Simple coating for accidental condensation |
| Air pressure |  |  |  |
| Operating | > $800 \mathrm{hPa} /<2000 \mathrm{~m}$ |  |  |
| Storage | > $660 \mathrm{hPa} /<3500 \mathrm{~m}$ |  |  |

## Option boards

Table 19: Standard temperature ranges with processor modules revision 0

| Option boards | Configuration | Processor modules | Operating temperature ranges | Derating |
| :---: | :---: | :---: | :---: | :---: |
| Digital I/O option boards |  |  |  |  |
| TA5101-4DI | Not relevant | PM5012-x-ETH | $0^{\circ} \mathrm{C} \ldots+55^{\circ} \mathrm{C}$ | No derating |
| TA5105-4DOT TA5110-2DI2DOT |  | $\begin{aligned} & \text { PM50x2-x-ETH } \\ & \text { PM5072-T-2ETH } \end{aligned}$ | $0^{\circ} \mathrm{C} \ldots+60^{\circ} \mathrm{C}$ | No derating |
| Analog input option boards |  |  |  |  |
| TA5120-2AI-UI TA5123-2AI-RTD | Not relevant | PM5012-x-ETH | $0^{\circ} \mathrm{C} \ldots+55^{\circ} \mathrm{C}$ | No derating |
|  |  | $\begin{aligned} & \text { PM50x2-x-ETH } \\ & \text { PM5072-T-2ETH } \end{aligned}$ | $0^{\circ} \mathrm{C} \ldots+60^{\circ} \mathrm{C}$ | No derating |
| Analog output option boards |  |  |  |  |
| TA5126-2AO-UI | $0 \mathrm{~V} \ldots+10 \mathrm{~V}$ | $\begin{aligned} & \text { PM5012-T-ETH } \\ & \text { PM50x2-R-ETH } \end{aligned}$ | $0^{\circ} \mathrm{C} \ldots+55^{\circ} \mathrm{C}$ | No derating |
|  |  | PM50x2-T-ETH <br> PM5072-T-2ETH | $0^{\circ} \mathrm{C} \ldots+60^{\circ} \mathrm{C}$ | No derating |
|  | $0 \mathrm{~mA} \ldots+20 \mathrm{~mA}$ | PM50x2-x-ETH <br> PM5072-T-2ETH | $0^{\circ} \mathrm{C} \ldots+45{ }^{\circ} \mathrm{C}$ | No derating <br> Load: $0 \Omega$... $500 \Omega$ |
|  |  |  | $+45^{\circ} \mathrm{C} \ldots+50^{\circ} \mathrm{C}$ | $\begin{aligned} & 50 \% \\ & \text { Load: } 250 \Omega \ldots 500 \Omega \end{aligned}$ |
|  |  |  | $+50{ }^{\circ} \mathrm{C} \ldots+55^{\circ} \mathrm{C}$ | $\begin{aligned} & 100 \% \\ & \text { Load: } 500 \Omega \end{aligned}$ |
| Accessory option boards |  |  |  |  |
| TA5130-KNXPB | Not relevant | PM5072-T-2ETH | $0^{\circ} \mathrm{C} \ldots 60^{\circ} \mathrm{C}$ | No derating |
| TA5131-RTC | Not relevant | PM5012-x-ETH | $0^{\circ} \mathrm{C} \ldots+55^{\circ} \mathrm{C}$ | No derating |
| Option boards for serial interface |  |  |  |  |
| TA5141-RS232I | Not relevant | PM5012-x-ETH | $0^{\circ} \mathrm{C} \ldots+55^{\circ} \mathrm{C}$ | No derating |
| TA5142-RS485I TA5142-RS485 |  | $\begin{aligned} & \text { PM50x2-x-ETH } \\ & \text { PM5072-T-2ETH } \end{aligned}$ | $0^{\circ} \mathrm{C} \ldots+60^{\circ} \mathrm{C}$ | No derating |

Table 20: Standard temperature ranges with processor modules revision 1

| Option boards | Configuration | Processor modules | Operating temperature ranges | Derating |
| :---: | :---: | :---: | :---: | :---: |
| Digital I/O option boards |  |  |  |  |
| TA5101-4DI | Not relevant | PM5012-x-ETH | $0^{\circ} \mathrm{C} \ldots+55^{\circ} \mathrm{C}$ | No derating |
| TA5105-4DOT TA5110-2DI2DOT |  | PM50x2-x-ETH <br> PM50x2-T-2ETH | $-20^{\circ} \mathrm{C} \ldots+60^{\circ} \mathrm{C}$ | No derating |
| Analog input option boards |  |  |  |  |
| TA5120-2AI-UI | Not relevant | PM5012-x-ETH | $0^{\circ} \mathrm{C} \ldots+55^{\circ} \mathrm{C}$ | No derating |
| TA5123-2AI-RTD |  | PM50x2-x-ETH <br> PM50x2-T-2ETH | $-20^{\circ} \mathrm{C} \ldots+60^{\circ} \mathrm{C}$ | No derating |
| Analog output option boards |  |  |  |  |
| TA5126-2AO-UI | $0 \vee \ldots+10 \mathrm{~V}$ | PM5012-T-ETH | $0^{\circ} \mathrm{C} \ldots+55^{\circ} \mathrm{C}$ | No derating |
|  |  | PM50x2-R-ETH | $-20^{\circ} \mathrm{C} \ldots+55^{\circ} \mathrm{C}$ | No derating |
|  |  | PM50x2-T-ETH <br> PM50x2-T-2ETH | $-20^{\circ} \mathrm{C} \ldots+60^{\circ} \mathrm{C}$ | No derating |
|  | $0 \mathrm{~mA} \ldots+20 \mathrm{~mA}$ | PM5012-x-ETH | $0^{\circ} \mathrm{C} \ldots+45^{\circ} \mathrm{C}$ | No derating |
|  |  | PM50x2-x-ETH <br> PM50x2-T-2ETH | $-20^{\circ} \mathrm{C} \ldots+45^{\circ} \mathrm{C}$ | Load: $0 \Omega$... $500 \Omega$ |
|  |  | PM50x2-x-ETH <br> PM50x2-T-2ETH | $+45^{\circ} \mathrm{C} \ldots+50^{\circ} \mathrm{C}$ | $\begin{aligned} & 50 \% \\ & \text { Load: } 250 \Omega \ldots 500 \Omega \end{aligned}$ |
|  |  | PM50x2-x-ETH <br> PM50x2-T-2ETH | $+50^{\circ} \mathrm{C} . . .+55^{\circ} \mathrm{C}$ | $100 \text { \% }$ <br> Load: $500 \Omega$ |
| Accessory option boards |  |  |  |  |
| TA5130-KNXPB | Not relevant | PM50x2-T-2ETH | $-20^{\circ} \mathrm{C} \ldots 60^{\circ} \mathrm{C}$ | No derating |
| TA5131-RTC | Not relevant | PM5012-x-ETH | $0^{\circ} \mathrm{C} \ldots+55^{\circ} \mathrm{C}$ | No derating |
| Option boards for serial interface |  |  |  |  |
| TA5141-RS232I | Not relevant | PM5012-x-ETH | $0^{\circ} \mathrm{C} \ldots+55^{\circ} \mathrm{C}$ | No derating |
| TA5142-RS485I TA5142-RS485 |  | PM50x2-x-ETH <br> PM50x2-T-2ETH | $-20^{\circ} \mathrm{C} \ldots+60^{\circ} \mathrm{C}$ | No derating |

Table 21: Wide temperature ranges

| Option boards | Configuration | Processor modules | Operating temperature ranges | Derating |
| :---: | :---: | :---: | :---: | :---: |
| Digital I/O option boards |  |  |  |  |
| TA5101-4DIW TA5105-4DOTW TA5110-2DI2DOW | Not relevant | PM5072-T-2ETHW | $-20^{\circ} \mathrm{C} \ldots+60^{\circ} \mathrm{C}$ | No derating |
|  |  |  | $+60^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ | I/O derating to 75 \% |
|  |  |  |  | Only 3 of 4 I/O channels are allowed to be energized simultaneously. |
| Analog input optio | ards |  |  |  |


| Option boards | Configuration | Processor modules | Operating temperature ranges | Derating |
| :---: | :---: | :---: | :---: | :---: |
| TA5120-2AI-UIW TA5123-2AI-RTW | Not relevant | PM5072-T-2ETHW | $-20^{\circ} \mathrm{C} . . .60^{\circ} \mathrm{C}$ | No derating |
| Analog output option boards |  |  |  |  |
| TA5126-2AO-UIW | $0 \mathrm{~V} . . .+10 \mathrm{~V}$ | PM5072-T-2ETHW | $-20^{\circ} \mathrm{C} \ldots+60^{\circ} \mathrm{C}$ | No derating |
|  | $0 \mathrm{~mA} \ldots+20 \mathrm{~mA}$ | PM5072-T-2ETHW | $-20^{\circ} \mathrm{C} \ldots+45^{\circ} \mathrm{C}$ | No derating <br> Load: $0 \Omega$... $500 \Omega$ |
|  |  |  | $+45^{\circ} \mathrm{C} . . .+50^{\circ} \mathrm{C}$ | 50 \% <br> Load: $250 \Omega$... $500 \Omega$ |
|  |  |  | $+50^{\circ} \mathrm{C} . . .+55^{\circ} \mathrm{C}$ | $100 \text { \% }$ <br> Load: $500 \Omega$ |
| Accessory option boards |  |  |  |  |
| TA5130-KNXPBW | Not relevant | PM5072-T-2ETHW | $-20^{\circ} \mathrm{C} \ldots 70^{\circ} \mathrm{C}$ | No derating |
| Option boards for serial interface |  |  |  |  |
| TA5141-RS232IW TA5142-RS485IW TA5142-RS485W | Not relevant | PM5072-T-2ETHW | $-20^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ | No derating |

### 5.1.1.2 Creepage distances and clearances

The creepage distances and clearances meet the requirements of the overvoltage category II, pollution degree 2 .

### 5.1.1.3 Power supply units

AC500 and AC500-eCo PLC devices are Class II/Class III devices and do not require a Protective Earth (PE) connection.
For proper EMC performance, all metal parts, DIN rails, mounting screws, and cable shield connection terminals are connected to a common ground and provide Functional Earth (FE). This is typically connected to a common reference potential, such as equipotential bonding rails.
Signal Grounds (SGND or GND) are used for signal reference and must not be connected to cable shields, FE or other signals unless otherwise specified in the specific device description.

For the supply of the modules, power supply units according to SELV or PELV specifications must be used.

## Safety Extra Low Voltage (SELV) and Protective Extra Low Voltage (PELV)

To ensure electrical safety of AC500/AC500-eCo extra low voltage circuits, 24 $V$ DC supply, communication interfaces, I/O circuits, and all connected devices must be powered from sources meeting requirements of SELV, PELV, class 2, limited voltage or limited power according to applicable standards.

## WARNING!

Improper installation can lead to death by touching hazardous voltages!
To avoid personal injury, safe separation, double or reinforced insulation and separation of the primary and secondary circuit must be observed and implemented during installation.

- Only use power converters for safety extra-low voltages (SELV) with safe galvanic separation of the primary and secondary circuit.
- Safe separation means that the primary circuit of mains transformers must be separated from the secondary circuit by double or reinforced insulation. The protective extra-low voltage (PELV) offers protection against electric shock.


### 5.1.1.4 Electromagnetic compatibility

Table 22: Electromagnetic compatibility

| Parameter | Value |
| :--- | :--- |
| Device suitable only as Control Equipment for Industrial Applications, including marine applica- <br> tions. <br> IEC 61131-2, zone B <br> を |  |
| Radiated emission according to <br> IEC 61000-6-4 CISPR11, class A | Yes |
| Conducted emission according to <br> IEC 61000-6-4 CISPR11, class A |  |
| Electrostatic discharge (ESD) according to <br> IEC 61000-4-2, criterion B | Air discharge: 8 kV |
| Fast transient interference voltages (burst) <br> according to | Contact discharge: 6 kV |
| IEC 61000-4-4, criterion B | Power supply (DC): 2 kV |
|  | Digital inputs/outputs (24 V DC): 1 kV |
|  | Angital inputs/outputs (240 V AC): 2 kV |


| Parameter | Value |
| :---: | :---: |
| High energy transient interference voltages (surge) according to IEC 61000-4-5, criterion B | Power supply (DC): <br> - Line to ground: 1 kV <br> - Line to line: $0,5 \mathrm{kV}$ <br> Digital inputs/outputs/relay: <br> (24 V DC): <br> - Line to ground: 1 kV (AC): <br> - Line to ground: 2 kV <br> - Line to line: 1 kV <br> Analog inputs/outputs: <br> - Line to ground: 1 kV <br> Communication lines: <br> - Line to ground: 1 kV |
| Influence of radiated disturbances IEC 61000-4-3, criterion A | Test field strength: $10 \mathrm{~V} / \mathrm{m}$ |
| Influence of line-conducted interferences IEC 61000-4-6, criterion A | Test voltage: 10 V |
| Power frequency magnetic fields IEC 61000-4-8, criterion A | $\begin{aligned} & 30 \mathrm{~A} / \mathrm{m} 50 \mathrm{~Hz} \\ & 30 \mathrm{~A} / \mathrm{m} 60 \mathrm{~Hz} \end{aligned}$ |

### 5.1.1.5 Mechanical data

| Parameter | Value |
| :---: | :---: |
| Mounting | Horizontal/vertical |
| Wiring method | Spring/screw terminals |
| Degree of protection | PLC system: IP 20 <br> - with all modules or option boards plugged in <br> - with all terminals plugged in <br> - with all covers closed |
| Housing | Classification V-0 according to UL 94 |
| Vibration resistance (sinusoidal) acc. to IEC 60068-2-6 | All three axes <br> $2 \mathrm{~Hz} \ldots 8.4 \mathrm{~Hz}, 3.5 \mathrm{~mm}$ peak, <br> $8.4 \mathrm{~Hz} . .150 \mathrm{~Hz}, 1 \mathrm{~g}$ |
| Shock test acc. to IEC 60068-2-27 | All three axes $15 \mathrm{~g}, 11 \mathrm{~ms}$, half-sinusoidal |
| Mounting of the modules: |  |
| Mounting Rail Top Hat according to IEC 60715 | 35 mm , depth 7.5 mm or 15 mm |
| Mounting with screws | M4 |
| Fastening torque | 1.2 Nm |

### 5.1.1.6 Approvals and certifications

The PLC Automation catalog contains an overview of the available approvals and certifications.

### 5.1.2 System data AC500

### 5.1.2.1 Environmental conditions

Table 23: Process and supply voltages

| Parameter | Value |
| :---: | :---: |
| 24 V DC |  |
| Voltage | 24 V (-15 \%, +20 \%) |
| Protection against reverse polarity | Yes |
| 100 V AC... 240 V AC wide-range supply |  |
| Voltage | 100 V ... 240 V (-15 \%, +10 \%) |
| Frequency | $50 / 60 \mathrm{~Hz}$ (-6 \%, +4 \%) |
| Allowed interruptions of power supply, according to EN 61131-2 |  |
| DC supply | Interruption < 10 ms , time between 2 interruptions > $1 \mathrm{~s}, \mathrm{PS} 2$ |
| AC supply | Interruption < 0.5 periods, time between 2 interruptions $>1 \mathrm{~s}$ |

## NOTICE!

## Risk of damaging the PLC due to improper voltage levels!

- Never exceed the maximum tolerance values for process and supply voltages.
- Never fall below the minimum tolerance values for process and supply voltages.
Observe the system data * Chapter 5.1.2 "System data AC500" on page 166 and the technical data of the module used.

NOTICE!
Improper voltage level or frequency range which cause damage of AC inputs:

- AC voltage above 264 V
- Frenquency below 47 Hz or above 62.4 Hz


## NOTICE!

Improper connection leads cause overtemperature on terminals.
PLC modules may be destroyed by using wrong cable type, wire size and cable temperature classification.

| Parameter | Value |
| :--- | :--- |
| Temperature |  |
|  | Operating |
|  | $0^{\circ} \mathrm{C} \ldots+60^{\circ} \mathrm{C}:$ Horizontal mounting of modules. |
| $0^{\circ} \mathrm{C} \ldots+40^{\circ} \mathrm{C}$ : Vertical mounting of modules. |  |
| Output load reduced to $50 \%$ per group. |  |


| Parameter |  | Value |
| :--- | :--- | :--- |
|  | Storage | $-40^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ |
|  | Transport | $-40^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ |
| Humidity | Max. $95 \%$, without condensation |  |
| Air pressure |  |  |
|  | Operating | $>800 \mathrm{hPa} /<2000 \mathrm{~m}$ |
|  | Storage | $>660 \mathrm{hPa} /<3500 \mathrm{~m}$ |

### 5.1.2.2 Creepage distances and clearances

The creepage distances and clearances meet the requirements of the overvoltage category II, pollution degree 2.

### 5.1.2.3 Power supply units

AC500 and AC500-eCo PLC devices are Class II/Class III devices and do not require a Protective Earth (PE) connection.
For proper EMC performance, all metal parts, DIN rails, mounting screws, and cable shield connection terminals are connected to a common ground and provide Functional Earth (FE). This is typically connected to a common reference potential, such as equipotential bonding rails.

Signal Grounds (SGND or GND) are used for signal reference and must not be connected to cable shields, FE or other signals unless otherwise specified in the specific device description.

For the supply of the modules, power supply units according to SELV or PELV specifications must be used.

## Safety Extra Low Voltage (SELV) and Protective Extra Low Voltage (PELV)

To ensure electrical safety of AC500/AC500-eCo extra low voltage circuits, 24 $V$ DC supply, communication interfaces, I/O circuits, and all connected devices must be powered from sources meeting requirements of SELV, PELV, class 2, limited voltage or limited power according to applicable standards.

## WARNING!

Improper installation can lead to death by touching hazardous voltages!
To avoid personal injury, safe separation, double or reinforced insulation and separation of the primary and secondary circuit must be observed and implemented during installation.

- Only use power converters for safety extra-low voltages (SELV) with safe galvanic separation of the primary and secondary circuit.
- Safe separation means that the primary circuit of mains transformers must be separated from the secondary circuit by double or reinforced insulation. The protective extra-low voltage (PELV) offers protection against electric shock.


### 5.1.2.4 Electromagnetic compatibility

Table 24: Electromagnetic compatibility

| Parameter | Value |
| :---: | :---: |
| Device suitable only as Control Equipment for Industrial Applications, including marine applications. <br> IEC 61131-2, zone B <br> « Chapter 5.1.2.6 "Approvals and certifications" on page 169 |  |
| Radiated emission according to IEC 61000-6-4 CISPR11, class A | Yes |
| Conducted emission according to IEC 61000-6-4 CISPR11, class A | Yes |
| Electrostatic discharge (ESD) according to IEC 61000-4-2, criterion B | Air discharge: 8 kV <br> Contact discharge: 6 kV |
| Fast transient interference voltages (burst) according to <br> IEC 61000-4-4, criterion B | Power supply (DC): 2 kV <br> Digital inputs/outputs ( 24 V DC): 1 kV <br> Digital inputs/outputs ( 240 VAC ): 2 kV <br> Analog inputs/outputs: 1 kV <br> Communication lines shielded: 1 kV |
| High energy transient interference voltages (surge) according to IEC 61000-4-5, criterion B | Power supply (DC): <br> - Line to ground: 1 kV <br> - Line to line: $0,5 \mathrm{kV}$ <br> Digital inputs/outputs/relay: <br> (24 V DC): <br> - Line to ground: 1 kV <br> (AC): <br> - Line to ground: 2 kV <br> - Line to line: 1 kV <br> Analog inputs/outputs: <br> - Line to ground: 1 kV <br> Communication lines: <br> - Line to ground: 1 kV |
| Influence of radiated disturbances IEC 61000-4-3, criterion A | Test field strength: $10 \mathrm{~V} / \mathrm{m}$ |
| Influence of line-conducted interferences IEC 61000-4-6, criterion A | Test voltage: 10 V |
| Power frequency magnetic fields IEC 61000-4-8, criterion A | $\begin{aligned} & 30 \mathrm{~A} / \mathrm{m} 50 \mathrm{~Hz} \\ & 30 \mathrm{~A} / \mathrm{m} 60 \mathrm{~Hz} \end{aligned}$ |

### 5.1.2.5 Mechanical data

| Parameter | Value |
| :---: | :---: |
| Mounting | Horizontal/Vertical |
| Wiring method | Spring/screw terminals |
| Degree of protection | PLC system: IP 20 <br> - with all modules or option boards plugged in <br> - with all terminals plugged in <br> - with all covers closed |
| Housing | Classification V-2 according to UL 94 |
| Vibration resistance (sinusoidal) acc. to IEC 60068-2-6 | All three axes <br> $2 \mathrm{~Hz} \ldots 8.4 \mathrm{~Hz}, 3.5 \mathrm{~mm}$ peak, $8.4 \mathrm{~Hz} . . .150 \mathrm{~Hz}, 1 \mathrm{~g}$ |
| Shock test acc. to IEC 60068-2-27 | All three axes $15 \mathrm{~g}, 11 \mathrm{~ms}$, half-sinusoidal |
| Mounting of the modules: |  |
| Mounting Rail Top Hat according to IEC 60715 | 35 mm , depth 7.5 mm or 15 mm |
| Mounting with screws | M4 |
| Fastening torque | 1.2 Nm |

### 5.1.2.6 Approvals and certifications

The PLC Automation catalog contains an overview of the available approvals and certifications.

### 5.1.3 System data AC500-XC

### 5.1.3.1 Environmental conditions

Table 25: Process and supply voltages

| Parameter | Value |
| :---: | :---: |
| 24 V DC |  |
| Voltage | 24 V (-15 \%, +20 \%) |
| Protection against reverse polarity | Yes |
| 100 V AC... 240 V AC wide-range supply |  |
| Voltage | 100 V ... 240 V (-15 \%, +10 \%) |
| Frequency | $50 / 60 \mathrm{~Hz}(-6 \%,+4$ \%) |
| Allowed interruptions of power supply, according to EN 61131-2 |  |
| DC supply | Interruption < 10 ms , time between 2 interruptions > $1 \mathrm{~s}, \mathrm{PS} 2$ |
| AC supply | Interruption < 0.5 periods, time between 2 interruptions > 1 s |

## NOTICE!

## Risk of damaging the PLC due to improper voltage levels!

- Never exceed the maximum tolerance values for process and supply voltages.
- Never fall below the minimum tolerance values for process and supply voltages.
Observe the system data ${ }^{\circledR}$ Chapter 5.1.2 "System data AC500" on page 166 and the technical data of the module used.


## NOTICE!

Improper voltage level or frequency range which cause damage of $A C$ inputs:

- AC voltage above 264 V
- Frenquency below 47 Hz or above 62.4 Hz


## NOTICE!

Improper connection leads cause overtemperature on terminals.
PLC modules may be destroyed by using wrong cable type, wire size and cable temperature classification.

| Parameter | Value |
| :---: | :---: |
| Temperature |  |
| Operating | $-40^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ <br> $-40^{\circ} \mathrm{C} \ldots 0^{\circ} \mathrm{C}$ : Due to the LCD technology, the display might respond very slowly. <br> $-40^{\circ} \mathrm{C} \ldots+40^{\circ} \mathrm{C}$ : Vertical mounting of modules possible, output load limited to $50 \%$ per group $+60^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ with the following deratings: <br> - System is limited to max. 2 communication modules per terminal base <br> - Digital inputs: maximum number of simultaneously switched on input channels limited to $75 \%$ per group (e.g. 8 channels => 6 channels) <br> - Digital outputs: output current maximum value (all channels together) limited to 75 \% per group (e.g. $8 \mathrm{~A}=>6 \mathrm{~A}$ ) <br> - Analog outputs only if configured as voltage output: maximum total output current per group is limited to $75 \%$ (e.g. $40 \mathrm{~mA}=>30 \mathrm{~mA}$ ) <br> - Analog outputs only if configured as current output: maximum number of simultaneously used output channels limited to $75 \%$ per group (e.g. 4 channels $=>3$ channels) |
| Storage / Transport | $-40^{\circ} \mathrm{C} \ldots+85^{\circ} \mathrm{C}$ |
| Humidity | Operating / Storage: 100 \% r. H. with condensation |


| Parameter | Value |
| :---: | :---: |
| Air pressure | Operating: $\begin{aligned} & \text {-1000 m .... } 5000 \mathrm{~m}(1080 \mathrm{hPa} \ldots 620 \mathrm{hPa}) \\ & >2000 \mathrm{~m}(<795 \mathrm{hPa}): \end{aligned}$ <br> - Max. operating temperature must be reducted by 10 K for each 1000 m exceeding 2000 m <br> - I/O module relay contacts must be operated with 24 V nominal only |
| Immunity to corrosive gases | Yes, according to: <br> ISA S71.04.1985 Harsh group A, G3/GX IEC60068-2-60 <br> Method 4 with following concentrations: <br> - H2S $100 \pm 10 \mathrm{ppb}$ <br> - NO2 $1250 \pm 20 \mathrm{ppb}$ <br> - CL2 $100 \pm 10 \mathrm{ppb}$ <br> - SO2 $300 \pm 20 \mathrm{ppb}$ |
| Immunity to salt mist | Yes, horizontal mounting only, according to IEC 60068-2-52 severity level: 1 |

## NOTICE!

Risk of corrosion!
Unused connectors and slots may corrode if XC devices are used in salt-mist environments.

Protect unused connectors and slots with TA535 protective caps for XC devices ² Chapter 5.2.8.3.6 "TA535 - Protective caps for XC devices" on page 1332.

## NOTICE!

## Risk of malfunctions!

Unused slots for communication modules are not protected against accidental physical contact.

- Unused slots for communication modules must be covered with dummy communication modules to achieve IP20 rating $\Longleftrightarrow$ Chapter 5.2.8.2.4 "TA524
- Dummy communication module" on page 1323.
- I/O bus connectors must not be touched during operation.


### 5.1.3.2 Creepage distances and clearances

The creepage distances and clearances meet the requirements of the overvoltage category II, pollution degree 2.

### 5.1.3.3 Power supply units

AC500 and AC500-eCo PLC devices are Class II/Class III devices and do not require a Protective Earth (PE) connection.
For proper EMC performance, all metal parts, DIN rails, mounting screws, and cable shield connection terminals are connected to a common ground and provide Functional Earth (FE). This is typically connected to a common reference potential, such as equipotential bonding rails.
Signal Grounds (SGND or GND) are used for signal reference and must not be connected to cable shields, FE or other signals unless otherwise specified in the specific device description.

Safety Extra Low Voltage (SELV) and Protective Extra Low Voltage (PELV)
To ensure electrical safety of AC500/AC500-eCo extra low voltage circuits, 24 $V$ DC supply, communication interfaces, I/O circuits, and all connected devices must be powered from sources meeting requirements of SELV, PELV, class 2, limited voltage or limited power according to applicable standards.

## WARNING!

Improper installation can lead to death by touching hazardous voltages!
To avoid personal injury, safe separation, double or reinforced insulation and separation of the primary and secondary circuit must be observed and implemented during installation.

- Only use power converters for safety extra-low voltages (SELV) with safe galvanic separation of the primary and secondary circuit.
- Safe separation means that the primary circuit of mains transformers must be separated from the secondary circuit by double or reinforced insulation. The protective extra-low voltage (PELV) offers protection against electric shock.


### 5.1.3.4 Electromagnetic compatibility

Table 26: Electromagnetic compatibility

| Parameter | Value |
| :--- | :--- |
| Device suitable only as Control Equipment for Industrial Applications, including marine applica- <br> tions. <br> IEC 61131-2, zone B <br> をy Chapter 5.1.3.6 "Approvals and certifications" on page 174 <br> Radiated emission according to <br> IEC 61000-6-4 CISPR11, class A <br> Conducted emission according to <br> IEC 61000-6-4 CISPR11, class A <br> Electrostatic discharge (ESD) according to <br> IEC 61000-4-2, criterion BYes |  |


| Parameter | Value |
| :---: | :---: |
| Fast transient interference voltages (burst) according to IEC 61000-4-4, criterion B | Power supply (DC): 4 kV <br> Digital inputs/outputs (24 V DC): 2 kV <br> Digital inputs/outputs ( 240 V AC): 4 kV <br> Analog inputs/outputs: 2 kV <br> Communication lines shielded: 2 kV |
| High energy transient interference voltages (surge) according to IEC 61000-4-5, criterion B | Power supply (DC): <br> - Line to ground: 1 kV <br> - Line to line: $0,5 \mathrm{kV}$ <br> Digital inputs/outputs/relay: <br> (24 V DC): <br> - Line to ground: 1 kV (AC): <br> - Line to ground: 2 kV <br> - Line to line: 1 kV <br> Analog inputs/outputs: <br> - Line to ground: 1 kV <br> Communication lines: <br> - Line to ground: 1 kV |
| Influence of radiated disturbances IEC 61000-4-3, criterion A | Test field strength: $10 \mathrm{~V} / \mathrm{m}$ |
| Influence of line-conducted interferences IEC 61000-4-6, criterion A | Test voltage: 10 V |
| Power frequency magnetic fields IEC 61000-4-8, criterion A | $\begin{aligned} & 30 \mathrm{~A} / \mathrm{m} 50 \mathrm{~Hz} \\ & 30 \mathrm{~A} / \mathrm{m} 60 \mathrm{~Hz} \end{aligned}$ |

### 5.1.3.5 Mechanical data

| Parameter | Value |
| :---: | :---: |
| Mounting | Horizontal/vertical (no application in salt mist environment) |
| Wiring method | Spring terminals |
| Degree of protection | PLC system: IP 20 <br> - with all modules or option boards plugged in <br> - with all terminals plugged in <br> - with all covers closed |
| Housing | Classification V-2 according to UL 94 |
| Vibration resistance (sinusoidal) acc. to IEC 60068-2-6 | 2 Hz ... $8.4 \mathrm{~Hz}, 3.5 \mathrm{~mm}$ peak, 8.4 Hz ... $500 \mathrm{~Hz}, 2 \mathrm{~g}$ |
| Vibration resistance (broadband random) acc. to IEC 60068-2-64 | $5 \mathrm{~Hz} \ldots 500 \mathrm{~Hz}, 1,9 \mathrm{~g} \mathrm{rms}$ (operational) <br> $5 \mathrm{~Hz} \ldots 500 \mathrm{~Hz}, 4 \mathrm{~g} \mathrm{rms}$ (non operational) |


| Parameter | Value |
| :--- | :--- |
| Shock resistance | All three axes |
|  | $15 \mathrm{~g}, 11 \mathrm{~ms}$, half-sinusoidal |
| Mounting of the modules: |  |
| Mounting Rail Top Hat according to IEC 60715 | 35 mm, depth 7.5 mm or 15 mm |
| Mounting with screws | M 4 |
| Fastening torque | 1.2 Nm |

### 5.1.3.6 Approvals and certifications

The PLC Automation catalog contains an overview of the available approvals and certifications.

### 5.1.4 AC500-S

Functional The AC500-S safety user manual must be read and understood before using the safety configusafety ration and programming tools of Automation Builder/PS501 Control Builder Plus. Only qualified personnel are permitted to work with AC500-S safety PLCs.

The AC500-S safety PLC includes the following safety-relevant hardware components.

- SM560-S/SM560-S-FD-1/SM560-S-FD-4
- DI581-S
- DX581-S
- AI581-S
- TU582-S


### 5.1.5 CP600

The technical data of the CP600 control panels are included in the CP600 data sheets.

### 5.2 Device specifications

### 5.2.1 Processor modules

### 5.2.1.1 AC500-eCo

5.2.1.1.1 PM50x2

Features


Fig. 28: Example: PM5072-T-2ETH
1 Micro memory card slot
25 LEDs to display the states of the processor module (Power, Error, Run, MC, MOD1)
3 RUN button
4 RJ45 female connector for Ethernet1 connection
5 RJ45 female connector for Ethernet2 connection (available for PM50x2-T-2ETH)
6 3-pin terminal block for power supply 24 V DC
72 holes for screw mounting
8 Option board slot cover for option board slot (the number of available slots varies according to the CPU type)
9 Cable fixing
10 13-pin terminal block for onboard I/Os
11 12-pin terminal block for onboard I/Os (not available on PM5012-x-ETH)
1212 LEDs to display the states of the signals
1310 LED s to display the states of the signals
14 Cable fixing accessory TA5301-CFA on the top of the housing (optional)


The processor module is shown with pluggable terminal blocks. These terminal blocks must be ordered separately.

The cable fixing accessory on the top of the housing is optional.
Please use TA5301-CFA cable fixing accessory to provide strain relief.
It can also be used for AC500-eCo I/O modules.

The PM50x2 processor modules are supplied with option board slot covers as standard.

There are various TA51xx option boards for the processor modules that can be ordered separately.
Which and how many option boards can be plugged, depends on the respective processor module.

Table 27: AC500-eCo V3 CPUs with their most important properties

| Processor modules | Total maximum downloadable application size | Configurable input/output | Digital inputs | Digital outputs | Power supply | Ethernet interfaces | Option board slots |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Basic CPUs |  |  |  |  |  |  |  |
| PM5012-T-ETH | 1 MB <br> thereof 256 kB for user program code and data dynamically allocated | - | 6 | 4 (Transistor) | 24 V DC | 1 | 1 |
| PM5012-R-ETH | 1 MB <br> thereof 256 kB for user program code and data dynamically allocated | - | 6 | 4 (Relay) | 24 V DC | 1 | 1 |
| Standard CPUs |  |  |  |  |  |  |  |
| PM5032-T-ETH | 5 MB <br> thereof 512 kB for user program code and data dynamically allocated | 2 (Transistor) | 12 | 8 (Transistor) | 24 V DC | 1 | 2 |
| PM5032-R-ETH | 5 MB <br> thereof 512 kB for user program code and data dynamically allocated | 2 (Transistor) | 12 | 6 (Relay) | 24 V DC | 1 | 2 |


| Processor modules | Total maximum downloadable application size | Configurable input/output | Digital inputs | Digital outputs | Power supply | Ethernet interfaces | Option board slots |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| PM5052-T-ETH | 7 MB <br> thereof 768 kB for user program code and data dynamically allocated | 2 (Transistor) | 12 | 8 (Transistor) | 24 V DC | 1 | 3 |
| PM5052-R-ETH | 7 MB <br> thereof 768 kB for user program code and data dynamically allocated | 2 (Transistor) | 12 | 6 (Relay) | 24 V DC | 1 | 3 |
| Pro CPUs |  |  |  |  |  |  |  |
| PM5072-T-2ETH | 9 MB <br> thereof 1 MB for user program code and data dynamically allocated | 2 (Transistor) | 12 | 8 (Transistor) | 24 V DC | 2 | 3 |
| $\begin{aligned} & \hline \text { PM5072- } \\ & \text { T-2ETHW *) } \end{aligned}$ | 9 MB <br> thereof 1 MB for user program code and data dynamically allocated | 2 (Transistor) | 12 | 8 (Transistor) | 24 V DC | 2 | 3 |
| PM5082-T-2ETH | 9 MB <br> thereof 1 MB for user program code and data dynamically allocated | 2 (Transistor) | 12 | 8 (Transistor) | 24 V DC | 2 | 3 |

*) W = wide temperature range

## Short description

The processor modules PM50x2 series are the central units of AC500-eCo V3 PLC. Their main characteristics are:

- Power supply 24 V DC
- I/O bus (not for PM5012-x-ETH)
- Real-time clock (PM5012-x-ETH needs additional RTC option board)
- Option board slots for extension on the CPU (1 for PM5012-x-ETH, 2 for PM5032-x-ETH, 3 for PM5052-x-ETH and PM50x2-T-2ETH)
- 6 digital inputs (PM5012-x-ETH), 12 digital inputs (PM5032-x-ETH, PM5052-x-ETH, PM50x2-T-2ETH)
- 4 transistor outputs (PM5012-T-ETH), 8 transistor outputs (PM5032-T-ETH, PM5052-T-ETH, PM50x2-T-2ETH)
- 4 relay outputs (PM5012-R-ETH), 6 relay outputs (PM5032-R-ETH, PM5052-R-ETH)
- 2 configurable digital inputs/outputs (not for PM5012-x-ETH)

The various processor module variants differ in the following characteristics:

- Type of the digital outputs (transistor or relays)
- Ethernet interface one or two independent interfaces

All processor module variants include a micro memory card slot.
Details and technical data are provided in the technical data section $\Leftrightarrow$ Chapter 5 .2.1.1.1.8
"Technical data" on page 188.

## Assortment

Detailed data

| Parameter | Value |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  | PM5012 | PM5032 | PM5052 | PM5072 | PM5082 |
| Total maximum downloadable application size ${ }^{\text {1) }}$ | 1 MB | 5 MB | 7 MB | 9 MB | 9 MB |
| Thereof user program code / data memory dynamically allocated | 256 KB | 512 KB | 768 KB | 1 MB | 1 MB |
| Thereof user web server memory for web visualization max. | no web | 1.5 MB | 3.2 MB | 7 MB | 7 MB |
| Flash memory for user data |  |  |  |  |  |
| Remaining for all other usage (project save, infra- structure...) | 30 MB | 30 MB | 30 MB | 30 MB | 30 MB |
| Buffered (FRAM) | 8 KB | 32 KB | 32 KB | 100 KB | 100 KB |
| VAR_RETAIN persistent | 4 KB | 16 KB | 16 KB | 36 KB | 36 KB |
| \%MB data | 4 KB | 16 KB | 16 KB | 64 KB | 64 KB |
| Expandable memory | None | None | None | None | None |
| Integrated mass storage memory (FLASH) | None | None | None | None | None |
| Slot for pluggable memory card | X | X | X | X | X |
| Real-time clock (RTC) | Optional with TA5131RTC | Built in |  |  |  |
| Min. retention time for RTC at room temperature (if at least powered for 8 hours) | 7 days | 20 days |  |  |  |
| Accuracy | $\pm 2$ s/day |  |  |  |  |
| Programming languages | - Instruction List (IL) <br> - Function Block Diagram (FBD) <br> - Ladder Diagram (LD) <br> - Sequential Function Chart (SFC) <br> - Structured Text (ST) <br> - Continuous Function Chart (CFC) |  |  |  |  |
| Processor type | TI ARM Cortex-A9 32-bit-RISC |  |  |  |  |


| Parameter | Value |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  | PM5012 | PM5032 | PM5052 | PM5072 | PM5082 |
| Processor clock speed | 300 MHz |  |  |  | $\begin{aligned} & \hline 600 \\ & \mathrm{MHz} \end{aligned}$ |
| Calculation time per instructions (minimum) | PM5012 | PM5032 | PM5052 | PM5072 | PM5082 |
| Binary | 20 ns |  |  |  | 10 ns |
| Word | 50 ns |  |  |  | 10 ns |
| Floating point | 600 ns |  |  |  | 10 ns |
| Lowest cycle time usable (also f. PTO) | - | 5 | 2 | 1 | 1 |
| Using onboard EtherCAT protocol (licensed) in preparation |  |  |  |  |  |
| EtherCAT onboard No. of synchronized axis | PM5012 | PM5032 | PM5052 | PM5072 | PM5082 |
| axis per 1 ms CM typically | - | - | - | - | 2-4* |
| axis per 2 msCM typically | - | - | - | 2-4* | 4-8* |
| axis per 4 ms CM typically | - | - | - | 4-8* | 8-16 |
| Cyclic min. configurable | 10 ms | 5 ms | 2 ms | 1 ms | 1 ms |
| Time-controlled | Yes |  |  |  |  |
| Multitasking | Yes |  |  |  |  |
| Interruption | Yes |  |  |  |  |
| LEDs | Power, Error, Run, MC, MOD1, States of I/Os |  |  |  |  |
| RUN/STOP button | Yes |  |  |  |  |
| Protection of the user program by password | On request |  |  |  |  |
| Usable accessories | On request |  |  |  |  |
| Remarks: <br> * Depending an application complexity <br> 1): The values are for information only and cannot be fulfilled altogether. The available resources are limited at the end by the maximal downloadable application size for each CPU. |  |  |  |  |  |


| Data of I/Os | $\begin{aligned} & \text { PM5012-x- } \\ & \text { ETH } \end{aligned}$ | $\begin{aligned} & \text { PM5032-x- } \\ & \text { ETH } \end{aligned}$ | $\begin{aligned} & \text { PM5052-x- } \\ & \text { ETH } \end{aligned}$ | $\begin{array}{\|l} \hline \text { PM5072- } \\ \text { T-2ETH(W) } \end{array}$ | $\begin{array}{\|l\|} \hline \text { PM5082- } \\ \text { T-2ETH } \end{array}$ |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Onboard digital inputs |  |  |  |  |  |
| Channels | 6 <br> (incl. 2 <br> counter inputs 5 kHz and 4 interrupts) | 12 <br> (incl. 4 fast counter/encoder inputs ( $100 \mathrm{kHz} / 200 \mathrm{kHz}$ ), 4 inputs ( 5 kHz ), 4 standard inputs) |  |  |  |
| Signal voltage | 24 V DC type 1 |  |  |  |  |
| Onboard digital outputs |  |  |  |  |  |
| Type of digital outputs | PM5012-T- <br> ETH: <br> Transistor | PM5032-T- <br> ETH: <br> Transistor | PM5052-T- <br> ETH: <br> Transistor | PM5072- <br> T-2ETH(W): <br> Transistor | PM5082-T-2ETH: Transistor |
|  | PM5012-R- <br> ETH: <br> Relay | PM5032-R- <br> ETH: <br> Relay | PM5052-R- <br> ETH: <br> Relay | - | - |


| Data of I/Os | $\begin{aligned} & \text { PM5012-x- } \\ & \text { ETH } \end{aligned}$ | PM5032-xETH | PM5052-xETH | $\begin{aligned} & \hline \begin{array}{l} \text { PM5072- } \\ \text { T-2ETH(W) } \end{array} \end{aligned}$ | $\begin{aligned} & \text { PM5082- } \\ & \text { T-2ETH } \end{aligned}$ |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Channels for transistor version | 4 <br> ( 5 kHz standard and PWM) | 8 (incl. 4 fast outputs for standard or 4 PWM/4 PTO ( $100 \mathrm{kHz} / 200 \mathrm{kHz}$ ), 4 standard outputs ( 5 kHz )) |  |  |  |
| Channels digital input/ output configurable <br> (valid for both PLC version relais or transistor) | - | 2 <br> Relay version <br> The DC chan as $1 \mathrm{PTO} / 2 \mathrm{P}$ standard digit <br> Transistor ve <br> The DC chan be used as st inputs/output | els can be used M (100 kHz) or inputs/outputs ion: <br> els can only ndard digital | 2 <br> Transistor ve The DC chan be used as s inputs/output | on: <br> es can only dard digital |
| Rated voltage transistor | 24 V DC |  |  |  |  |
| Nominal current per transistor channel | 0.5 A resistive |  |  |  |  |
| Channels for relay version | 4 | 6 |  | - | - |
| Rated voltage relay | $100 \text { V AC ... } 240 \text { V AC }$ <br> or $24 \text { V DC }$ |  |  | - | - |
| Nominal current per relay channel | 2 A resistive |  |  | - | - |
| Analog inputs | Optional |  |  |  |  |
| Analog outputs | Optional |  |  |  |  |
| Number of option board slots | 1 | 2 | 3 | 3 | 3 |
| Usage of option board | Each slot can be used for all type of existing option boards, same option board for serial interface or digital/analog I/O extension can be used on several slot per CPU. <br> Note: RTC option board is only for PM5012 possible. |  |  |  |  |
| KNX address switch | No |  |  | TA5130-KNXPB only on 1 slot |  |
| Real-time clock (RTC) | TA5131-RTC | No |  |  |  |
| Serial interface | TA5141-RS232I, TA5142-RS485/TA5142-RS485I |  |  |  |  |
| Digital in/out channels | TA5101-4DI, TA5105-4DOT, TA5110-2DI2DOT |  |  |  |  |
| Analog in/out channels | TA5120-2AI-UI, TA5123-2AI-RTD, TA5126-2AO-UI |  |  |  |  |


| Data of I/Os | $\begin{aligned} & \text { PM5012-x- } \\ & \text { ETH } \end{aligned}$ | $\begin{aligned} & \text { PM5032-x- } \\ & \text { ETH } \end{aligned}$ | $\begin{aligned} & \text { PM5052-x- } \\ & \text { ETH } \end{aligned}$ | $\begin{array}{\|l\|} \hline \text { PM5072- } \\ \text { T-2ETH(W) } \end{array}$ | $\begin{aligned} & \text { PM5082- } \\ & \text { T-2ETH } \end{aligned}$ |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Max. number of I/O modules on I/O bus | 0 | 10 |  |  |  |
| Digital inputs | Onboard I/O only | 128 byte | 1 KB |  |  |
| Digital outputs |  | 128 byte | 1 KB |  |  |
| Number of decentralized inputs and outputs | Depending on the fieldbus used |  |  |  |  |
| Internal interfaces |  |  |  |  |  |
| Serial COMx | Optional, use a dedicated serial interface option board (up to 1) | Optional, use a dedicated serial interface option board (up to 2) | Optional, use a dedicated serial interface option board (up to 3) |  |  |
|  | Modbus RTU Master/Slave, ASCII |  |  |  |  |
| Ethernet interface RJ45 | 1 |  |  | 2 <br> Independent with switch functionality |  |
| Ethernet functions | Programming, TCP/IP, UDP/IP, DHCP, PING, network variables, and other listed below |  |  |  |  |
| Modbus TCP/IP client/server | $\begin{array}{\|l\|} \hline \text { Yes } \\ 8 / 3 \end{array}$ | $\begin{array}{\|l\|} \hline \text { Yes } \\ 13 / 8 \end{array}$ | $\begin{array}{\|l\|} \hline \text { Yes } \\ 20 / 10 \end{array}$ | $\begin{aligned} & \hline \text { Yes } \\ & 30 / 15 \end{aligned}$ | $\begin{array}{\|l\|} \hline \text { Yes } \\ 30 / 15 \end{array}$ |
| SNTP client/ server | No | Yes |  |  |  |
| HTTPs and WebVisu <br> number of connections | No | $\begin{aligned} & \hline \text { Yes } \\ & 1 \end{aligned}$ | $\begin{array}{\|l\|} \hline \text { Yes } \\ 2 \end{array}$ | $\begin{aligned} & \text { Yes } \\ & 4 \end{aligned}$ | $\begin{aligned} & \text { Yes } \\ & 4 \end{aligned}$ |
| FTPs <br> number of connections | No | $\begin{aligned} & \text { Yes } \\ & 1 \end{aligned}$ | $\begin{aligned} & \mathrm{Yes} \\ & 2 \end{aligned}$ | $\begin{aligned} & \text { Yes } \\ & 2 \end{aligned}$ | $\begin{aligned} & \text { Yes } \\ & 2 \end{aligned}$ |
| OPC UA server number of free tags | No | $\begin{aligned} & \text { Yes } \\ & 125 \end{aligned}$ | $\begin{aligned} & \text { Yes } \\ & 250 \end{aligned}$ | $\begin{array}{\|l\|} \hline \text { Yes } \\ 1000 \end{array}$ | $\begin{aligned} & \text { Yes } \\ & 3000 \end{aligned}$ |
| MQTT and JSON library | No | Yes |  |  |  |
| OPC DA server | Yes |  |  |  |  |
| IEC 60870-5-104 telecontrol protocol | No |  |  | Yes <br> Substation only, 5 connections max., <br> only 1 Ethernet supported |  |
| Licensed protocols (runtime protocol per CPU) |  |  |  |  |  |


| Data of I/Os | $\begin{aligned} & \text { PM5012-x- } \\ & \text { ETH } \end{aligned}$ | PM5032-x- <br> ETH | $\begin{aligned} & \text { PM5052-x- } \\ & \text { ETH } \end{aligned}$ | PM5072- <br> T-2ETH(W) | $\begin{aligned} & \text { PM5082- } \\ & \text { T-2ETH- } \end{aligned}$ |
| :---: | :---: | :---: | :---: | :---: | :---: |
| BACnet IP BBC © Chapter 6.3.4.9 <br> "BACnet-BC" <br> on page 1733 | No |  |  | Yes (max. 1000 object variables) |  |
| KNXIP « Chapter 6.3.4.8 "KNX configurator" on page 1730 | No |  |  | Yes (max. 1000 object variables) |  |
| IEC 61850 MMS server/ goose pub/sub | No |  |  | Yes (max. 1000 data attributes) |  |
| EtherNet/IP adapter/ scanner | No | Yes max. 512 byte in / 512 byte out data for adapter |  |  |  |
| EtherCAT Master | No | Yes <br> max. 512 byte in / 512 byte out data for adapter |  |  |  |

## Connections and interfaces

## General

I/O bus

Option board slot interface

The I/O bus is not available for PM5012-T-ETH and PM5012-R-ETH. I/O channel extension using option board slot only.

The I/O bus is the I/O data bus for the I/O modules. Through this bus, I/O and diagnosis data are transferred between the processor module and the I/O modules. Up to 10 I/O modules for PM5032-x-ETH (but with a limit of 128 Bytes input/ 128 Bytes output variables) and 10 I/O modules for PM5052-x-ETH and PM50x2-T-2ETH can be added.

Depending on the processor module variants, an additional option board can be connected to the option board slot to extend the feature of the processor module.
② Chapter 5.2.1.1.3 "Option boards" on page 218

Serial interface RS-232 communication interface is available by using option board:

- TA5141-RS232I (isolated)

② Chapter 5.2.1.1.3.9 "TA5141-RS232I - Option board for COMx serial communication" on page 285

RS-485 communication interface is available by using option boards:

- TA5142-RS485I (isolated)
(4) Chapter 5.2.1.1.3.10 "TA5142-RS485I - Option board for COMx serial communication" on page 289
- TA5142-RS485 (non isolated)
\# Chapter 5.2.1.1.3.11 "TA5142-RS485-Option board for COMx serial communication" on page 296


## Ethernet inter- The Ethernet interface is carried out via a RJ45 jack. face

Table 28: Pin assignment of the Ethernet interface

| Interface | Pin | Description |  |
| :---: | :---: | :---: | :---: |
| 18 | 1 | Tx+ | Transmit data + |
| T171T1 | 2 | Tx- | Transmit data - |
| $\square \square \square$ | 3 | Rx+ | Receive data + |
|  | 4 | NC | Not connected |
|  | 5 | NC | Not connected |
|  | 6 | Rx- | Receive data - |
|  | 7 | NC | Not connected |
|  | 8 | NC | Not connected |
|  | Shield | Cable shield | Functional earth |

Onboard I/Os $\quad \begin{aligned} & \text { The processor modules PM50x2 have onboard I/Os which provide several functionalities. } \\ & \text { According to the CPU type, the number or the functionality of the onboard I/Os can be different } \\ & \\ & y \text { Chapter 5.2.1.1.2 "Onboard I/Os in processor module PM50x2" on page } 196 .\end{aligned}$

## Modbus RTU connection details

The Modbus RTU protocol is implemented in the AC500 processor modules.
Modbus is a master-slave (client-server) protocol. The client sends a request to the server(s) and receives the response(s).
Available serial interfaces can work as Modbus interfaces simultaneously.
The Modbus client operating mode of an interface is set with the function block COM_MOD_MAST.

## Technical data Table 29: Description of the Modbus protocol

| Parameter | Value |
| :--- | :--- |
| Supported standard | Modbus RTU Server, Modbus RTU Client <br> ( Chapter 6.3.2.14.2 "Configuring Modbus <br> RTU on serial interface" on page 1588 |
| Number of connection points | 1 client <br> Max. 1 server with RS-232 interface <br> Max. 31 servers with RS-485 |
| Protocol | Modbus |
| Operating mode | Client/server |
| Address | Server only |
| Data transmission control | CRC16 |
| Data transmission speed | From 9,600 bits/s to 115,200 bits/s <br> 2 Chapter 6.3.2.14.2 "Configuring Modbus <br> RTU on serial interface" on page 1588 |


| Parameter | Value |
| :--- | :--- |
| Encoding | 1 start bit |
|  | 8 data bits |
|  | 1 or 2 stop bits <br> 1 parity bit $\%$ Chapter 6.3.2.14.2 "Configuring <br> Modbus RTU on serial interface" on page 1588 |
| Max. cable length for RS-485 on serial inter- <br> face option board used on the CPU. | 1.200 m at 19.200 baud |

Bus topology
Point-to-point with RS-232 or bus topology with RS-485. Modbus is a master-slave protocol.
凶 Chapter 6.8.2.9 "Communication with Modbus RTU" on page 4506

## Power supply

The processor modules PM50x2 can be connected to the 24 V DC supply voltage via a removable 3-pin spring terminal block or a 3-pin screw terminal block.

Table 30: Removable terminal block for the supply voltage 24 V DC

| 3-pin spring terminal block | 3-pin screw terminal block |
| :---: | :---: |
|  |  |

The terminal block is available as a set for AC500-eCo V3 processor modules.

| Basic CPU (PM5012) |  | Standard CPUs (PM5032, PM5052) and <br> Pro CPUs (PM5072, PM5082) |  |  |
| :--- | :--- | :--- | :--- | :---: |
| Spring type | Screw type | Spring type | Screw type |  |
| TA5211-TSPF-B | TA5211-TSCL-B | TA5212-TSPF | TA5212-TSCL |  |

Further information on power supply and onboard inputs/outputs of the terminal blocks

| Pin assignment | Pin Assignment | Pin | Label | Function | Description |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  | Terminal block inserted | 1 | $\stackrel{1}{=}$ | FE | Functional earth |
|  |  | 2 | L+ | +24 V DC | Positive pin of the power supply voltage |
|  |  | 3 | M | 0 V | Negative pin of the power supply voltage |
|  |  |  |  |  |  |

## NOTICE!

## Risk of damaging the PLC due to improper voltage levels!

- Never exceed the maximum tolerance values for process and supply voltages.
- Never fall below the minimum tolerance values for process and supply voltages.
Observe the system data and the technical data of the used module.
${ }^{\wedge}$ Chapter 5.1.1 "System data AC500-eCo" on page 159


## State LEDs and operating elements

RUN/STOP The processor modules, PM50xx series, have a RUN/STOP button. By pressing the RUN/STOP button button, the processor modules switch between RUN mode and STOP mode. By long-pressing RUN/STOP button during the processor module power on phase, the processor module will be in MOD1.

State LEDs The processor modules PM50xx indicate their states of operation via 5 LEDs located on the upper left side of the processor module.

| LED | State | Color | LED = ON | LED = OFF | LED flashing |
| :--- | :--- | :--- | :--- | :--- | :--- |
| PWR | Power supply | Green | Power supply <br> present | Power supply <br> missing | - |
| MC | card indication | Yellow | Micro memory <br> card is in the <br> socket | Micro memory <br> card is not in the <br> socket | Micro memory <br> card is in read/ <br> write state: any <br> file on card is <br> opened, means <br> activity on card |
| ERR | Error indication | Red | An error occurred | No errors or only <br> warnings <br> encountered (E4 <br> errors). <br> The LED <br> Fast flashing (4 <br> Hz) displays <br> together with the <br> RUN LED a cur- <br> rently running <br> firmware-upgrade <br> or writing data to <br> the Flash- <br> error classes 2 to the <br> EPROM. Slow <br> is configurable. <br> flashing (1 Hz) <br> alone displays <br> shutdown of |  |
| Request To |  |  |  |  |  |
| Send. Medium |  |  |  |  |  |
| flashing (2 Hz) |  |  |  |  |  |
| alone displays at |  |  |  |  |  |
| start of PLC if |  |  |  |  |  |
| reboot after |  |  |  |  |  |
| watchdog. |  |  |  |  |  |


| LED | State | Color | LED = ON | LED = OFF | LED flashing |
| :--- | :--- | :--- | :--- | :--- | :--- |
| RUN | RUN/STOP state | Green | Processor <br> module is in state <br> RUN | Processor <br> module is in state <br> STOP | Fast flashing (4 <br> Hz): <br> The processor <br> module is <br> reading/writing <br> data from/to the <br> memory card. <br> If the ERR-LED <br> is also flashing, <br> data is being <br> written to the <br> Flash-EPROM. |
|  |  |  |  |  |  |

User configurable LEDs

The AC500-eCo V3 processor module also provides 2 LEDs below the state LEDs which can be used by user and driven by an application.
The LEDs can be used into a project and controlled using special function blocks which are contained in the PM AC500 library. The POU is PmLedSet located in folder LED control.

I/O LEDs
The processor module provides up to 10 LEDs (PM5012-x-ETH), 20 LEDs (PM5032-R-ETH,

PM5052-R-ETH), or 22 LEDs (PM5032-T-ETH, PM5052-T-ETH, PM5072-T-2ETH(W), PM5082-T-2ETH) to display the states of the inputs and outputs.

| Processor <br> module | LED | State | Color | LED = ON | LED = OFF |
| :--- | :--- | :--- | :--- | :--- | :--- |
| PM5012-x-ETH | IO .. I5 | Digital input | Yellow | Input is ON | Input is OFF |
|  | O0 ... O3 | Transistor <br> output | Yellow | Output is ON | Output is OFF |


| Processor <br> module | LED | State | Color | LED = ON | LED = OFF |
| :--- | :--- | :--- | :--- | :--- | :--- |
|  | NO0 ... NO3 | Relay output | Yellow | Output is ON | Output is OFF |
| PM5032-x-ETH <br> PM5052-x-ETH | IO ... I11 | Digital input | Yellow | Input is ON | Input is OFF |
|  | O0 ... O7 | Transistor <br> output | Yellow | Output is ON | Output is OFF |
|  | NO0 ... NO5 | Relay output | Yellow | Output is ON | Output is OFF |
|  | C12, C13 | Digital configu- <br> rable input/ <br> output | Yellow | Input/Output <br> is ON | Input/Output is <br> OFF |
|  | IO ... I11 | Digital input | Yellow | Input is ON | Input is OFF |
|  | O0. O7 | Transistor <br> output | Yellow | Output is ON | Output is OFF |
|  | C12, C13 | Digital configu- <br> rable input/ <br> output | Yellow | Input/Output <br> is ON | Input/Output is <br> OFF |

## Ethernet state LEDs

Table 31: State LEDs at Ethernet connector

| LED | Color | OFF | ON | Flashing |
| :--- | :--- | :--- | :--- | :--- |
| Activity | Yellow | No activity | --- | Activity |
| Link | Green | No link | Link | --- |

## Diagnosis

The AC500 processor module can display various errors according to the error classes. The following error classes are possible. The reaction of the processor module is different for each type of error.

| Error class | Type | Description | Example |
| :--- | :--- | :--- | :--- |
| E1 <br> ERR-LED is ON | Fatal error | A safe function of the operating <br> system is no longer guaranteed. | Checksum error in the system <br> Flash or RAM error |
| ERR-LED is ON | Severe error | The operating system is func- <br> tioning without problems, but the <br> error-free processing of the user <br> program is no longer guaranteed. | Checksum error in the user <br> Flash, independent of the task <br> duration |
| E3 <br> ERR-LED is ON/OFF <br> $*$ | Light error | It depends on the application <br> if the user program should be <br> stopped by the operating system <br> or not. The user should deter- <br> mine which reaction is necessary. | Flash could not be pro- <br> grammed, I/O module has <br> failed |
| E4 <br> ERR-LED is ON/OFF <br> $*$ | Warning | Error in the periphery (e.g. I/O) <br> which may show an impact in <br> the future. The user should deter- <br> mine which reaction is necessary. | Short-circuit at an I/O module, <br> the battery is run down or not <br> inserted |
| *) The behaviour if the ERR-LED lights up at error classes E3 or E4 is configurable. |  |  |  |

Occurred errors can be displayed with the commands diagshow all in the PLC-Browser of
Automation Builder software.

## Technical data

The system data of AC500-eCo V3 apply ${ }^{\Perp}$ Chapter 5.1.1 "System data AC500-eCo" on page 159

Only additional details are therefore documented below.

| General data | Parameter | Value |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | PM5012 | PM5032 | PM5052 | PM5072 | PM5082 |
|  | Power supply | 24 V DC |  |  |  |  |
|  | Connection of power supply | Via removable 3-pin terminal |  |  |  |  |
|  | Current consumption from power supply (max.) |  |  |  |  |  |
|  | Transistor version | 200 mA | 340 mA | 400 mA | 420 mA | 420 mA |
|  | Relay version | 200 mA | 340 mA | 400 mA | - | - |
|  | Melting integral of a fuse at 24 V DC | $0.9 \mathrm{~A}^{2} \mathrm{~s}$ |  |  |  |  |
|  | Peak inrush current from 24 V DC | 65 A |  |  |  |  |
|  | Max. power dissipation within the processor module |  |  |  |  |  |
|  | Transistor version | 5.7 W | 8.1 W | 9.0 W | 9.2 W | 9.2 W |
|  | Relay version | 5.9 W | 8.3 W | 9.2 W | - | - |
|  | Processor module interfaces | RS485/RS232 (optional), Ethernet |  |  |  |  |
|  |  | - | I/O bus |  |  |  |
|  | Weight |  |  |  |  |  |
|  | Transistor version | 225 g | 253 g | 257 g | 265 g | 265 g |
|  | Relay version | 235 g | 268 g | 273 g |  |  |
|  | Mounting position | Horizontal or vertical |  |  |  |  |

Detailed data

| Parameter | Value |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  | PM5012 | PM5032 | PM5052 | PM5072 | PM5082 |
| Total maximum downloadable application size ${ }^{1)}$ | 1 MB | 5 MB | 7 MB | 9 MB | 9 MB |
| Thereof user program code / data memory dynamically allocated | 256 KB | 512 KB | 768 KB | 1 MB | 1 MB |
| Thereof user web server memory for web visualization max. | no web | 1.5 MB | 3.2 MB | 7 MB | 7 MB |
| Flash memory for user data |  |  |  |  |  |
| Remaining for all other usage (project save, infra- structure...) | 30 MB | 30 MB | 30 MB | 30 MB | 30 MB |
| Buffered (FRAM) | 8 KB | 32 KB | 32 KB | 100 KB | 100 KB |
| VAR_RETAIN persistent | 4 KB | 16 KB | 16 KB | 36 KB | 36 KB |
| \%MB data | 4 KB | 16 KB | 16 KB | 64 KB | 64 KB |
| Expandable memory | None | None | None | None | None |
| Integrated mass storage memory (FLASH) | None | None | None | None | None |
| Slot for pluggable memory card | X | X | x | X | X |


| Parameter | Value |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  | PM5012 | PM5032 | PM5052 | PM5072 | PM5082 |
| Real-time clock (RTC) | Optional with TA5131RTC | Built in |  |  |  |
| Min. retention time for RTC at room temperature (if at least powered for 8 hours) | 7 days | 20 days |  |  |  |
| Accuracy | $\pm 2$ s/day |  |  |  |  |
| Programming languages | - Instruction List (IL) <br> - Function Block Diagram (FBD) <br> - Ladder Diagram (LD) <br> - Sequential Function Chart (SFC) <br> - Structured Text (ST) <br> - Continuous Function Chart (CFC) |  |  |  |  |
| Processor type | TI ARM Cortex-A9 32-bit-RISC |  |  |  |  |
| Processor clock speed | 300 MHz |  |  |  | $\begin{array}{\|l} 600 \\ \mathrm{MHz} \end{array}$ |
| Calculation time per instructions (minimum) | PM5012 | PM5032 | PM5052 | PM5072 | PM5082 |
| Binary | 20 ns |  |  |  | 10 ns |
| Word | 50 ns |  |  |  | 10 ns |
| Floating point | 600 ns |  |  |  | 10 ns |
| Lowest cycle time usable (also f. PTO) | - | 5 | 2 | 1 | 1 |
| Using onboard EtherCAT protocol (licensed) in preparation |  |  |  |  |  |
| EtherCAT onboard No. of synchronized axis | PM5012 | PM5032 | PM5052 | PM5072 | PM5082 |
| axis per 1 ms CM typically | - | - | - | - | 2-4* |
| axis per 2 msCM typically | - | - | - | 2-4* | 4-8* |
| axis per 4 ms CM typically | - | - | - | 4-8* | 8-16 |
| Cyclic min. configurable | 10 ms | 5 ms | 2 ms | 1 ms | 1 ms |
| Time-controlled | Yes |  |  |  |  |
| Multitasking | Yes |  |  |  |  |
| Interruption | Yes |  |  |  |  |
| LEDs | Power, Error, Run, MC, MOD1, States of I/Os |  |  |  |  |
| RUN/STOP button | Yes |  |  |  |  |
| Protection of the user program by password | On request |  |  |  |  |
| Usable accessories | On request |  |  |  |  |
| Remarks: <br> * Depending an application complexity <br> 1): The values are for information only and cannot be fulfilled altogether. The available resources are limited at the end by the maximal downloadable application size for each CPU. |  |  |  |  |  |


| Data of I/Os | PM5012-x- <br> ETH | PM5032-xETH | PM5052-x- ETH | $\begin{aligned} & \hline \text { PM5072- } \\ & \text { T-2ETH(W) } \end{aligned}$ | $\begin{aligned} & \hline \text { PM5082- } \\ & \text { T-2ETH } \end{aligned}$ |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Onboard digital inputs |  |  |  |  |  |
| Channels | 6 (incl. 2 counter inputs 5 kHz and 4 interrupts) | (incl. 4 fast counter/encoder inputs ( $100 \mathrm{kHz} / 200 \mathrm{kHz}$ ), 4 inputs ( 5 kHz ), 4 standard inputs) |  |  |  |
| Signal voltage | 24 V DC type 1 |  |  |  |  |
| Onboard digital outputs |  |  |  |  |  |
| Type of digital outputs | PM5012-T- <br> ETH: <br> Transistor | PM5032-T- <br> ETH: <br> Transistor | PM5052-T- <br> ETH: <br> Transistor | PM5072- <br> T-2ETH(W): <br> Transistor | PM5082 T-2ETH: Transistor |
|  | $\begin{aligned} & \text { PM5012-R- } \\ & \text { ETH: } \\ & \text { Relay } \end{aligned}$ | PM5032-R- ETH: Relay | PM5052-R- <br> ETH: <br> Relay | - | - |
| Channels for transistor version | 4 <br> ( 5 kHz standard and PWM) | 8 <br> (incl. 4 fast outputs for standard or 4 PWM/4 PTO ( $100 \mathrm{kHz} / 200 \mathrm{kHz}$ ), 4 standard outputs ( 5 kHz )) |  |  |  |
| Channels digital input/ output configurable <br> (valid for both PLC version relais or transistor) | - | 2 <br> Relay version: <br> The DC channels can be used as $1 \mathrm{PTO} / 2 \mathrm{PWM}$ ( 100 kHz ) or standard digital inputs/outputs <br> Transistor version: <br> The DC channels can only be used as standard digital inputs/outputs |  | 2 <br> Transistor version: <br> The DC channels can only be used as standard digital inputs/outputs |  |
| Rated voltage transistor | 24 V DC |  |  |  |  |
| Nominal current per transistor channel | 0.5 A resistive |  |  |  |  |
| Channels for relay version | 4 | 6 |  | - | - |
| Rated voltage relay | $100 \text { V AC ... } 240 \text { V AC }$ <br> or $24 \text { V DC }$ |  |  | - | - |
| Nominal current per relay channel | 2 A resistive |  |  | - | - |
| Analog inputs | Optional |  |  |  |  |
| Analog outputs | Optional |  |  |  |  |
| Number of option board slots | 1 | 2 | 3 | 3 | 3 |


| Data of I/Os | $\begin{aligned} & \text { PM5012-x- } \\ & \text { ETH } \end{aligned}$ | $\begin{aligned} & \text { PM5032-x- } \\ & \text { ETH } \end{aligned}$ | $\begin{aligned} & \text { PM5052-x- } \\ & \text { ETH } \end{aligned}$ | $\begin{aligned} & \text { PM5072- } \\ & \text { T-2ETH(W) } \end{aligned}$ | $\begin{array}{\|l\|} \hline \text { PM5082- } \\ \text { T-2ETH } \end{array}$ |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Usage of option board | Each slot can be used for all type of existing option boards, same option board for serial interface or digital/analog I/O extension can be used on several slot per CPU. <br> Note: RTC option board is only for PM5012 possible. |  |  |  |  |
| KNX address switch | No |  |  | TA5130-KNXPB only on 1 slot |  |
| Real-time clock (RTC) | TA5131-RTC | No |  |  |  |
| Serial interface | TA5141-RS232I, TA5142-RS485/TA5142-RS485I |  |  |  |  |
| Digital in/out channels | TA5101-4DI, TA5105-4DOT, TA5110-2DI2DOT |  |  |  |  |
| Analog in/out channels | TA5120-2AI-UI, TA5123-2AI-RTD, TA5126-2AO-UI |  |  |  |  |
| Max. number of I/O modules on I/O bus | 0 | 10 |  |  |  |
| Digital inputs | Onboard I/O only | 128 byte | 1 KB |  |  |
| Digital outputs |  | 128 byte | 1 KB |  |  |
| Number of decentralized inputs and outputs | Depending on the fieldbus used |  |  |  |  |
| Internal interfaces |  |  |  |  |  |
| Serial COMx | Optional, use a dedicated serial interface option board (up to 1) | Optional, use a dedicated serial interface option board (up to 2) | Optional, use a dedicated serial interface option board (up to 3) |  |  |
|  | Modbus RTU Master/Slave, ASCII |  |  |  |  |
| Ethernet interface RJ45 | 1 |  |  | $2$ <br> Independent with switch functionality |  |
| Ethernet functions | Programming, TCP/IP, UDP/IP, DHCP, PING, network variables, and other listed below |  |  |  |  |
| Modbus TCP/IP client/server | $\begin{aligned} & \text { Yes } \\ & 8 / 3 \end{aligned}$ | $\begin{array}{\|l\|} \hline \text { Yes } \\ 13 / 8 \end{array}$ | $\begin{array}{\|l\|} \hline \text { Yes } \\ 20 / 10 \end{array}$ | Yes $30 / 15$ | Yes $30 / 15$ |
| SNTP client/ server | No | Yes |  |  |  |
| HTTPs and WebVisu number of connections | No | Yes 1 | $\begin{aligned} & \text { Yes } \\ & 2 \end{aligned}$ | $\begin{aligned} & \text { Yes } \\ & 4 \end{aligned}$ | $\begin{aligned} & \text { Yes } \\ & 4 \end{aligned}$ |
| FTPs <br> number of connections | No | $\begin{aligned} & \text { Yes } \\ & 1 \end{aligned}$ | $\begin{aligned} & \text { Yes } \\ & 2 \end{aligned}$ | $\begin{array}{\|l\|} \hline \text { Yes } \\ 2 \end{array}$ | $\begin{aligned} & \mathrm{Yes} \\ & 2 \end{aligned}$ |


| Data of I/Os | $\begin{aligned} & \text { PM5012-x- } \\ & \text { ETH } \end{aligned}$ | $\begin{aligned} & \text { PM5032-x- } \\ & \text { ETH } \end{aligned}$ | $\begin{aligned} & \text { PM5052-x- } \\ & \text { ETH } \end{aligned}$ | $\begin{array}{\|l\|} \hline \text { PM5072- } \\ \text { T-2ETH(W) } \end{array}$ | $\begin{aligned} & \text { PM5082- } \\ & \text { T-2ETH } \end{aligned}$ |
| :---: | :---: | :---: | :---: | :---: | :---: |
| OPC UA server number of free tags | No | $\begin{aligned} & \text { Yes } \\ & 125 \end{aligned}$ | $\begin{aligned} & \hline \text { Yes } \\ & 250 \end{aligned}$ | $\begin{aligned} & \hline \text { Yes } \\ & 1000 \end{aligned}$ | $\begin{aligned} & \text { Yes } \\ & 3000 \end{aligned}$ |
| MQTT and JSON library | No | Yes |  |  |  |
| OPC DA server | Yes |  |  |  |  |
| IEC 60870-5-104 telecontrol protocol | No |  |  | Yes <br> Substation only, 5 connections max., only 1 Ethernet supported |  |
| Licensed protocols (runtime protocol per CPU) |  |  |  |  |  |
| BACnet IP BBC $\stackrel{y}{ }$ Chapter 6.3.4.9 <br> "BACnet-BC" on page 1733 | No |  |  | Yes (max. 1000 object variables) |  |
| KNXIP ${ }^{4}$ Chapter 6.3.4.8 "KNX configurator" on page 1730 | No |  |  | Yes (max. 1000 object variables) |  |
| IEC 61850 MMS server/ goose pub/sub | No |  |  | Yes (max. 1000 data attributes) |  |
| EtherNet/IP adapter/ scanner | No | Yes max. 512 byte in / 512 byte out data for adapter |  |  |  |
| EtherCAT Master | No | Yes max. 512 byte in / 512 byte out data for adapter |  |  |  |

## Dimensions



The dimensions are in mm and in brackets in inch.

## Ordering Data

Processor modules for AC500-eCo V3 products

To enable better product availability into the production and to provide some new features, a revision of the existing AC500-eCo V3 processor module was necessary. The existing AC500eCo V3 processor module with rubric R007x will move to classic and will be replaced by compatible new AC500-eCo V3 processor module revision 1 with rubric R017x.

For example: The processor module revision 1 PM5012-T-ETH (1SAP122 600 R0172) replaces the existing processor module PM5012-T-ETH (1SAP 122600 R0072) and provides the same features or functionality of the previous ones.

Following points must be considered with the processor module revision 1:

- The processor module revision 1 (R017x) requires a new BootFW / CPUFW from V3.6.x and higher.
- It cannot be downgraded and used with lower FW versions than V3.6.0.
- The processor module revision $1(R 017 x)$ provides the same features as the processor module (R007x) existing today and is fully compatible.
- An existing application using a processor module (R007x) built with

Automation Builder < 2.6 can run on a processor module revision 1 (R017x) but the application must be upgraded to at least $A B 2.6 .0$ or higher.

What must be done using a new processor module revision 1 (R017x)?

- On a new application?
- Just use the processor module revision 1 (R017x)
- Use the latest Automation Builder Software from 2.6.0 or higher.
- On an existing application using an Automation Builder software version smaller than 2.6.0?
- To use a processor module revision 1 in an existing application (e.g., replacement of the processor module), the application must be upgraded to at least AB 2.6.0 or higher.
- If several processor module (revision 1 and revision 0 ) are used within the same project, all the processor modules used in the same application must be upgraded to the FW Version V3.6.x and higher.

Table 32: Processor modules for AC500-eCo V3

| Part no. | Description | Product life cycle phase *) |
| :---: | :---: | :---: |
| 1SAP 122600 R0072 <br> (processor module revision 0) | Basic CPU PM5012-T-ETH, AC500eCo V3 processor module, programmable logic controller 1 MB, 6DI/4DO-Transistor, Ethernet, 24 V DC, 1 option board slot | Classic <br> (replaced by processor module revision 1) |
| 1SAP 122600 R0172 <br> (processor module revision 1) | Basic CPU PM5012-T-ETH, AC500eCo V3 processor module, programmable logic controller 1 MB, 6DI/4DO-Transistor, Ethernet, 24 V DC, 1 option board slot | Active |
| 1SAP 122700 R0072 <br> (processor module revision 0) | Basic CPU PM5012-R-ETH, AC500eCo V3 processor module, programmable logic controller 1 MB , 6DI/4DO-Relay, Ethernet, 24 V DC, 1 option board slot | Classic <br> (replaced by processor module revision 1) |
| 1SAP 122700 R0172 <br> (processor module revision 1) | Basic CPU PM5012-R-ETH, AC500eCo V3 processor module, programmable logic controller 1 MB , 6DI/4DO-Relay, Ethernet, 24 V DC, 1 option board slot | Active |
| 1SAP 123400 R0072 <br> (processor module revision 0) | Standard CPU PM5032-T-ETH, AC500-eCo V3 processor module, programmable logic controller 2 MB , 12DI/8DO-Transistor/2DC, Ethernet, 24 V DC, 2 option board slots | Classic <br> (replaced by processor module revision 1) |
| 1SAP 123400 R0172 <br> (processor module revision 1) | Standard CPU PM5032-T-ETH, AC500-eCo V3 processor module, programmable logic controller 2 MB , 12DI/8DO-Transistor/2DC, Ethernet, 24 V DC, 2 option board slots | Active |
| 1SAP 123500 R0072 <br> (processor module revision 0) | Standard CPU PM5032-R-ETH, AC500-eCo V3 processor module, programmable logic controller 2 MB , 12DI/6DO-Relay/2DC, Ethernet, 24 V DC, 2 option board slots | Classic <br> (replaced by processor module revision 1) |
| 1SAP 123500 R0172 <br> (processor module revision 1) | Standard CPU PM5032-R-ETH, AC500-eCo V3 processor module, programmable logic controller 2 MB , 12DI/6DO-Relay/2DC, Ethernet, 24 V DC, 2 option board slots | Active |


| Part no. | Description | Product life cycle phase *) |
| :---: | :---: | :---: |
| $\begin{aligned} & \text { 1SAP } 124000 \text { R0072 } \\ & \text { (processor module } \\ & \text { revision 0) } \end{aligned}$ | Standard CPU PM5052-T-ETH, AC500-eCo V3 processor module, programmable logic controller 4 MB , 12DI/8DO-Transistor/2DC, Ethernet, 24 V DC, 3 option board slots | Classic <br> (replaced by processor module revision 1) |
| 1SAP 124000 R0172 <br> (processor module revision 1) | Standard CPU PM5052-T-ETH, AC500-eCo V3 processor module, programmable logic controller 4 MB, 12DI/8DO-Transistor/2DC, Ethernet, $24 \mathrm{VDC}, 3$ option board slots | Active |
| 1SAP 124100 R0072 <br> (processor module revision 0) | Standard CPU PM5052-R-ETH, AC500-eCo V3 processor module, programmable logic controller 4 MB, 12DI/6DO-Relay/2DC, Ethernet, 24 V DC, 3 option board slots | Classic <br> (replaced by processor module revision 1) |
| 1SAP 124100 R0172 <br> (processor module revision 1) | Standard CPU PM5052-R-ETH, AC500-eCo V3 processor module, programmable logic controller 4 MB, 12DI/6DO-Relay/2DC, Ethernet, 24 V DC, 3 option board slots | Active |
| $\begin{aligned} & \text { 1SAP } 124500 \text { R0073 } \\ & \text { (processor module } \\ & \text { revision 0) } \end{aligned}$ | Pro CPU PM5072-T-2ETH, AC500eCo V3 processor module, programmable logic controller 8 MB, 12DI/8DO-Transistor/2DC, 2 Ethernet, 24 V DC, 3 option board slots | Classic <br> (replaced by processor module revision 1) |
| $\begin{aligned} & \text { 1SAP } 124500 \text { R0173 } \\ & \text { (processor module } \\ & \text { revision 1) } \end{aligned}$ | Pro CPU PM5072-T-2ETH, AC500eCo V3 processor module, programmable logic controller 8 MB, 12DI/8DO-Transistor/2DC, 2 Ethernet, 24 V DC, 3 option board slots | Active |
| 1SAP 124400 R0073 <br> (processor module revision 0) | Pro CPU PM5072-T-2ETHW, AC500eCo V3 processor module, programmable logic controller 8 MB , 12DI/8DO-Transistor/2DC, 2 Ethernet, 24 V DC, 3 option board slots, wide temperature | Classic <br> (replaced by processor module revision 1) |
| $\begin{aligned} & \text { 1SAP } 124400 \text { R0173 } \\ & \text { (processor module } \\ & \text { revision 1) } \end{aligned}$ | Pro CPU PM5072-T-2ETHW, AC500eCo V3 processor module, programmable logic controller 8 MB , 12DI/8DO-Transistor/2DC, 2 Ethernet, 24 V DC, 3 option board slots, wide temperature | Active |
| 1SAP 124600 R0173 <br> (processor module revision 1) | Pro CPU PM5082-T-2ETH, AC500eCo V3 processor module, programmable logic controller $8 \mathrm{MB}, 12 \mathrm{DI} /$ 8DO-Transistor/2DC, 2 Ethernet, 24 V DC, 3 option board slots | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

Table 33: Accessories for AC500-eCo V3

| Part no. | Description | Product life cycle phase *) |
| :---: | :---: | :---: |
| Required accessories |  |  |
| 1SAP 187400 R00xx | TA52xx, terminal block set Chapter 5.2.8.1.2 "TA52xx (-x) - Terminal block sets" on page 1292 | Active |
| Optional accessories |  |  |
| 1SAP 187000 R0xxx | TA51xx, option board <br> ${ }^{\wedge}$ Chapter 5.2.1.1.3 "Option boards" on page 218 | Active |
| 1SAP 180100 R0002 | MC5102, memory card ${ }^{4}$ Chapter 5.2.8.1.1 "MC5102 Micro memory card with adapter" on page 1286 | Active |
| 1SAP 182800 R0001 | TA543, screw mounting accessory, 20 pieces per packing unit <br> „ Chapter 4.3.8.5 "Mounting a processor module on a metal plate" on page 121 | Active |
| 1SAP 187500 R0003 | TA5301-CFA, cable fixing part accessory, 20 pieces per packing unit « Chapter 4.3.8.7 "Mounting of the cable fixing accessory TA5301-CFA" on page 123 | Active |
| 1SAP 187500 R0001 | TA5300-CVR, option board slot cover, 6 pieces | Active |
| 1SAP 187600 R0001 | TA5400-SIM, input simulator for testing and training purposes, 6 switches <br> *) Chapter 5.2.8.1.4 "TA5400-SIM Input simulator" on page 1305 | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.1.1.2 Onboard I/Os in processor module PM50x2

## Functionality

The AC500-eCo V3 processor modules PM50x2 have onboard I/Os which provide several functionalities. According to the CPU type, the number or the functionality of the onboard I/Os can be different.

With the processor modules revision 1 (rubric R017x), the fast output channels provides up to 4 PTO pulse/direction with up to 200 kHz (rubric R007x, only 100 kHz ). The processor modules revision 1 are supported from $A B 2.6 .0$ and the $A B$ software automatically recognizes the used processor module.

| Parameter | Value |  |  |  |
| :---: | :---: | :---: | :---: | :---: |
|  | PM5012-T-ETH | PM5012-R-ETH | PM5032-T-ETH <br> PM5052-T-ETH <br> PM5072- <br> T-2ETH(W) <br> PM5082-T-2ETH | PM5032-R-ETH PM5052-R-ETH |
| Digital inputs | 6 |  | 12 |  |
| Functionality of digital inputs (encoder, fast counter, counter, interrupt) | 6 DI fast input 24 V DC (max. 5 kHz) <br> usable as <br> - 6 DI 24 V DC standard <br> - 2 channel 5 kHz encoder with frequency measurement or <br> - 2 channel 5 kHz encoder with frequency measurement and with touch/reset using standard DI or <br> - 2 fast counter ( 5 kHz ) <br> - 4 DI as interrupt input with 1 dedicated interrupt task and input information |  | 4 DI fast input 24 V DC (max. 200 kHz) <br> usable as <br> - 4 DI 24 V DC standard or <br> - 4 fast counter ( 100 kHz ) or <br> - $2 \mathrm{~A} / \mathrm{B}$ encoder $(200 \mathrm{kHz})$ with frequency measurement or <br> - 2 full A/B encoders 0 and 1 (200 kHz ) with frequency measurement and with touch/reset using standard highspeed ( 5 kHz ) DI <br> - 1 full A/B encoder 0 ( 200 kHz ) with frequency measurement and optional with touch/reset using 2 touch/sync inputs with A/B encoder 0 |  |
|  |  |  | 4 DI fast input 24 V DC ( 5 kHz ) usable as <br> - 4 DI 24 V DC standard or <br> - 4 DI as interrupt input with 1 dedicated interrupt task and input information <br> - 4 touch/sync inputs with $A / B$ encoder 0 or 1 |  |
|  |  |  | 4 standard DI 24 V DC |  |
| Digital outputs | 4 |  | 8 | 6 |


| Parameter | Value |  |  |  |
| :---: | :---: | :---: | :---: | :---: |
|  | PM5012-T-ETH | PM5012-R-ETH | PM5032-T-ETH <br> PM5052-T-ETH <br> PM5072- <br> T-2ETH(W) <br> PM5082-T-2ETH | PM5032-R-ETH PM5052-R-ETH |
| Functionality of digital outputs | 4 fast output DO-T <br> 24 V DC/0.5 A (max. 5 kHz ) usable as <br> - 4 DO-T 24 V DC/0.5 A or <br> - 4 PWM Note: The speed must be limited below 100 Hz . The low speed PWM can be used for heating control. <br> - 4 limit switch | $\begin{aligned} & \hline 4 \text { DO-R } \\ & 24 \text { V DC / } \\ & 120 / 240 \text { V AC 2A } \\ & \text { in } 2 \text { groups } \end{aligned}$ | 4 fast output DO-T <br> 24 V DC (100 kHz) <br> usable as <br> - 4 DO-T 24 V DC/0.5 A <br> - 4 limit/ switch outputs for encoder/ counter or <br> - 4 PWM (30 $\mathrm{kHz}, 2 \mu \mathrm{~s}$ accuracy and maximum duty 95 \%) or <br> - 2 PTO (200 kHz) CW/CCW or Pulse/Direction <br> - 4 PTO (PWM) $100 . . .200 \mathrm{kHz}$ Pulse/Direction using standard output | $\begin{aligned} & \hline 6 \text { DO-R } \\ & 24 \text { V DC / } \\ & 120 / 240 \text { V AC 2A } \\ & \text { in } 2 \text { groups } \end{aligned}$ |


| Parameter | Value |  |  |  |
| :---: | :---: | :---: | :---: | :---: |
|  | PM5012-T-ETH | PM5012-R-ETH | PM5032-T-ETH <br> PM5052-T-ETH <br> PM5072- <br> T-2ETH(W) <br> PM5082-T-2ETH | PM5032-R-ETH PM5052-R-ETH |
|  |  |  | 4 fast output DO-T <br> 24 V DC/0.5 A (5 kHz ) (max. 5 kHz ) <br> usable as <br> - 4 DO-T 24 V DC/0.5 A <br> - 4 limit/ switch outputs for encoder/ counter or <br> - 4 PWM Note: The speed must be limited below 100 Hz . The low speed PWM can be used for heating control. <br> - 4 direction outputs together with 4 high speed pulses for up to 4 PTO Pulse/Direction outputs up to 200 kHz |  |
| Digital inputs/ outputs, configurable | - | - | 2 | 2 |
| Functionality of digital inputs/ outputs, configurable | - | - | 2 DC 24 V DC <br> - 2 standard I/Os configurable | 2 DC 24 V DC <br> usable as <br> - 2 DC standard (DI 24 V DC or DO-T) or <br> - 2 PWM (30 kHz) or <br> - 1 PTO (200 kHz ) as Pulse/Direction or CW/CCW |
| LED displays | For signal states |  |  |  |


| Parameter | Value |  |  |  |
| :---: | :---: | :---: | :---: | :---: |
|  | PM5012-T-ETH | PM5012-R-ETH | PM5032-T-ETH <br> PM5052-T-ETH <br> PM5072- <br> T-2ETH(W) <br> PM5082-T-2ETH | PM5032-R-ETH PM5052-R-ETH |
| Internal power supply | Via processor module |  |  |  |
| External power supply | Via UP and ZP terminal |  |  |  |

## Connections

## General

## WARNING!

## Risk of death by electric shock!

Hazardous voltages can be present at the terminals of the module.
Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.


NOTICE!
Risk of damaging the PLC modules!
The PLC modules must not be removed while the plant is connected to a power supply.

Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove or replace a module.


## NOTICE!

Risk of damaging the PLC modules!
Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

When replacing a processor module, it is recommended to mark each wire connected to the onboard I/O terminal block before disconnecting it. This should make sure that the wires can be reconnected in the same order.

The connection is carried out by using removable 12-pin and 13-pin terminal blocks.


Table 34: Assignment of the terminals for PM5012-T-ETH:

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 1 | COM 0..5 | Input common for digital input signals I0 to I5 |
| 2 | IO | Digital input signal I0 (5 kHz) |
| 3 | I1 | Digital input signal I1 (5 kHz) |
| 4 | I2 | Digital input signal I2 (5 kHz) |
| 5 | I3 | Digital input signal I3 (5 kHz) |
| 6 | I4 | Digital input signal I4 (5 kHz) |
| 7 | O1 | Digital input signal I5 (5 kHz) |
| 8 | O2 | Digital output signal O0 (5 kHz) |
| 9 | O3 | Digital output signal O1 (5 kHz) |
| 10 | UP | Digital output signal O2 (5 kHz) |
| 11 | ZP | Digital output signal O3 (5 kHz) |
| 12 | Process supply voltage UP +24 V DC |  |
| 13 | Process supply voltage ZP 0 V DC |  |



Table 35: Assignment of the terminals for PM5012-R-ETH:

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 1 | COM $0 . .5$ | Input common for digital input signals IO to I5 |
| 2 | IO | Digital input signal I0 $(5 \mathrm{kHz})$ |
| 3 | I1 | Digital input signal I1 $(5 \mathrm{kHz})$ |
| 4 | I2 | Digital input signal I2 $(5 \mathrm{kHz})$ |
| 5 | I3 | Digital input signal I3 $(5 \mathrm{kHz})$ |
| 6 | I4 | Digital input signal I4 $(5 \mathrm{kHz})$ |
| 7 | I5 | Digital input signal I5 $(5 \mathrm{kHz})$ |
| 8 | NO1 | Normally-open relay contact of the output NO0 |
| 9 | RO..1 | Normally-open relay contact of the output NO1 |
| 10 | NO2 | Output common for signals NO0 to NO1 |
| 11 | RO3 | Normally-open relay contact of the output NO2 |
| 12 |  | Normally-open relay contact of the output NO3 |
| 13 |  | Output common for signals NO2 to NO3 |



Table 36: Assignment of the terminals for PM5032-T-ETH, PM5052-T-ETH, PM5072-T-2ETH(W) and PM5082-T-2ETH:

| Terminal | Signal | Description |
| :---: | :---: | :---: |
| 1 | COM $0 . .11$ | Input common for digital input signals 10 to 111 |
| 2 | 10 | Digital input signal 10 (max. 5 kHz ) |
| 3 | 11 | Digital input signal 11 (max. 5 kHz ) |
| 4 | 12 | Digital input signal 12 (max. 5 kHz ) |
| 5 | 13 | Digital input signal 13 (max. 5 kHz ) |
| 6 | 14 | Digital input signal 14 <br> Forward counter (max. 100 kHz), Encoder (max. 200 kHz) |
| 7 | 15 | Digital input signal I5 (100 kHz) <br> Forward counter (max. 100 kHz), Encoder (max. 200 kHz) |
| 8 | 16 | Digital input signal 16 ( 100 kHz ) <br> Forward counter (max. 100 kHz), Encoder (max. 200 kHz) |
| 9 | 17 | Digital input signal $17(100 \mathrm{kHz})$ <br> Forward counter (max. 100 kHz), Encoder (max. 200 kHz) |
| 10 | 18 | Digital input signal 18 |
| 11 | 19 | Digital input signal 19 |
| 12 | 110 | Digital input signal I10 |
| 13 | 111 | Digital input signal I11 |
| 14 | O0 | Digital output signal O0 (max. 5 kHz ) |
| 15 | O1 | Digital output signal O 1 (max. 5 kHz ) |
| 16 | O2 | Digital output signal O 2 (max. 5 kHz ) |
| 17 | O3 | Digital output signal O3 (max. 5 kHz ) |
| 18 | O4 | Digital output signal O4 PWM (max. 100 kHz), PTO (max. 200 kHz) |
| 19 | O5 | Digital output signal O5 PWM (max. 100 kHz), PTO (max. 200 kHz) |
| 20 | O6 | Digital output signal O6 PWM (max. 100 kHz), PTO (max. 200 kHz) |
| 21 | O7 | Digital output signal O7 <br> PWM (max. 100 kHz), PTO (max. 200 kHz) |
| 22 | C12 | Digital input/output signal configurable C12 |
| 23 | C13 | Digital input/output signal configurable C13 |
| 24 | UP | Process supply voltage UP +24 V DC |
| 25 | ZP | Process supply voltage ZP 0 V DC |

Table 37: Assignment of the terminals for PM5032-R-ETH and PM5052-R-ETH:


| Terminal | Signal | Description |
| :---: | :---: | :---: |
| 1 | COM $0 . .11$ | Input common for digital input signals I0 to I11 |
| 2 | 10 | Digital input signal 10 (max. 5 kHz ) |
| 3 | 11 | Digital input signal 11 (max. 5 kHz ) |
| 4 | 12 | Digital input signal 12 (max. 5 kHz ) |
| 5 | 13 | Digital input signal I3 (max. 5 kHz ) |
| 6 | 14 | Digital input signal 14 <br> Forward counter (max. 100 kHz), Encoder (max. 200 kHz) |
| 7 | 15 | Digital input signal I5 <br> Forward counter (max. 100 kHz), Encoder (max. 200 kHz) |
| 8 | 16 | Digital input signal I6 <br> Forward counter (max. 100 kHz), Encoder (max. 200 kHz) |
| 9 | 17 | Digital input signal I7 <br> Forward counter (max. 100 kHz), Encoder (max. 200 kHz ) |
| 10 | 18 | Digital input signal 18 |
| 11 | 19 | Digital input signal 19 |
| 12 | 110 | Digital input signal 110 |
| 13 | 111 | Digital input signal 111 |
| 14 | NOO | Normally-open relay contact of the output NOO |
| 15 | NO1 | Normally-open relay contact of the output NO1 |
| 16 | NO2 | Normally-open relay contact of the output NO2 |
| 17 | R0.. 2 | Output common for signals NO0 to NO2 |
| 18 | NO3 | Normally-open relay contact of the output NO3 |
| 19 | NO4 | Normally-open relay contact of the output NO4 |
| 20 | NO5 | Normally-open relay contact of the output NO5 |
| 21 | R3..5 | Output common for signals NO3 to NO5 |
| 22 | C12 | Digital input/output signal configurable C12 PWM (max. 100 kHz), PTO (max. 200 kHz) |
| 23 | C13 | Digital input/output signal configurable C13 PWM (max. 100 kHz), PTO (max. 200 kHz) |
| 24 | UP | Process supply voltage UP +24 V DC |
| 25 | ZP | Process supply voltage ZP 0 V DC |

Table 38: Internal structure of the onboard I/Os

| PM5012-T-ETH | PM5012-R-ETH | PM5032-ETH PM5052-T-ETH PM5072-T-2ETH(W) PM5082-T-2ETH | PM5032-R-ETH <br> PM5052-R-ETH |
| :---: | :---: | :---: | :---: |
|  |  |  |  |

## Connection of the digital inputs

The digital inputs can be used as source inputs or as sink inputs.

## - NOTICE!

## Risk of malfunctions in the plant!

A ground fault, e. g. caused by a damaged cable insulation, can bridge switches accidentally.

Use sink inputs when possible or make sure that, in case of error, there will be no risks to persons or plant.

Table 39: Connection of the digital inputs to the PM50x2 processor modules


## Connection of the digital transistor outputs (PM50xx-T-xETH only)



Fig. 29: Connection of digital transistor outputs and configurable digital inputs/outputs
C12 used as configurable digital input
C13 used as configurable digital transistor output

## CAUTION!

## Risk of damaging the processor module!

The outputs are not protected against short circuit and overload.

- Never short-circuit or overload the outputs.
- Never connect the outputs to other voltages.
- Use an external fuse for each output.

The configurable digital channels (C12 and C13) used as digital inputs have the same electrical characteristics as standard digital inputs.

The configurable digital channels (C12 and C13) used as digital transistor outputs have the same electrical characteristics as standard digital outputs.

## Connection of the digital relay outputs (PM50xx-R-ETH only)



Fig. 30: Connection of digital relay outputs and configurable digital inputs/outputs
C12 used as configurable digital input
C13 used as configurable digital transistor output

## WARNING!

## Risk of death by electric shock!

Hazardous voltages can be present at the terminals of the module.
Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.

## CAUTION!

## Risk of damaging the processor module!

- Never short-circuit or overload the outputs.
- Never connect inductive loads without an external suppression against voltage peaks due to inductive kickback.
- Never connect voltages > 240 V . All outputs must be fed from the same phase.
- Use an external fuse for each output.

> The configurable digital channels (C12 and C13) used as digital inputs have the same electrical characteristics as standard digital inputs.
> The configurable digital channels (C12 and C13) used as digital transistor outputs have the same electrical characteristics as standard digital outputs.

## I/O configuration

The configuration data of the onboard I/Os is stored in the processor modules PM50x2.
Chapter 6.3.2.5 "Configure the onboard I/O channel" on page 1483

## Parameterization

For information about parameterization, refer to the configuration description for onboard I/Os for processor modules PM50x2.

* "Basic CPU: PM5012-x-ETH" on page 1484

を, "Standard CPU: PM5032-x-ETH, PM5052-x-ETH Pro CPU: PM5072-T-2ETH, PM5082-T-2ETH" on page 1486

## Diagnosis

No diagnosis is generated for the onboard I/O.
There is only an error message if the configuration does not work. A log entry is generated. The Automation Builder already prevents faulty values from being entered in the configuration. If the configuration does not work, there is a system error, if e.g. faulty software or wrong versions are installed.
Otherwise there are error messages from the blocks for the individual functions.


## State LEDs

Table 40: States of the I/Os

| LED | Status | Color | LED $=$ ON | LED = OFF |
| :--- | :--- | :--- | :--- | :--- |
| I | Digital input | yellow | Input is ON | Input is OFF |
| O | Digital transistor output | yellow | Output is ON | Output is OFF |
| NO | Digital relay output | yellow | Relay contact is <br> closed | Relay contact is <br> open |
| C | Digital configurable <br> input/output | yellow | Configured input// <br> output is ON | Configured input/ <br> output is OFF |

## Technical data

## Technical data of the digital inputs

Table 41: PM5012-x-ETH

| Parameter | Value |  |
| :---: | :---: | :---: |
| Number of channels per module | 6 |  |
| Distribution of the channels into groups | 1 group of 6 channels |  |
| Galvanic isolation | Yes, per group |  |
| Connections of the channels 10 to 111 | Terminals 2 to 7 |  |
| Reference potential for the channels I0 to 111 | Terminal 1 |  |
| Indication of the input signals | 1 yellow LED per channel; the LED is ON when the input signal is high (signal 1) and the module's logic is in operation |  |
| Input type according to EN 61131-2 | Type 1 source | Type 1 sink |
| Input signal voltage | -24 V DC | +24 V DC |
| Signal 0 | -5 V .. .+3V | -3V ... +5V |
| Undefined signal | -15 V ... - 5 V | +5 V ... +15 V |
| Signal 1 | -30 V ... -15 V | +15V ... +30 V |
| Ripple with signal 0 | Within -5 V ... +3 V | Within -3 V ... +5 V |
| Ripple with signal 1 | $\begin{aligned} & \text { Within } \\ & -30 \vee \ldots-15 \vee \end{aligned}$ | $\begin{aligned} & \text { Within } \\ & +15 \mathrm{~V} \ldots+30 \mathrm{~V} \end{aligned}$ |
| Input current per channel |  |  |
| Input voltage +24 V | Typ. 4.6 mA |  |
| Input voltage +5 V | Typ. 0.8 mA |  |
| Input voltage +15 V | $>2.5 \mathrm{~mA}$ |  |
| Input voltage +30 V | $<8 \mathrm{~mA}$ |  |
| Max. permissible leakage current (at 2-wire proximity switches) | 1 mA |  |
| Input delay (0->1 or 1->0) | On request |  |
| Max. cable length *) |  |  |
| Shielded | 500 m |  |
| Unshielded | 300 m |  |

${ }^{*}$ ) For fast inputs, a shielded cable must be used and the max. cable length is 50 m .

Table 42: PM5032-x-ETH, PM5052-x-ETH, PM5072-T-2ETH $(W)$ and PM5082-T-2ETH

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 12 |
| Distribution of the channels into groups | 1 group of 12 channels |
| Galvanic isolation | Yes, per group |
| Connections of the channels I0 to I11 | Terminals 2 to 13 |
| Reference potential for the channels I0 to I11 | Terminal 1 |
| Indication of the input signals | 1 yellow LED per channel; the LED is ON <br> when the input signal is high (signal 1) <br> and the module's logic is in operation |
| Input type according to EN 61131-2 | Type 1 source | Type 1 sink | Ty |
| :--- |


| Parameter | Value |  |
| :---: | :---: | :---: |
| Input signal voltage | -24 V DC | +24 V DC |
| Signal 0 | -5 V ... +3V | -3V ... +5V |
| Undefined signal | -15 V ... - 5 V | +5 V ... +15 V |
| Signal 1 | -30 V ... -15 V | +15V ... +30 V |
| Ripple with signal 0 | Within $-5 \vee \ldots+3 \vee$ | Within $-3 \vee \ldots+5 \mathrm{~V}$ |
| Ripple with signal 1 | $\begin{aligned} & \text { Within } \\ & -30 \vee \ldots-15 \vee \end{aligned}$ | $\begin{aligned} & \text { Within } \\ & +15 \mathrm{~V} \ldots+30 \mathrm{~V} \end{aligned}$ |
| Input current per channel |  |  |
| Input voltage +24 V | Typ. 4.6 mA |  |
| Input voltage +5 V | Typ. 0.8 mA |  |
| Input voltage +15 V | $>2.5 \mathrm{~mA}$ |  |
| Input voltage +30 V | $<8 \mathrm{~mA}$ |  |
| Max. permissible leakage current (at 2-wire proximity switches) | 1 mA |  |
| Input delay (0->1 or 1->0) | On request |  |
| Max. cable length *) |  |  |
| Shielded | 500 m |  |
| Unshielded | 300 m |  |

${ }^{*}$ ) For fast inputs, a shielded cable must be used and the max. cable length is 50 m .

## Technical data of the fast counter inputs

For AC500 devices the function "fast counter" is available in S500 I/O modules as of firmware version V1.3.

For AC500-eCo V3 devices the function "fast counter" is available in onboard I/Os of PM50xx.

The AC500-eCo V3 processor modules with onboard I/Os provide some special functionality on the digital inputs or digital outputs. Fast counter, encoder inputs, interrupt inputs or PWM/PTO outputs are available depending on the device used.
The fast counter functionality can be activated within the onboard I/O configuration.
The fast counter can work in pulse/direction mode or A/B track counter mode.
\& Further information about the operating modes of the fast counter
(4) Further information about the configuration of the fast counter

As AC500-eCo V3 PLCs provide the fast counters via their onboard I/Os not only the correct power distribution to the PLC has to be made, but also the correct wiring to the signal wires. How to connect and use fast counters in AC500-eCo V3 PLCs is described in an application example.

The pulse/direction counter detects the rising edge of the counter input. It will increase or decrease the count value (depending on the direction input) at every rising edge.

The A/B track counter is used to count the signal from an encoder.

The counter can count with quad phases. In the following the behavior of the A/B track counter is described.

| Parameter | PM5012-T-ETH | PM5012-R-ETH | $\begin{aligned} & \text { PM5032-T-ETH } \\ & \text { PM5052-T-ETH } \\ & \text { PM5072- } \\ & \text { T-2ETH(W) } \\ & \text { PM5082- } \\ & \text { T-2ETH } \end{aligned}$ | PM5032-R- <br> ETH <br> PM5052-R- <br> ETH |
| :---: | :---: | :---: | :---: | :---: |
| Fast counter |  |  |  |  |
| Useable inputs | 2 | 2 | 4 | 4 |
| Fast input max. 5 kHz | $14 \ldots \mathrm{I}$. | $14 \ldots .15$ | - | - |
| Fast input, max. 100 kHz | - | - | $14 \ldots 17$ | $14 \ldots 17$ |

Technical data of the interrupt inputs

| Parameter | PM5012-T-ETH | PM5012-R-ETH | $\begin{aligned} & \text { PM5032-T-ETH } \\ & \text { PM5052-T-ETH } \\ & \text { PM5072- } \\ & \text { T-2ETH(W) } \\ & \text { PM5082- } \\ & \text { T-2ETH } \end{aligned}$ | PM5032-R- <br> ETH <br> PM5052-R- <br> ETH |
| :---: | :---: | :---: | :---: | :---: |
| Interrupt |  |  |  |  |
| Useable inputs | 4 | 4 | 4 | 4 |
| Fast input max. 5 kHz | $10 . . .13$ | $10 . . .13$ | IO ... I3 | I0 ... I3 |

## Technical data of the Touch/Reset inputs

\(\left.$$
\begin{array}{|l|l|l|l|l|}\hline \text { Parameter } & \text { PM5012-T-ETH } & \text { PM5012-R-ETH } & \begin{array}{l}\text { PM5032-T-ETH } \\
\text { PM5052-T-ETH } \\
\text { PM5072- } \\
\text { T-2ETH(W) } \\
\text { PM5082- }\end{array} & \begin{array}{l}\text { PM5032-R- } \\
\text { ETH } \\
\text { T-2ETH }\end{array}
$$ <br>
\hline PM5052-R- <br>

ETH\end{array}\right]\)| Touch/Reset |
| :--- |
| Useable inputs |
| - |


| Parameter | PM5012-T-ETH | PM5012-R-ETH | PM5032-T-ETH <br> PM5052-T-ETH <br> PM5072- <br> T-2ETH(W) <br> PM5082- <br> T-2ETH | PM5032-R- <br> ETH <br> PM5052-R- <br> ETH |
| :--- | :--- | :--- | :--- | :--- |

Technical data of the digital transistor outputs


| Parameter | Value |
| :--- | :--- |
| Short-circuit-proof / Overload-proof | No |
| Overload message | No |
| Output current limitation | No |
| Resistance to feedback against 24 V DC | No |
| Connection of 2 outputs in parallel | Not possible |
| Max. cable length *) |  |
|  | Shielded |
|  | Unshielded |

*) For PWM and PTO outputs, a shielded cable must be used.


Table 44: PM5032-T-ETH, PM5052-T-ETH, PM5072-T-2ETH $(W)$ and PM5082-T-2ETH

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 8 |
| Distribution of the channels into groups | 1 group of 8 channels |
| Galvanic isolation | Yes, per group |
| Connection of the channels O 0 to O 7 | Terminals 14 to 21 |
| Common power supply voltage | Terminals 24 (+24 V DC, signal name UP) |
| Reference potential for the channels O 0 to O 7 | Terminal 25 ( 0 V DC, negative pole of the process voltage, signal name ZP) |
| Indication of the output signals | 1 yellow LED per channel; the LED is on when the output signal is high (signal 1 ) |
| Way of operation | Non-latching type |
| Min. output voltage at signal 1 | UP-0.1 V |
| Output delay (max. at rated load) |  |
| 0 to 1 | On request |
| 1 to 0 | On request |
| Rated protection fuse (per channel) | 2 A fast |
| Output current |  |
| Rated current per channel (max.) | 0.5 A at UP 24 V DC (resistance, general use and pilot duty) |
| Rated current per group (max.) | 4 A |
| Rated current (all channels together, max.) | 4 A |
| Max. leakage current with signal 0 | 0.5 mA |
| Demagnetization when inductive loads are switched off | Must be performed externally according to driven load specification |
| Switching Frequencies |  |
| With inductive loads | On request |
| Short-circuit-proof / Overload-proof | No |
| Overload message | No |
| Output current limitation | No |
| Resistance to feedback against 24 V DC | No |
| Connection of 2 outputs in parallel | Not possible |
| Max. cable length *) |  |


| Parameter |  | Value |
| :--- | :--- | :--- |
|  | Shielded | 500 m |
|  | Unshielded | 150 m |

*) For PWM and PTO outputs, a shielded cable must be used.

## Technical data of the digital relay outputs



Table 45: PM5012-R-ETH

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 4 normally-open relay outputs |
| Distribution of the channels into groups | 2 groups for 2 channels |
| Galvanic isolation | Yes, per group |
| Connection of the channels NO 0 to NO 1 | Terminals 8 to 9 |
| Connection of the channels NO2 to NO3 | Terminals 11 to 12 |
| Reference potential R0.. 1 for the channels NOO to NO1 | Terminal 10 |
| Reference potential R2. 3 for the channels NO2 to NO3 | Terminal 13 |
| Relay output voltage |  |
| Rated value | $\begin{aligned} & 24 \mathrm{~V} \text { DC or } \\ & 100 \mathrm{~V} \mathrm{AC} \mathrm{...} 240 \mathrm{~V} \mathrm{AC} \\ & 50 \mathrm{~Hz} / 60 \mathrm{~Hz} \end{aligned}$ |
| Range | $5 \mathrm{~V} \text { DC ... } 30 \mathrm{~V} \text { DC }$ <br> or $5 \mathrm{~V} \text { AC ... } 250 \mathrm{~V} \text { AC }$ |
| Indication of the output signals | 1 yellow LED per channel; the LED is on when the output signal is high (signal 1) |
| Way of operation | Non-latching type |
| Output delay |  |
| 0 to 1 | Typ. 10 ms |
| 1 to 0 | Typ. 10 ms |
| Output current |  |
| Rated current per channel (max.) | 2.0 A (24 V DC resistance and general use, 100 V AC ... 240 V AC, resistance, general use and pilot duty) |
| Rated current per group (max.) | 6 A |
| Rated current (all channels together, max.) | 12 A |
| Demagnetization when inductive loads are switched off | External demagnetization measures must be implemented when switching inductive loads. |
| Spark suppression with inductive AC loads | Must be performed externally according to driven load specification |
| Switching frequencies |  |
| With resistive loads | Max. 1 Hz |
| With inductive loads | On request |


| Parameter |  |
| :--- | :--- |
| With lamp loads | On request |
| Short-circuit-proof / Overload-proof | No, should be provided by an external fuse <br> or circuit breaker |
| Rated protection fuse (for each channel) | 5 A fast |
| Overload message | No |
| Output current limitation | No |
| Resistance to feedback against 24 V DC | No |
| Connection of 2 outputs in parallel | Not possible |
| Lifetime of relay contacts (cycles) | 100,000 at rated load |
| Max. cable length *) |  |
| Shielded | 500 m |
| Unshielded | 150 m |

*) For PWM and PTO outputs, a shielded cable must be used.


Table 46: PM5032-R-ETH and PM5052-R-ETH

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 6 normally-open relay outputs |
| Distribution of the channels into groups | 2 groups for 3 channels |
| Galvanic isolation | Yes, per group |
| Connection of the channels NO 0 to NO 2 | Terminals 14 to 16 |
| Connection of the channels NO3 to NO5 | Terminals 18 to 20 |
| Reference potential R0.. 2 for the channels NOO to NO2 | Terminal 17 |
| Reference potential R3.. 5 for the channels NO3 to NO5 | Terminal 21 |
| Relay output voltage |  |
| Rated value | $\begin{aligned} & 24 \mathrm{~V} \text { DC or } \\ & 100 \mathrm{~V} \mathrm{AC} \mathrm{...} 240 \mathrm{~V} \mathrm{AC} \\ & 50 \mathrm{~Hz} / 60 \mathrm{~Hz} \end{aligned}$ |
| Range | $\begin{aligned} & 5 \mathrm{~V} \text { DC ... } 30 \vee \mathrm{DC} \\ & \text { or } \\ & 5 \mathrm{~V} \mathrm{AC} \ldots 250 \vee \mathrm{AC} \end{aligned}$ |
| Indication of the output signals | 1 yellow LED per channel; the LED is on when the output signal is high (signal 1) and the module is powered through the I/O bus |
| Way of operation | Non-latching type |
| Output delay |  |
| 0 to 1 | Typ. 10 ms |
| 1 to 0 | Typ. 10 ms |
| Output current |  |
| Rated current per channel (max.) | 2.0 A (24 V DC resistance and general use, 100 V AC ... 240 V AC, resistance, general use and pilot duty) |
| Rated current per group (max.) | 6 A |


| Parameter |  |
| :--- | :--- |
| Rated current (all channels together, max.) | 12 A |
| Demagnetization when inductive loads are <br> switched off | External demagnetization measures must be <br> implemented when switching inductive loads. |
| Spark suppression with inductive AC loads | Must be performed externally according to <br> driven load specification |
| Switching frequencies | Max. 1 Hz |
|  | With resistive loads |
| With inductive loads | On request |
| With lamp loads | On request |
| Short-circuit-proof / Overload-proof | No, should be provided by an external fuse <br> or circuit breaker |
| Rated protection fuse (for each channel) | 5 A fast |
| Overload message | No |
| Output current limitation | No |
| Resistance to feedback against 24 V DC | No |
| Connection of 2 outputs in parallel | Not possible |
| Lifetime of relay contacts (cycles) | 100,000 at rated load |
| Max. cable length *) | 500 m |
| Shielded | 150 m |
| Unshielded |  |

${ }^{*}$ ) For PWM and PTO outputs, a shielded cable must be used.

## Technical data of the limit switch outputs

| Parameter | PM5012-T-ETH | PM5012-R-ETH | $\begin{aligned} & \text { PM5032-T-ETH } \\ & \text { PM5052-T-ETH } \\ & \text { PM5072- } \\ & \text { T-2ETH(W) } \\ & \text { PM5082- } \\ & \text { T-2ETH } \end{aligned}$ | PM5032-R- <br> ETH <br> PM5052-R- <br> ETH |
| :---: | :---: | :---: | :---: | :---: |
| Limit switch |  |  |  |  |
| Useable outputs | 4 | - | 8 | 2 |
| Fast output max. 5 kHz | O0 ... O3 | - | O0 ... O3 | - |
| Fast output, max. 100 kHz | - | - | O4 ... O7 | C12 ... C13 |

Technical data of the PTO outputs

| Parameter | PM5012-T-ETH | PM5012-R-ETH | $\begin{aligned} & \text { PM5032-T-ETH } \\ & \text { PM5052-T-ETH } \\ & \text { PM5072- } \\ & \text { T-2ETH(W) } \\ & \text { PM5082- } \\ & \text { T-2ETH } \end{aligned}$ | PM5032-R- <br> ETH <br> PM5052-R- <br> ETH |
| :---: | :---: | :---: | :---: | :---: |
| PTO |  |  |  |  |
| Useable outputs | - | - | 4 | $1$ pair of output |
| Fast output, max. up to 200 kHz ${ }^{2}$ ) | - | - | O4 ... O7 <br> For 2 PTO 200 kHz ${ }^{1}$ ) Pulse/ Direction or CC/Ccw modes as pair of outputs | C12 ... C13 |
|  |  |  | O4 ... 07 <br> as 4 PTO up to 200 kHz Pulse outputs / Direction using fast output 5 kHz O0...O3 ${ }^{2}$ ) |  |

${ }^{1}$ ) If the load is less than 100 mA it is strongly recommended to connect an additional load resistor ( $240 \Omega / 5 \mathrm{~W}$ or $270 \Omega / 5 \mathrm{~W}$ ) to the output to improve the pulse signal.
${ }^{2}$ ) With the processor modules revision 1 (rubric R017x), the fast output provides then 200 kHz PTO Pulse output instead of 100 kHz . The processor modules revision 1 needs at least AB 2.6.0 and higher.

## Technical data of the PWM outputs

| Parameter | PM5012-T-ETH | PM5012-R-ETH | $\begin{aligned} & \text { PM5032-T-ETH } \\ & \text { PM5052-T-ETH } \\ & \text { PM5072- } \\ & \text { T-2ETH(W) } \\ & \text { PM5082- } \\ & \text { T-2ETH } \end{aligned}$ | PM5032-R- <br> ETH <br> PM5052-R- <br> ETH |
| :---: | :---: | :---: | :---: | :---: |
| PWM |  |  |  |  |
| Useable outputs | 4 | - | 8 | 2 |
| Fast output max. 5 kHz | O0 ... O3 | - | O0 ... O3 | - |
| Fast output, max. 100 kHz | - | - | O4... O7 | C12 ... C13 |

### 5.2.1.1.3 Option boards

## TA5101-4DI - Digital input option board

## Features

- 4 digital inputs 24 V DC (IO to I 3 ) in 1 group
- Module-wise galvanically isolated
- W variant available for use in extended (wide) temperature range


14 yellow LEDs to display the signal states at the digital inputs IO to I3
2 Allocation of signal name
3 5-pin terminal block for input signals

## NOTICE!

Risk of damaging the PLC modules!
Overvoltages and short circuits might damage the PLC modules.
Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.

## Intended purpose

The device is used as an optional I/O extension module for AC500-eCo V3 CPUs (PM50x2).
The inputs/outputs are group-wise galvanically isolated from each other.
All other circuitry of the module is galvanically isolated from the inputs/outputs.

## Functionality

| Parameter | Value |
| :--- | :--- |
| LED displays | For signal states |
| Internal power supply | Via internal CPU connection |
| External power supply | Not necessary |

## Connections

For a detailed description of the mounting, disassembly and connection of the module, please refer to the system assembly chapter.

The connection is carried out by using a removable 5-pin terminal block. For more information, please refer to the chapter terminal blocks for AC500-eCo V3 system. The terminal blocks are included in the module's scope of delivery and additional terminal blocks as spare parts can be ordered separately.


Fig. 31: Internal construction of the digital inputs
Table 47: Assignment of the terminals:

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 1 | COM 0..3 | Input common for signals IO to I3 |
| 2 | I0 | Input signal IO |
| 3 | I1 | Input signal I1 |
| 4 | I2 | Input signal I2 |
| 5 | I3 | Input signal I3 |

The internal power supply voltage for the module's circuitry is carried out via the connection to CPU. Thus, the current consumption from 24 V DC power supply at the terminals $\mathrm{L}+$ and M of the CPU module increases by 10 mA per TA5101-4DI.

An external power supply connection is not needed.

## WARNING!

## Removal/Insertion under power

The option boards are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug option boards with the power being ON.
Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace an option board.

Disconnecting any powered option board while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.
Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.


The digital inputs can be used as source inputs or as sink inputs.

## NOTICE!

## Risk of malfunctions in the plant!

A ground fault, e. g. caused by a damaged cable insulation, can bridge switches accidentally.
Use sink inputs when possible or make sure that, in case of error, there will be no risks to persons or plant.

Table 48: Connection of the option board for digital I/O extension TA5101-4DI


The module provides several diagnosis functions $\Leftrightarrow$ "Diagnosis" on page 221.
The meaning of the LEDs is described in the section 'State LEDs' «"State LEDs" on page 221.

## I/O configuration

The module itself does not store configuration data. It receives its parameterization data from the CPU module during power-up of the system.
Hence, replacing optional modules is possible without any re-parameterization via software.

## Parameterization

The arrangement of the parameter data is performed with Automation Builder software.

1. In the device tree, double-click the desired option board.
2. Select the "TA51xx Parameters" tab to edit the parameterization of the desired option board.

| 日月 TA5101＿4DI $\times$ |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| TA5101－4DI Parameters | Parameter <br> －Run on config fault | Type <br> Enumeration of BYTE | Value No | Default Value <br> No | Unit | Description <br> Start PLC program even on configuration fault |
| TA5101－4DI I／O Mapping |  |  |  |  |  |  |
| I／O mapping list |  |  |  |  |  |  |
| TA5101－4DI IEC Objects |  |  |  |  |  |  |
| Diagnosis |  |  |  |  |  |  |

## Diagnosis

1．In the device tree，double－click the desired option board．
2．Select the＂Diagnosis＂tab to view the diagnosis messages of the desired option board．

Table 49：Diagnosis messages

| Device | Severity | Error <br> code | Description |  |
| :--- | :--- | :--- | :--- | :--- |
|  |  | Error Message | Remedy |  |
| TA5101－4DI | 11 | 1 | Wrong or no board <br> plugged | Replace with correct func－ <br> tional board |
| TA5101－4DI | 11 | 2 | Board defective | Replace with correct func－ <br> tional board |
| TA5101－4DI | 11 | 3 | Failed to set direction | Replace with correct func－ <br> tional board |
| TA5101－4DI | 11 | 4 | Parameter wrong | Verify setting of parameter <br> ＂Run on config fault＂ |

State LEDs

| LED | State | Color | LED＝OFF | LED＝ON |
| :--- | :--- | :--- | :--- | :--- |
| Inputs I0．．．13 | Digital input | Yellow | Input is OFF | Input is ON |

## Technical data

The system data of AC500－eCo V3 apply ${ }^{\wedge}$ Chapter 5．1．1＂System data AC500－eCo＂ on page 159
Only additional details are therefore documented below．

| Parameter | Value |
| :--- | :--- |
| Galvanic isolation | Yes, between the input group and the rest of <br> the module |
|  | Isolated groups |
| Current consumption from 24 V DC power <br> supply at the L+ and M terminals of the CPU | Ca. 10 mA |
| Max. power dissipation within the module | 0.8 W |
| Weight | 15 g |
| Mounting position | Horizontal or vertical |
| Cooling | The natural convection cooling must not be hin- <br> dered by cable ducts or other parts in the con- <br> trol cabinet. |

Table 50: Technical data of the digital inputs

| Parameter | Value |  |
| :---: | :---: | :---: |
| Number of channels per module | 4 inputs 24 V DC |  |
| Distribution of the channels into groups | 1 (4 channels per group) |  |
| Connections of the channels I0 to I3 | Terminals 2 to 5 |  |
| Reference potential for the channels IO to I3 | Terminal 1 (plus or negative pole of the process supply voltage, signal name COM $0 \ldots 3$ ) |  |
| Indication of the input signals | 1 yellow LED per channel; the LED is ON when the input signal is high (signal 1). The module is powered through the CPU connection. |  |
| Monitoring point of input indicator | LED |  |
| Input type according to EN 61131-2 | Type 1 source | Type 1 sink |
| Input signal range | -24 V DC | +24 V DC |
| Signal 0 | -5V ... +3 V | -3V...+5V |
| Undefined signal | -15V... -5V | +5V ... +15V |
| Signal 1 | -30 V ... -15 V | +15 V ... +30 V |
| Input current per channel |  |  |
| Input voltage 24 V | Typ. 5 mA |  |
| Input voltage 5 V | Typ. 1 mA |  |
| Input voltage 14 V |  |  |
| Input voltage 15 V | $<3 \mathrm{~mA}$ |  |
| Input voltage 27 V |  |  |
| Input voltage 30 V | $<7 \mathrm{~mA}$ |  |
| Max. permissible leakage current (at 2-wire proximity switches) | 1 mA |  |
| Input delay (0->1 or 1->0) | Typ. 8 ms |  |
| Input data length | 1 byte |  |
| Max. cable length |  |  |
| Shielded | 500 m |  |
| Unshielded | 300 m |  |

## Dimensions



The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 187 000 R0001 | TA5101-4DI: AC500-eCo V3, digital <br> input option board, 4DI 24 V DC, <br> spring/cable front terminal 3.50 mm <br> pitch | Active |
| 1SAP 187 000 R0201 | TA5101-4DIW: AC500-eCo V3, digital <br> input option board, 4DI 24 V DC, <br> spring/cable front terminal 3.50 mm <br> pitch, wide temperature range | Active |
| Spare parts <br> 1SAP 187 400 R0012 <br> * <br> TA5220-SPF5: spring terminal block, <br> removable, 5-pin, spring front, cable <br> front, 6 pieces per packing unit Active |  |  |

## *) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

$\left.{ }^{* *}\right)$ The needed spring terminal block is always delivered with the option board.
The terminal block listed in the table is for spare part only if needed.

## TA5105-4DOT - Digital output option board

## Features

- 4 digital outputs 24 V DC ( O 0 to O 3 ) in 1 group
- Module-wise galvanically isolated
- W variant available for use in extended (wide) temperature range


14 yellow LEDs to display the signal states at the digital outputs $\mathrm{O} 0 \ldots \mathrm{O} 3$
2 Allocation of signal name
3 7-pin terminal block for output signals

## NOTICE!

Risk of damaging the PLC modules!
Overvoltages and short circuits might damage the PLC modules.
Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.

## Intended purpose

The device is used as an optional I/O extension module for AC500-eCo V3 CPUs (PM50x2).
The inputs/outputs are group-wise galvanically isolated from each other.
All other circuitry of the module is galvanically isolated from the inputs/outputs.

Functionality

| Parameter | Value |
| :--- | :--- |
| LED displays | For signal states |
| Internal power supply | Via internal CPU connection |
| External power supply | Via the terminals ZP and UP (process supply <br> voltage 24 V DC) |

## Connections

For a detailed description of the mounting, disassembly and connection of the module, please refer to the system assembly chapter.

The connection is carried out by using a removable 7-pin terminal block. For more information, please refer to the chapter terminal blocks for AC500-eCo V3 system. The terminal blocks are included in the module's scope of delivery and additional terminal blocks as spare parts can be ordered separately.


Fig. 32: Internal construction of the digital outputs
Table 51: Assignment of the terminals:

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 1 | NC | Not connected |
| 2 | O0 | Output signal O0 |
| 3 | O1 | Output signal O1 |
| 4 | O2 | Output signal O2 |
| 5 | O3 | Output signal O3 |
| 6 | UP | Process supply voltage UP +24 V DC |
| 7 | ZP | Process supply voltage ZP 0 V DC |

The internal power supply voltage for the module's circuitry is carried out via the connection to CPU. Thus, the current consumption from 24 V DC power supply at the terminals $\mathrm{L}+$ and M of the CPU module increases by 10 mA per TA5105-4DOT.

The external power supply connection is carried out via the UP (+24 V DC) and ZP ( 0 V DC) terminals.

## WARNING!

## Removal/Insertion under power

The option boards are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug option boards with the power being ON.
Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace an option board.

Disconnecting any powered option board while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.
Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

## NOTICE!

## Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.
Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.


Fig. 33: Connection of the option board for digital I/O extension TA5105-4DOT

## - NOTICE!

- Risk of malfunctions in the plant!

Only if $L+/ M$ of the CPU is available and the outputs are already configured in the AB program, the outputs will switch on as soon as the UP/ZP is available.

This must be considered in the application planning.

## NOTICE! <br> Risk of damaging the I/O module!

The outputs are not protected against short circuits and overload.

- Never short-circuit or overload the outputs.
- Never connect the outputs to other voltages.
- Use an external fuse for the outputs.

The module provides several diagnosis functions $\stackrel{y}{ } \stackrel{y}{c}$ Further information on page 227.
The meaning of the LEDs is described in the section State LEDs "State LEDs" on page 228.

## I/O configuration

The module itself does not store configuration data. It receives its parameterization data from the CPU module during power-up of the system.
Hence, replacing optional modules is possible without any re-parameterization via software.

## Parameterization

The arrangement of the parameter data is performed with Automation Builder software.

1. In the device tree, double-click the desired option board.
2. Select the "TA51xx Parameters" tab to edit the parameterization of the desired option board.


## Diagnosis



1. In the device tree, double-click the desired option board.
2. Select the "Diagnosis" tab to view the diagnosis messages of the desired option board.

Table 52: Diagnosis messages

| Device | Severity | Error <br> code | Description |  |
| :--- | :--- | :--- | :--- | :--- |
|  |  | Error Message | Remedy |  |
| TA5105-4DOT | 11 | 1 | Wrong or no board <br> plugged | Replace with correct func- <br> tional board |
| TA5105-4DOT | 11 | 2 | Board defective | Replace with correct func- <br> tional board |
| TA5105-4DOT | 11 | 3 | Failed to set direction | Replace with correct func- <br> tional board |
| TA5105-4DOT | 11 | 4 | Parameter wrong | Verify setting of parameter <br> "Run on config fault" |

State LEDs

| LED | State | Color | LED = OFF | LED = ON |
| :--- | :--- | :--- | :--- | :--- |
| Outputs O0...O3 | Digital output | Yellow | Output is OFF | Output is ON <br> (The output <br> voltage (normally <br> 24 V DC) is <br> only displayed if <br> UP/ZP and L+/M <br> (supply voltages <br> for the module) are <br> switched ON) |

## Technical data

The system data of AC500-eCo V3 apply ${ }^{\wedge}$ ) Chapter 5.1.1 "System data AC500-eCo" on page 159
Only additional details are therefore documented below.

| Parameter |  | Value |
| :--- | :--- | :--- |
| Process supply voltage UP |  |  |
|  | Connections | Terminal 6 for UP (+24 V DC) and terminal 7 <br> for ZP (0 V DC) |
|  | Rated value | 24 V DC |
|  | Current consumption via UP terminal | $5 \mathrm{~mA}+$ max. 0.5 A per output |
|  | Max. ripple | $5 \%$ |
|  | Inrush current | $0.000002 \mathrm{~A}^{2} \mathrm{~s}$ |
|  | Protection against reversed voltage | Yes |
|  | Rated protection fuse for UP | On request |
| Current consumption from 24 V DC power <br> supply at the L+/M terminals of the CPU | Ca. 10 mA |  |
| Galvanic isolation | Yes, between the output group and the rest of <br> the module |  |


| Parameter | Value |
| :--- | :--- |
| Isolated groups | $1(4$ channels per group) |
| Surge-voltage (max.) | 35 V DC for 0.5 s |
| Max. power dissipation within the module | 0.5 W |
| Weight | 16 g |
| Mounting position | Horizontal or vertical |
| Cooling | The natural convection cooling must not be hin- <br> dered by cable ducts or other parts in the con- <br> trol cabinet. |

Table 53: Technical data of the digital outputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 4 transistor outputs (24 V DC, 0.5 A max.) |
| Distribution of the channels into groups | 1 (4 channels per group) |
| Connection of the channels O 0 to O 3 | Terminals 2 to 5 |
| Common power supply voltage | Terminal 6 (positive pole of the process voltage, signal name UP) |
| Reference potential for the channels O 0 to O3 | Terminal 7 (negative pole of the process voltage, signal name ZP) |
| Indication of the output signals | 1 yellow LED per channel; the LED is on when the output signal is high (signal 1). <br> Only internal logic is powered from CPU. <br> Outputs are powered from UP/ZP terminals. |
| Way of operation | Non-latching type |
| Min. output voltage at signal 1 | UP - 0.1 V |
| Output delay (max. at rated load) |  |
| 0 to 1 | $50 \mu \mathrm{~s}$ |
| 1 to 0 | $200 \mu \mathrm{~s}$ |
| Output data length | 1 byte |
| Output current |  |
| Rated current per channel (max.) | 0.5 A at UP 24 V DC (resistance, general use and pilot duty) |
| Rated current per group (max.) | 2 A (4 channels * 0.5 A) |
| Max. leakage current with signal 0 | 0.5 mA |
| Output type | Non-protected |
| Protection type | External fuse on each channel |
| Rated protection fuse (for each channel) | On request |
| Demagnetization when inductive loads are switched off | Must be performed externally according to driven load specification |
| Switching Frequencies |  |
| With resistive load | Limited by CPU cycle time |
| With inductive load | Max. 0.5 Hz |
| With lamp load | Max. 11 Hz at max. 5 W |
| Short-circuit-proof / Overload-proof | No |


| Parameter |  | Value |
| :--- | :--- | :--- |
|  | Overload message | No |
|  | Output current limitation | No |
|  | Resistance to feedback against 24 V <br> DC | No |
| Connection of 2 outputs in parallel | Not possible |  |
| Max. cable length |  |  |
|  | Shielded | 500 m |
|  | Unshielded | 150 m |

## Dimensions



The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 187 000 R0002 | TA5105-4DOT: AC500-eCo V3, digital <br> output potion board, 4DO-T 24 V DC / <br> 0.5 A, spring/cable front terminal 3.50 <br> mm pitch | Active |
| 1SAP 187 000 R0202 | TA5105-4DOTW: AC500-eCo V3, dig- <br> ital output option board, 4DO-T 24 <br> V DC /0.5 A, spring/cable front ter- <br> minal 3.50 mm pitch, wide tempera- <br> ture range | Active |
| Spare parts |  |  |
| 1SAP 187 400 R0014 <br> *) | TA5220-SPF7: spring terminal block, <br> removable, 7-pin, spring front, cable <br> front, 6 pieces per packing unit | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.
${ }^{* *}$ ) The needed spring terminal block is always delivered with the option board.
The terminal block listed in the table is for spare part only if needed.

## TA5110-2DI2DOT - Digital I/O option board

## Features

- 2 digital inputs 24 V DC ( I 0 to I 1 ) in 1 group
- 2 digital transistor outputs 24 V DC ( O 0 to O 1 ) in 1 group
- Group-wise galvanically isolated
- W variant available for use in extended (wide) temperature range


12 yellow LEDs to display the signal states at the digital outputs O 0 to O 1
22 yellow LEDs to display the signal states at the digital inputs IO to I1
3 Allocation of signal name
4 7-pin terminal block for input/output signals

## NOTICE!

Risk of damaging the PLC modules!
Overvoltages and short circuits might damage the PLC modules.
Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.

## Intended purpose

The device is used as an optional I/O extension module for AC500-eCo V3 CPUs (PM50x2).
The inputs/outputs are group-wise galvanically isolated from each other.
All other circuitry of the module is galvanically isolated from the inputs/outputs.

Functionality

| Parameter | Value |
| :--- | :--- |
| LED displays | For signal states |
| Internal power supply | Via internal CPU connection |
| External power supply | Via the terminals ZP and UP (process supply <br> voltage 24 V DC) |

## Connections

For a detailed description of the mounting, disassembly and connection of the module, please refer to the system assembly chapter.

The connection is carried out by using a removable 7-pin terminal block. For more information, please refer to the chapter terminal blocks for AC500-eCo V3 system. The terminal blocks are included in the module's scope of delivery and additional terminal blocks as spare parts can be ordered separately.


Fig. 34: Internal construction of the digital inputs and outputs
Table 54: Assignment of the terminals:

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 1 | COM 0 $\ldots 1$ | Input common for signals IO to I1 |
| 2 | IO | Input signal I0 |
| 3 | I1 | Input signal I1 |
| 4 | O0 | Output signal O0 |
| 5 | O1 | Output signal O1 |
| 6 | UP | Process supply voltage UP +24 V DC |
| 7 | ZP | Process supply voltage ZP 0 V DC |

The internal power supply voltage for the module's circuitry is carried out via the connection to CPU. Thus, the current consumption from 24 V DC power supply at the terminals $\mathrm{L}+$ and M of the CPU module increases by 10 mA per TA5110-2DI2DOT.

The external power supply connection is carried out via the UP (+24 V DC) and ZP (0 V DC) terminals.

## WARNING!

## Removal/Insertion under power

The option boards are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug option boards with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace an option board.

Disconnecting any powered option board while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

## NOTICE!

Risk of damaging the PLC modules!
Overvoltages and short circuits might damage the PLC modules.
Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.

The digital inputs can be used as source inputs or as sink inputs.

NOTICE!
Risk of malfunctions in the plant!
A ground fault, e. g. caused by a damaged cable insulation, can bridge switches accidentally.
Use sink inputs when possible or make sure that, in case of error, there will be no risks to persons or plant.

Table 55: Connection for inputs of the option board for digital I/O extension TA5110-2DI2DOT


Fig. 35: Connection for outputs of the option board for digital I/O extension TA5110-2DI2DOT

## - NOTICE! <br> Risk of malfunctions in the plant!

Only if $L+/ M$ of the CPU is available and the outputs are already configured in the AB program, the outputs will switch on as soon as the UP/ZP is available.
This must be considered in the application planning.

## NOTICE! <br> Risk of damaging the I/O module!

The outputs are not protected against short circuits and overload.

- Never short-circuit or overload the outputs.
- Never connect the outputs to other voltages.
- Use an external fuse for the outputs.

The module provides several diagnosis functions $\stackrel{y}{ } \stackrel{y}{c}$ Further information on page 236.
The meaning of the LEDs is described in the section State LEDs "State LEDs" on page 236.

## I/O configurations

The module itself does not store configuration data. It receives its parameterization data from the CPU module during power-up of the system.
Hence, replacing optional modules is possible without any re-parameterization via software.


If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.

## Parameterization

The arrangement of the parameter data is performed with Automation Builder software.

1. In the device tree, double-click the desired option board.
2. Select the "TA51xx Parameters" tab to edit the parameterization of the desired option board.

| TA5110-2DI2DOT Parameters | Parameter <br> - Run on config fault | Type | Value | Default Value | Unit | Description |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| TA5110-2DI2DOT I/O Mapping |  | Enumeration of BYTE | No | No |  | Start PLC program even on configuration fault |
| I/O mapping list |  |  |  |  |  |  |
| TA5110-2DI2DOT IEC Objects |  |  |  |  |  |  |
| Diagnosis |  |  |  |  |  |  |
| Information |  |  |  |  |  |  |

## Diagnosis



1. In the device tree, double-click the desired option board.
2. Select the "Diagnosis" tab to view the diagnosis messages of the desired option board.

Table 56: Diagnosis messages

| Device | Severity | Error <br> code | Description |  |
| :--- | :--- | :--- | :--- | :--- |
|  |  | Error Message | Remedy |  |
| TA5110-2DI2DOT | 11 | 1 | Wrong or no board <br> plugged | Replace with correct func- <br> tional board |
| TA5110-2DI2DOT | 11 | 2 | Board defective | Replace with correct func- <br> tional board |
| TA5110-2DI2DOT | 11 | 3 | Failed to set direction | Replace with correct func- <br> tional board |
| TA5110-2DI2DOT | 11 | 4 | Parameter wrong | Verify setting of parameter <br> "Run on config fault" |


| LED | State | Color | LED = OFF | LED = ON |
| :--- | :--- | :--- | :--- | :--- |
| Inputs I0...I1 | Digital input | Yellow | Input is OFF | Input is ON |
| Outputs O0...O1 | Digital output | Yellow | Output is OFF | Output is ON |

## Technical data

The system data of AC500-eCo V3 apply ${ }^{\text {y }}$ Chapter 5.1.1 "System data AC500-eCo" on page 159

Only additional details are therefore documented below.

| Parameter |  | Value |
| :--- | :--- | :--- |
| Process supply voltage UP |  |  |
|  | Connections | Terminal 6 for UP (+24 V DC) and ter- <br> minal 7 for ZP (0 V DC) |
|  | Rated value | 24 V DC |
|  | Current consumption via UP terminal | $5 \mathrm{~mA}+$ max. 0.5 A per output |
|  | Max. ripple | $5 \%$ |


| Parameter |  | Value |
| :--- | :--- | :--- |
| Inrush current | $0.000002 \mathrm{~A}^{2} \mathrm{~s}$ |  |
|  | Protection against reversed voltage | Yes |
|  | Rated protection fuse for UP | On request |
| Current consumption from 24 V DC power supply <br> at the L+/M terminals of the CPU | Ca. 10 mA |  |
| Galvanic isolation | Yes, between the input group and the <br> output group and the rest of the module |  |
| Isolated groups | 2 groups (1 group for 2 input channels, 1 <br> group for 2 output channels) |  |
| Surge-voltage (max.) | 35 V DC for 0.5 s |  |
| Max. power dissipation within the module | 0.7 W |  |
| Weight | 15 g |  |
| Mounting position | Horizontal or vertical |  |
| Cooling | The natural convection cooling must not <br> be hindered by cable ducts or other parts <br> in the control cabinet. |  |

Table 57: Technical data of the digital inputs

| Parameter | Value |  |
| :---: | :---: | :---: |
| Number of channels per module | 2 |  |
| Distribution of the channels into groups | 1 group for 2 channels |  |
| Connections of the channels 10 to I1 | Terminals 2 to 3 |  |
| Reference potential for the channels 10 to 11 | Terminal 1 |  |
| Indication of the input signals | 1 yellow LED per channel; the LED is ON when the input signal is high (signal 1) |  |
| Monitoring point of input indicator | LED <br> It is not part of input circuit (its controlled by processor side, not process side) |  |
| Input type according to EN 61131-2 | Type 1 source | Type 1 sink |
| Input signal range | -24 V DC | +24 V DC |
| Signal 0 | -5V ... +3 V | -3 V ... +5 V |
| Undefined signal | -15V ... + 5 V | +5V ... +15 V |
| Signal 1 | -30 V ... -15V | +15 V ... +30 V |
| Ripple with signal 0 | -5V ... +3 V | -3 V ... +5 V |
| Ripple with signal 1 | -30 V ... -15 V | +15 V ... +30 V |
| Input current per channel |  |  |
| Input voltage +24 V | Typ. 5 mA |  |
| Input voltage +5 V | Typ. 1 mA |  |
| Input voltage +15 V | $<3 \mathrm{~mA}$ |  |
| Input voltage +30 V | $<7 \mathrm{~mA}$ |  |
| Max. permissible leakage current (at 2-wire proximity switches) | 1 mA |  |


| Parameter | Value |
| :--- | :--- |
| Input delay (0->1 or 1->0) | Typ. 8 ms |
| Input data length | 1 byte |
| Max. cable length |  |
|  | Shielded |
|  | Unshielded |

Table 58: Technical data of the digital outputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 2 transistor outputs (24 V DC, 0.5 A max.) |
| Distribution of the channels into groups | 1 group of 2 channels |
| Connection of the channels O 0 to O 1 | Terminals 4 to 5 |
| Reference potential for the channels O0 to O17 | Terminal 7 (negative pole of the process voltage, name ZP) |
| Common power supply voltage | Terminal 6 (positive pole of the process voltage, name UP) |
| Indication of the output signals | 1 yellow LED per channel; the LED is on when the output signal is high (signal 1) and the module is powered via the I/O bus |
| Monitoring point of output indicator | Controlled together with transistor |
| Way of operation | Non-latching type |
| Min. output voltage at signal 1 | UP - 0.1 V |
| Output delay |  |
| 0 to 1 | $50 \mu \mathrm{~s}$ |
| 1 to 0 | $200 \mu \mathrm{~s}$ |
| Output data length | 1 byte |
| Output current |  |
| Rated current per channel (max.) | 0.5 A at UP 24 V DC (resistance, general use and pilot duty) |
| Rated current per group (max.) | 1 A |
| Rated current (all channels together, max.) | 1 A |
| Max. leakage current with signal 0 | 0.5 mA |
| Output type | Non-protected |
| Protection type | External fuse on each channel |
| Rated protection fuse (for each channel) | On request |
| Demagnetization when inductive loads are switched off | Must be performed externally according to driven load specification |
| Switching Frequencies |  |
| With resistive load | Limited by CPU cycle time |
| With inductive load | Max. 0.5 Hz |
| With lamp load | Max. 11 Hz at max. 5 W |
| Short-circuit-proof / Overload-proof | No |
| Overload message | No |


| Parameter |  | Value |
| :--- | :--- | :--- |
|  | Output current limitation | No |
|  | Resistance to feedback against 24 V DC | No |
| Connection of 2 outputs in parallel | Not possible |  |
| Max. cable length |  |  |
|  |  |  |
|  | Shielded | Unshielded |

## Dimensions



The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 187 000 R0003 | TA5110-2DI2DOT: AC500-eCo V3, <br> digital I/O option board, 2DI 24 V DC, <br> 2DO-T 24 V DC / 0.5 A, spring/cable <br> front terminal 3.50 mm pitch | Active |
| 1SAP 187 000 R0203 | TA5110-2DI2DOW: AC500-eCo V3, <br> digital I/O option board, 2DI 24 V DC, <br> 2DO-T 24 V DC / 0.5 A, spring/cable <br> front terminal 3.50 mm pitch, wide <br> temperature range | Active |
| Spare parts <br> 1SAP 187 400 R0014 <br> $* *)$ <br> TA5220-SPF7: spring terminal block, <br> removable, 7-pin, spring front, cable <br> front, 6 pieces per packing unitActive |  |  |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.
${ }^{* *}$ ) The needed spring terminal block is always delivered with the option board.
The terminal block listed in the table is for spare part only if needed.

## TA5120-2AI-UI - Analog input option board

## Features

- 2 configurable analog inputs ( I 0 and I 1 ) in 1 group

Resolution 12 bits including sign

- Option board is galvanically isolated
- W variant available for use in extended (wide) temperature range


1 Allocation of terminal and signal name
22 yellow LEDs to display the signal states at the analog inputs IO and I1
3 6-pin terminal block for analog input signals and power supply (UP, ZP)
4 Input connection diagram for $U$ and $I$

## Intended purpose

The option board is used as analog input extension module for AC500-eCo V3 CPUs (PM50xx).

The analog option boards TA5120-2AI-UI and TA5126-2AO-UI can only be used from AB 2.5.2, SystemFW 3.5.0_HF-7, BootFW 3.5.1 and higher.
Using the option boards with lower versions will create a configuration error and the CPU will not start.

## Functionality

2 analog inputs, individually configurable for

- Unused (default setting)
- 0 V ... 10 V
- 0 mA ... 20 mA
- 4 mA ... 20 mA

| Parameter |  |
| :--- | :--- |
| Resolution of the analog channels |  |
|  | Value |
|  | Voltage $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ |
|  | Current $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$, |
| Current $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 12 bits |
| LED displays | 12 bits |
| Internal power supply | 12 bits |
| External power supply | 2 LEDs for signals I0 and I1 |
| Required CPU | Via the CPU PM50xx the terminals UP and ZP (process <br> voltage 24 V DC) |

## Connection

## WARNING!

## Removal/Insertion under power

The option boards are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug option boards with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace an option board.

Disconnecting any powered option board while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The option board TA5120-2AI-UI for analog input extension is plugged into an AC500-eCo V3 processor module PM50xx.
Chapter 4.3.11.1 "Mounting and demounting the option boards" on page 127
The electrical connection is made via a removable 6-pin terminal block.
he terminal block is included in the scope of delivery of the option board. Further terminal blocks can be ordered separately as spare parts.
ⓨ Chapter 5.2.8.1.2 "TA52xx (-x) - Terminal block sets" on page 1292

Table 59: Assignment of the terminals:

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 1 | I0+ | Positive analog input IO |
| 2 | I0- | Negative analog input IO |
| 3 | I1+ | Positive analog input I1 |
| 4 | I1- | Negative analog input I1 |
| 5 | UP | Process voltage UP = +24 V DC |
| 6 | ZP | Process voltage ZP = 0 V DC |



## CAUTION!

The negative terminal of the analog inputs (voltage $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ ) are connected internally and form an internal analog ground (AGND). This analog ground is connected to ZP via a PTC resistor. There is no galvanic isolation between the analog circuitry and ZP/UP. Hence, analog inputs can not be connected in series.

The internal power supply of the circuits of the module takes place via the connection to the CPU. Thus, the current consumption from 24 V DC power supply at the terminals $\mathrm{L}+$ and M of the CPU module increases by $\ll 1 \mathrm{~mA}$ per TA5120-2AI-UI.
The external power supply is connected via the terminals UP (+24 V DC) and ZP ( $0 \vee \mathrm{DC}$ ).

## NOTICE!

Risk of damaging the PLC modules!
Overvoltages and short circuits might damage the PLC modules.

Generally, analog signals must be laid in shielded cables. The cable shields must be grounded at both sides of the cables. In order to avoid unacceptable potential differences between different parts of the installation, low resistance equipotential bonding conductors must be laid.

The following figure shows the connection of the module:


Fig. 36: Internal construction of the analog inputs

## CAUTION!

By installing equipotential bonding conductors between the different parts of the system, it must be ensured that the potential difference between ZP and AGND never can exceed 1 V .

## CAUTION!

The process supply voltage must be included in the grounding concept (e. g. grounding of the negative terminal).

The option board provides several diagnosis functions.

* Chapter 5.2.1.1.3.4.8 "Diagnosis" on page 248

Connection of active-type analog sensors (Voltage) with galvanically iso- $U_{I N}$ lated power supply


By connecting the sensor's negative terminal of the output voltage to AGND, the galvanically isolated voltage source of the sensor is referred to $Z P$.

Table 60: Configurable measuring ranges for TA5120-2AI-UI

| Parameter | Value |
| :--- | :--- |
| Channel configuration | 0 V...+10 V |

In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as "not used".

Connection of active-type analog sensors (Current) with galvanically isolated power supply


| Parameter | Value |
| :--- | :--- |
| Channel configuration | $0 \mathrm{~mA} \ldots+20 \mathrm{~mA}$ |
|  | or |
|  | $+4 \mathrm{~mA} \ldots+20 \mathrm{~mA}$ |

In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as "not used".

Connection of active-type analog sensors (Voltage) with no galvanically isolated power supply


## CAUTION!

The potential difference between AGND and ZP at the module must not be greater than 1 V , not even in case of long lines.

If AGND does not get connected to ZP, the sensor supply current flows to ZP via the AGND line. This current will distort the measuring signal, as a very small current flows through the AGND line. ZP connection should be close to the sensor. The total current through the PTC should not exceed 50 mA . This measuring method is therefore only suitable for short lines and small compensation currents via the AGND line.

| Parameter | Value |
| :--- | :--- |
| Channel configuration | 0 V...+10 V |

In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as "not used".

Connection of passive-type analog sensors (current)


| Parameter | Value |
| :--- | :--- |
| Channel configuration | $0 \mathrm{~mA} \ldots+20 \mathrm{~mA}$ |
|  | or <br>  <br>  $4 \mathrm{~mA} ..+20 \mathrm{~mA}$ |

In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as "not used".

CAUTION!
If, during initialization, an analog current sensor supplies more than 25 mA for more than 1 second to an analog input, this input is switched off by the module (input protection). In such cases, it is recommended to protect the analog input by a $10-$ Volt Zener diode (in parallel to I+ and I-). But, in general, sensors with fast initialization or without current peaks higher than 25 mA are preferrable.

## I/O configuration

The option board itself does not store configuration data. It receives its parameterization from the CPU module during power-up of the system.

Hence, replacing option boards is possible without any re-parameterization via software.

| Parameter | Value |
| :--- | :--- |
| Configurability | $0 \mathrm{~V} . .10 \mathrm{~V}$ |
|  | $0 \mathrm{~mA} \ldots . .20 \mathrm{~mA}$ |
|  | $4 \mathrm{~mA} . .20 \mathrm{~mA}$ |
|  | (each input can be configured individu- <br> ally $)$ |
| Unused voltage inputs | Must be configured as "not used" |
| Unused current inputs | Must be configured as "not used" |

## Firmware update via memory card

The following steps describe the procedure for updating the firmware for the analog option boards using a memory card. Prerequisite is the previous download of the current firmware to the memory card either from the Automation Builder or as online download from ABB.
Direct from $A B B$ Software.
Click this link and on the next web page find the relevant firmware package and download it.

- Unpack this .zip archive file at any location of your hard disc
- Insert empty formatted (FAT16 / FAT32) memory card in the PC card reader
- Execute the unpacked *.exe file
- Select PC card reader as the final destination and confirm.

All directories, files and SDCARD.INI file will be automatically created on memory card and properly configured. After the process is complete, one has the prepared memory card with relevant updates.

Firmware $\quad \checkmark$ Precondition: Prepared memory card with boot project and firmware $\Leftrightarrow$ Chapter update 6.3.1.4.5.2.2 "Preparation of memory card" on page 1427.

1. Switch off the device.
2. Insert the memory card.
3. Switch on the device.
$\Rightarrow$ The alternate flashing of the RUN and the ERR LED indicates the running update process.
At the end of the update process a reboot is executed and the system firmware is started for the finishing of the update process.
If RUN LED blinks (ERR LED is off), the update was successful and the display shows done.
If ERR LED blinks (RUN LED is off), the update failed and the display shows FAIL.
The text file "SDCARD.RDY" includes the results of the different updates. If the update fails, the file contains the reasons for the abort. Based on this, further steps can be taken to fix the problem.
4. Switch off the device.
5. Remove the memory card.
6. Switch on the device.
$\Rightarrow$ The system starts with the new firmware.

## Parameterization

The arrangement of the parameter data is performed with Automation Builder software.

1. In the device tree, double-click the desired option board.
2. Select the "TA5120-2AI-UI Parameters" tab to edit the parameterization of the desired option board.

| TA5120-2AI-UI Parameters | Parameter <br> - Run on config fault | Type | Value | Default Value | Unit | Description |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| TA5120-2AI-UI I/O Mapping |  | Enumeration of BYTE | No | No |  | Start PLC program even on configuration fault |
|  | - Check supply | Enumeration of BYTE | on | On |  | Check supply |
| I/0 mapping list | - Input 0, channel configuration | Enumeration of BYTE | Not used | Not used |  | Analog input 0 - Configuration of analog input channel |
|  | - Input 0, check channel | Enumeration of BYTE | Plausib, Cut wire, Short circuit | Plausib, Cut wire, Short cirait |  | Analog input 0-Check channel |
| TA5120-2AI-UI IEC Objects | - Input 1, charnel configuration | Enumeration of SYTE | Not used | Not used |  | Analog input 1-Configuration of analog input channel |
|  | - Input 1, check channel | Enumeration of BYTE | Plausib, Cut wire, Short circuit | Plausb, Cut wire, Short dirauit |  | Analog input 1-Check channel |
| Diagnosis |  |  |  |  |  |  |
| Information |  |  |  |  |  |  |

## Diagnosis



1. In the device tree, double-click the desired option board.
2. Select the "Diagnosis" tab to view the diagnosis messages of the desired option board.

| Severity | Error | Description | Remedy |
| :---: | :---: | :---: | :---: |
| Option board error (Channel 255) |  |  |  |
| 3 | 3 | Timeout <br> Communication timeout to CPU | If the process voltage is connected properly, replace the option board. |
| 3 | 51 | Invalid slot <br> Wrong or missing option board in the appropriate slot | Check the configuration and the hardware setup. |
| 4 | 11 | Process voltage is too low | Check the process voltage. |
| 4 | 34 | Data not ready <br> Data synchronization warning | Check PLC program and synchronize. ${ }^{1}$ ) |
| 3 | 40 | SW-mismatch <br> Hardware does not match the firmware version | Replace the option board. |
| 3 | 53 | Download failed <br> Power loss during firmware update | Replace the option board. ${ }^{2}$ ) |
| 3 | 9 | DIAG_BUF_OVERFLOW Overflow in Diagnosis buffer | Diagnosis overflow usually means too many repeated warnings or errors. Please check all diagnosis in detail and take the appropriate action. A restart will clear the diagnosis. |
| 3 | 43 | DIAG_INTERNAL_ERR <br> Internal error in the option board | Replace the option board. |
| 3 | 26 | DIAG_CFG_PRM_ERR <br> Parameterization error | Check the CPU parameterization. |
| 3 | 19 | DIAG_CRC_ERR <br> Checksum error in option board | Replace the option board. |
| Channel warning |  |  |  |
| 4 | 4 | Highest level <br> Voltage measurement overflow | Check the input value. |
| 4 | 7 | Lowest level <br> Measurement underflow, or broken wire in current range 4 mA ... 20 mA | Check the input value and the wiring. |
| 4 | 48 | Overload wire break <br> Broken wire in voltage mode, overflow in current mode | Check the input value and the wiring. |

## Remarks:

| $\left.{ }^{1}\right)$ | If no other error occurs, the programs are not identical and the param- <br> eter is set to report an error in this case. |
| :--- | :--- |
| $\left.{ }^{2}\right)$ | Do not remove the power supply from the option board during the firm- <br> ware update. |

## State LEDs

|  | LED | State | Color | LED = OFF | LED = ON | LED flashes |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | $\text { Inputs } 10 \text {... }$ | Analog input | Yellow | Input is OFF or input value is too low | Input is ON <br> (brightness depends on the value of the analog signal) | -- |

## Measuring ranges

Measuring The represented resolution corresponds to 12 bits.
ranges - Input
ranges of
voltage and cur- Table 61: Measuring range: $0 \mathrm{~V} \ldots 10 \mathrm{~V}$
rent

| Range | Input [V] | Digital value |  |
| :--- | :---: | :---: | :---: |
|  |  | Decimal | Hex. |
| Wire break | $>13.1784$ | 32767 | 7FFF |
| Overflow | $>11.7564$ | 32767 | $7 F F F$ |
| Input voltage too high | 11.7564 | 32504 | 7 FF8 |
|  | $:$ | $:$ | $:$ |
|  | 10.0029 | 27656 | 6 C08 |
| Normal range | 10.0000 | 27648 | 6 C00 |
|  | $:$ | $:$ | $:$ |
|  | 0.0029 | 8 | 0008 |
| Input voltage too low | 0.0000 | 0 | 0 |
|  | -0.0029 | -8 | FFF8 |
|  | $:$ | -4864 | ED00 |
| Underflow | -1.7593 | -32768 | 8000 |

Table 62: Measuring range: $0 \mathrm{~mA} . .20 \mathrm{~mA}$

| Range | Input [mA] | Digital value |  |
| :--- | :---: | :---: | :---: |
|  |  | Decimal | Hex. |
| Overflow | $>23.5127$ | 32767 | 7FFF |
| Input current too high | 23.5127 | 32504 | 7 FF 8 |
|  | $:$ | $:$ | $:$ |
|  | 20.0058 | 27656 | 6 C08 |
| Normal range | 20.0000 | 27648 | 6 C00 |
|  | $:$ | $:$ | 0008 |


| Range | Input [mA] | Digital value |  |
| :--- | :---: | :---: | :---: |
|  |  | Decimal | Hex. |
| Input current too low | -0.0058 | -8 | FFF8 |
|  | $:$ | $:$ | $:$ |
|  | -3.5185 | -4864 | ED00 |
| Underflow | $<-3.5185$ | -32768 | 8000 |
| Negative overange | $<-23.5185$ | -32768 | 8000 |

Table 63: Measuring range: $4 \mathrm{~mA} . .20 \mathrm{~mA}$

| Range | Input [mA] | Digital value |  |
| :--- | :---: | :---: | :---: |
|  |  | Decimal | Hex. |
| Overflow | $>22.8102$ | 32767 | $7 F F F$ |
| Input current too high | 22.8102 | 32504 | 7 EF 8 |
|  | $:$ | $:$ | $:$ |
|  | 20.0046 | 27656 | 6 C08 |
| Normal range | 20.0000 | 27648 | 6 C00 |
|  | $:$ | $:$ | $:$ |
|  | 4.0046 | 8 | 0008 |
| Input current too low | 4.0000 | 0 | 0 |
|  | 3.9954 | -8 | FFF8 |
|  | $:$ | -4856 | ED08 |
|  | 1.1898 | -4864 | ED00 |
| Negative overange | $<-14.8148$ | -32768 | 8000 |

## Technical data

The system data of AC500-eCo V3 apply ${ }^{\text {² }}$ Chapter 5.1.1 "System data AC500-eCo" on page 159
Only additional details are therefore documented below.

| Parameter | Value |
| :---: | :---: |
| Process voltage |  |
| Connections | Terminal 5 for +24 V (UP) as well as terminal 6 for 0 V (ZP) |
| Rated value | 24 V DC |
| Protection against reversed voltage | Yes |
| Rated protection fuse on UP | 1 A slow |
| Galvanic isolation | Yes, per module (no isolation between channels) |
| Current consumption |  |


| Parameter |  | Value |
| :--- | :--- | :--- |
|  | From 24 V DC power supply at the termi- <br> nals UP/L+ and ZP/M of the CPU PM50xx | $<1 \mathrm{~mA}$ |
|  | From UP at normal operation | max. 20 mA |
| Inrush current from UP (at power up) | $0.005 \mathrm{~A}^{2} \mathrm{~s}$ |  |
| Max. length of analog cables, conductor cross sec- <br> tion > 0.2 $\mathrm{mm}^{2}$ | On request |  |
| Weight | 20 g |  |
| Mounting position | Horizontal or vertical with derating |  |
| Cooling | The natural convection cooling must not <br> be hindered by cable ducts or other parts <br> in the control cabinet. |  |

## Technical data of the analog inputs

## NOTICE!

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and temporary overvoltage up to 30 V DC.

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 2 |
| Distribution of channels into groups | 1 group of 2 channels |
| Connections of the channel 0 |  |
| Voltage mode |  |
| Terminal 10+ and IO- | Terminals 1 and 2 |
| Current mode |  |
| Terminal IO+ and ZP | Termnal 1 and 6 |
| Connections of the channel 1 |  |
| Voltage mode |  |
| Terminal I1+ and I1- | Terminals 3 and 4 |
| Current mode |  |
| Terminal I1+ and ZP | Terminals 3 and 6 |
| Input type | Unipolar |
| Galvanic isolation | Yes, per option board (no isolation between channels) |
| Configurability | ```0 V... }10\mathrm{ V 0 mA... }20\textrm{mA 4 mA... }20\textrm{mA (each input can be configured individu- ally)``` |
| Channel input resistance | Voltage: > $100 \mathrm{k} \Omega$ <br> Current: ca. $330 \Omega$ <br> Not used: > 100 k $\Omega$ |


| Parameter | Value |  |
| :---: | :---: | :---: |
| Time constant of the input filter | Voltage: $10 \mu \mathrm{~s}$ Current: $10 \mu \mathrm{~s}$ |  |
| Indication of the input signals | 1 LED per channel |  |
| Conversion cycle *) |  |  |
| 1 activated channel | 1 ms |  |
| 2 activated channel | 2 ms |  |
| Resolution for all configurations | 12 bits |  |
| Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range | Max. | $\pm 0.3 \%$ at $+25^{\circ} \mathrm{C}$ |
|  | Max. | $\pm 0.5 \%$ over full temperature range |
| Temperature coefficient <br> (is related to the max error at $+25^{\circ} \mathrm{C}$ and max error in the full range) | $\pm 0.005$ \%/K |  |
| Temporary deviation during EMC disturbance | Max. | $\pm 1$ \% |
| Mapping between input signal and digital value | Input ranges of voltage and current |  |
| Unused voltage inputs | Must be configured as "not used" |  |
| Unused current inputs | Must be configured as "not used" |  |
| Overvoltage protection | Yes |  |

${ }^{*}$ ) The value is the sampling time on the option board. The complete conversion cycle time is also related to the CPU cycle time.

## Dimensions



The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 187 100 R0001 | TA5120-2AI-UI: AC500-eCo V3, <br> analog input option board, 2AI U/I, 12 <br> bits, spring/cable front terminal 3.50 <br> mm pitch | Active |
| 1SAP 187 100 R0201 | TA5120-2AI-UIW: AC500-eCo V3, <br> analog input option board, 2AI U/I, 12 <br> bits, spring/cable front terminal 3.50 <br> mm pitch, wide temperature range | Active |
| Spare parts <br> 1SAP 187 400 R0013 <br> $* *$ <br> TA5220-SPF6: spring terminal block, <br> removable, 6-pin, spring front, cable <br> front, 6 pieces per packing unit Active |  |  |


#### Abstract

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.


$\left.{ }^{* *}\right)$ The needed spring terminal block is always delivered with the option board.
The terminal block listed in the table is for spare part only if needed.

## TA5123-2AI-RTD - Analog input option board

## Features

- 2 configurable analog inputs ( 10 and I1) in 1 group

Resolution 16 bits including sign

- Option board is galvanically isolated
- W variant available for use in extended (wide) temperature range


1 Allocation of signal name
22 yellow LEDs to display the signal states at the analog inputs 10 and I1
3 8-pin terminal block for analog input signals and power supply (UP, ZP)

## Intended purpose

The option board is used as analog input extension module for AC500-eCo V3 CPUs (PM50xx).


The analog option board TA5123-2AI-RTD can only be used from AB 2.6.0, SystemFW 3.6.0, BootFW 3.6 and higher.
Using the option board with lower versions will create a configuration error and the CPU will not start.

## Functionality

2 analog inputs, individually configurable for

- Unused (default setting)
- Pt100, $-50^{\circ} \mathrm{C} . . .+400^{\circ} \mathrm{C}$ (2-wire)
- Pt100, $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ (3-wire)
- Pt100, $-50^{\circ} \mathrm{C} \ldots+130^{\circ} \mathrm{C}$ (2-wire)
- Pt100, $-50^{\circ} \mathrm{C} \ldots+130^{\circ} \mathrm{C}$ (3-wire)
- Pt1000, $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ (2-wire)
- Pt1000, $-50^{\circ} \mathrm{C} . . .+400^{\circ} \mathrm{C}$ (3-wire)
- Ni1000, $-50^{\circ} \mathrm{C} . . .+150^{\circ} \mathrm{C}$ (2-wire)
- Ni1000, $-50^{\circ} \mathrm{C} . . .+150^{\circ} \mathrm{C}$ (3-wire)
- NTC (2-wire)

| Parameter | Value |
| :--- | :--- |
| Resolution of the analog channels |  |
|  | Temperature |
| LED displays | $0.1^{\circ} \mathrm{C}$ |
| Internal power supply | 2 LEDs for signals I0 and I1 |
| External power supply | Via the CPU PM50xx <br> voltage terminals UP and ZP (process <br> Required CPU |

## Connections

## WARNING!

## Removal/Insertion under power

The option boards are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug option boards with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace an option board.

Disconnecting any powered option board while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.
Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The option board TA5123-2AI-RTD for analog input extension is plugged into an AC500-eCo V3 CPU PM50x2.

Insert the module and press it until it locks into place.

## A detailed description of the assembly and disassembly of the module can be found in the chapter Mounting and demounting option boards.

② Chapter 4.3.11.1 "Mounting and demounting the option boards" on page 127

The electrical connection is made via a removable 8-pin terminal block. For more information, please refer to the chapter TA52xx(-x) - Terminal block sets.

The terminal block is included in the scope of delivery of the option board.
Further terminal blocks can be ordered separately as spare parts.

Table 64: Assignment of the terminals:

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 1 | IOA | Connection for 3-wire measurement of the analog input <br> IO |
| 2 | IO+ | Positive analog input IO |
| 3 | I0- | Negative analog input I0 |
| 4 | I1A | Connection for 3-wire measurement of the analog input <br> I1 |
| 5 | I1+ | Positive analog input I1 |
| 6 | I1- | Negative analog input I1 |
| 7 | UP | Process voltage UP $=+24 \mathrm{VDC}$ |
| 8 | ZP | Process voltage ZP $=0 \mathrm{~V}$ |

## CAUTION!

The negative terminal of the analog inputs are connected internally and form an internal analog ground (AGND). This analog ground is connected to ZP via a PTC resistor. There is no galvanic isolation between the analog circuitry and ZP/UP. Hence, analog inputs can not be connected in series.

The internal power supply of the circuits of the module takes place via the connection to the CPU. Thus, the current consumption from 24 VDC power supply at the terminals $\mathrm{L}+$ and M of the CPU module increases by << 1 mA per TA5123-2AI-RTD.
The external power supply is connected via the UP (+24 VDC) and the ZP ( 0 VDC ) terminals.

## NOTICE!

Risk of damaging the PLC modules!
Overvoltages and short circuits might damage the PLC modules.

Generally, analog signals must be laid in shielded cables. The cable shields must be grounded at both sides of the cables. In order to avoid unacceptable potential differences between different parts of the installation, low resistance equipotential bonding conductors must be laid.

The following block diagram shows the internal construction of the analog inputs:


## CAUTION!

By installing equipotential bonding conductors between the different parts of the system, it must be ensured that the potential difference between ZP and AGND never can exceed 1 V .

## CAUTION!

The process supply voltage must be included in the grounding concept (e. g. grounding of the negative terminal).

The module provides several diagnosis functions, see section Diagnosis.
The meaning of the LEDs is described in the section State LEDs.

## Connection of resistance thermometers in 2wire configuration



Fig. 37: Connection example
The following measuring ranges can be configured $\Longleftrightarrow$ Chapter 5.2.4.4.2.3.7 "Parameterization" on page 749:

| Parameter | Resistance thermometers | Value |
| :--- | :--- | :--- |
| Channel configuration | Pt100 | $-50{ }^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ |
|  |  | $-50^{\circ} \mathrm{C} \ldots+130{ }^{\circ} \mathrm{C}$ |
|  | Pt1000 | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ |
|  | Ni1000 | $-50^{\circ} \mathrm{C} \ldots+150{ }^{\circ} \mathrm{C}$ |
|  | NTC | $803 \Omega \ldots 100 \mathrm{k} \Omega$ |

In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as "not used".

## Connection of resistance thermometers in 3wire configuration

Fig. 38: Connection example

| Parameter | Resistance thermometers | Value |
| :--- | :--- | :--- |
| Channel configuration | Pt100 | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ |
|  |  | $-50^{\circ} \mathrm{C} \ldots+130^{\circ} \mathrm{C}$ |
|  | Pt1000 | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ |
|  | Ni1000 | $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ |

The module linearizes the resistance thermometer characteristics. In order to keep measuring errors as small as possible, it is necessary by all means to have all the involved conductors in the same cable. All the conductors must have the same cross section.
In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as "not used".

## I/O configuration

The option board itself does not store configuration data. It receives its parameterization data from the CPU module during power-up of the system.

Hence, replacing option boards is possible without any re-parameterization via software.
ⓨ Further information to the option board.

## Firmware update via memory card

The following steps describe the procedure for updating the firmware for the analog option boards using a memory card. Prerequisite is the previous download of the current firmware to the memory card either from the Automation Builder or as online download from ABB.

Direct from ABB Software.
Click this link and on the next web page find the relevant firmware package and download it.

- Unpack this .zip archive file at any location of your hard disc
- Insert empty formatted (FAT16 / FAT32) memory card in the PC card reader
- Execute the unpacked *.exe file
- Select PC card reader as the final destination and confirm.

All directories, files and SDCARD.INI file will be automatically created on memory card and properly configured. After the process is complete, one has the prepared memory card with relevant updates.

| Firmware |  |
| :--- | :--- |
| update | $\checkmark$ Precondition: Prepared memory card with boot project and firmware ${ }^{4} \Rightarrow$ Chapter |
| 6.3.1.4.5.2.2 "Preparation of memory card" on page 1427. |  |

1. Switch off the device.
2. Insert the memory card.
3. Switch on the device.
$\Rightarrow$ The alternate flashing of the RUN and the ERR LED indicates the running update process.
At the end of the update process a reboot is executed and the system firmware is started for the finishing of the update process.

If RUN LED blinks (ERR LED is off), the update was successful and the display shows done.

If ERR LED blinks (RUN LED is off), the update failed and the display shows FAIL.
The text file "SDCARD.RDY" includes the results of the different updates. If the update fails, the file contains the reasons for the abort. Based on this, further steps can be taken to fix the problem.
4. Switch off the device.
5. Remove the memory card.
6. Switch on the device.
$\Rightarrow$ The system starts with the new firmware.

## Parameterization

The arrangement of the parameter data is performed with Automation Builder software.

1. In the device tree, double-click the desired option board.
2. Select the "TA5123-2AI-RTD Parameters" tab to edit the parameterization of the desired option board.

3. The two inputs are set to "Not used" by default.

Select your 2 individually configurable analog inputs.
2. If you select Pt100 (2-wire), PT100 (3-wire), Pt100 (2-wire), PT100 (3-wire), NI1000 (2wire) or NI1000 (3-wire) by double-clicking, parameterization is completed.
If you select NTC (2-wire) by double-clicking, you still have to make further entries.
3. The following steps are only required for the NTC (2-wire) selection.

If you have selected NTC (2-wire) on input 0 (1) , a folder Input 0 (1) - R, T, B configuration appears.

## Data sheet of used NTC

Take the following values from the data sheet of the NTC used.

- Resistance at temperature $T[\Omega]$
- Temperature at resistance $R\left[{ }^{\circ} \mathrm{C}\right.$ ] (the reference temperature is usually $25^{\circ} \mathrm{C}$ )
- B [K] - The shape of the curve, represents the relationship between the resistance and the temperature of a particular thermistor.

4. Enter these values in the column "Value" in the respective cells R, T and B. Observe the correct units in the note.
When you have entered the 3 values with double-click, the parameterization is completed.
② Table 68 "Measuring range: NTC" on page 265

## Diagnosis

| TA5123-2AI-RTD Parameters | Start refresh Stop refresh Acknowledge Selected Alarms |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| TA5123-2AI-RTD I/O Mapping | Type | Device | Timestamp | Severity | Error Code | Description | Additional Data |
| I/O mapping list |  |  |  |  |  |  |  |
| TA5123-2AI-RTD IEC Objects |  |  |  |  |  |  |  |
| Diagnosis |  |  |  |  |  |  |  |
| Information |  |  |  |  |  |  |  |

1. In the device tree, double-click the desired option board.
2. Select the "Diagnosis" tab to view the diagnosis messages of the desired option board.

The following table shows the diagnosis messages.

| Severity | Error | Description | Remedy |
| :---: | :---: | :---: | :---: |
| Option board error (Channel 255) |  |  |  |
| 3 | 3 | Timeout <br> Communication timeout to CPU | If the process voltage is connected properly, replace the option board. |
| 3 | 51 | Invalid slot <br> Wrong or missing option board in the appropriate slot | Check the configuration and the hardware setup. |
| 4 | 11 | Process voltage is too low | Check the process voltage. |
| 4 | 34 | Data not ready <br> Data synchronization warning | Check PLC program and synchronize. ${ }^{1}$ ) |
| 3 | 40 | SW-mismatch <br> Hardware does not match the firmware version | Replace the option board. |
| 3 | 53 | Download failed <br> Power loss during firmware update | Replace the option board. ${ }^{2}$ ) |
| 3 | 9 | DIAG_BUF_OVERFLOW Overflow in Diagnosis buffer | Diagnosis overflow usually means too many repeated warnings or errors. Please check all diagnosis in detail and take the appropriate action. A restart will clear the diagnosis. |
| 3 | 43 | DIAG_INTERNAL_ERR <br> Internal error in the option board | Replace the option board. |
| 3 | 26 | DIAG_CFG_PRM_ERR <br> Parameterization error | Check the CPU parameterization. |
| 3 | 19 | DIAG_CRC_ERR <br> Checksum error in option board | Replace the option board. |
| Channel warning |  |  |  |
| 4 | 5 | DIAG_HIGH_LEVEL NTC value too high | Check the NTC input and configuration. |
| 4 | 6 | DIAG_LOW_LEVEL NTC value too low | Check the input value and the wiring. |
| 4 | 7 | Lowest Level <br> Measurement underflow | Check the input value. |
| 4 | 47 | DIAG_WIRE_SHORTCUT <br> Short circuit at analog input | Check the wiring. |
| 4 | 48 | Overload Cut Wire Overflow or broken wire | Check the NTC input and configuration. |

Remarks:

| $\left.{ }^{1}\right)$ | If no other error occurs, the programs are not identical and the param- <br> eter is set to report an error in this case. |
| :--- | :--- |
| $\left.{ }^{2}\right)$ | Do not remove the power supply from the option board during the firm- <br> ware update. |

State LEDs

|  | LED | State | Color | LED = OFF | LED = ON | LED flashes |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | $\begin{array}{\|l\|l\|} \hline \text { Inputs } \\ \text { IO...I1 } \end{array}$ | Analog input | Yellow | Input is OFF or input value is too low | Input is ON <br> (brightness depends on the value of the analog signal) | -- |

## Measuring ranges

Measuring ranges - resistance temperature detector

Table 65: Measuring range: Pt100 /Pt1000-50 ${ }^{\circ} \mathrm{C} \ldots+400{ }^{\circ} \mathrm{C}$

| Range/ <br> LED behavior | Input | Digital value |  |
| :---: | :---: | :---: | :---: |
|  |  | Decimal | Hex. |
| Wire break/LED OFF | >> $450.0{ }^{\circ} \mathrm{C}$ | 32767 | 7FFF |
| Overflow/LED OFF | $>450.0{ }^{\circ} \mathrm{C}$ | 32767 | 7FFF |
| Temperature too high/LED 100 \% | $\begin{gathered} 450.0^{\circ} \mathrm{C} \\ : \\ 400.1^{\circ} \mathrm{C} \end{gathered}$ | $\begin{gathered} 4500 \\ : \\ 4001 \end{gathered}$ | $\begin{gathered} 1194 \\ : \\ \text { 0FA1 } \end{gathered}$ |
| Normal range/LED 0 \% ... 100 \% | $\begin{gathered} 400^{\circ} \mathrm{C} \\ : \\ 0.1^{\circ} \mathrm{C} \end{gathered}$ | $\begin{gathered} 4000 \\ : \\ 1 \end{gathered}$ | $\begin{gathered} \text { 0FAO } \\ : \\ 0001 \end{gathered}$ |
|  | 0.0000 | 0 | 0 |
|  | $\begin{gathered} -0.1^{\circ} \mathrm{C} \\ : \\ -50.0^{\circ} \mathrm{C} \end{gathered}$ | $\begin{gathered} \hline-1 \\ : \\ -500 \end{gathered}$ | $\begin{gathered} \text { FFFF } \\ : \\ \text { FE0C } \end{gathered}$ |
| Temperature too low/LED OFF | $\begin{gathered} -50.1^{\circ} \mathrm{C} \\ : \\ -60.0^{\circ} \mathrm{C} \end{gathered}$ | $\begin{gathered} -501 \\ : \\ -600 \end{gathered}$ | $\begin{gathered} \text { FEOB } \\ : \\ \text { FDA8 } \end{gathered}$ |
| Underflow/LED OFF | $<-60.0^{\circ} \mathrm{C}$ | -32768 | 8000 |
| Short circuit/LED OFF | $\ll-60.0{ }^{\circ} \mathrm{C}$ | -32768 | 8000 |

Table 66: Measuring range: Pt100/Pt1000-50 ${ }^{\circ} \mathrm{C} \ldots+130{ }^{\circ} \mathrm{C}$

| Range/ <br> LED behavior | Input | Digital value |  |
| :--- | :---: | :---: | :---: |
|  |  | Decimal | Hex. |
| Wire break/LED OFF | $\gg 140.0^{\circ} \mathrm{C}$ | 32767 | $7 F F F$ |
| Overflow/LED OFF | $>140.0^{\circ} \mathrm{C}$ | 32767 | 7 FFF |
| Temperature too <br> high/LED $100 \%$ | $140.0^{\circ} \mathrm{C}$ | 1400 | 0578 |
|  | $:$ | $:$ | $:$ |
|  | $130.1^{\circ} \mathrm{C}$ | 1301 | 0515 |


| Range/ <br> LED behavior | Input | Digital value |  |
| :---: | :---: | :---: | :---: |
|  |  | Decimal | Hex. |
| Normal range/LED 0 \% ... 100 \% | $130.0{ }^{\circ} \mathrm{C}$ | 1300 | 0514 |
|  | $0.1{ }^{\circ} \mathrm{C}$ | 1 | 0001 |
|  | 0.0000 | 0 | 0 |
|  | $-0.1{ }^{\circ} \mathrm{C}$ | -1 | FFFF |
|  |  |  |  |
|  | $-50.0^{\circ} \mathrm{C}$ | -500 | FEOC |
| Temperature too low/LED OFF | $-50.1{ }^{\circ} \mathrm{C}$ | -501 | FEOB |
|  |  |  | : |
|  | $-60.0{ }^{\circ} \mathrm{C}$ | -600 | FDA8 |
| Underflow/LED OFF | $<-60.0{ }^{\circ} \mathrm{C}$ | -32768 | 8000 |
| Short circuit/LED OFF | <- $60.0{ }^{\circ} \mathrm{C}$ | -32768 | 8000 |

Table 67: Measuring range: Ni1000 $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$

| Range/ <br> LED behavior | Input | Digital value |  |
| :---: | :---: | :---: | :---: |
|  |  | Decimal | Hex. |
| Wire break/LED OFF | >> $160.0{ }^{\circ} \mathrm{C}$ | 32767 | 7FFF |
| Overflow/LED OFF | $>160.0{ }^{\circ} \mathrm{C}$ | 32767 | 7FFF |
| Temperature too high/LED 100 \% | $160.0^{\circ} \mathrm{C}$ <br> $150.1^{\circ} \mathrm{C}$ | $\begin{gathered} 1600 \\ : \\ 1501 \\ \hline \end{gathered}$ | $\begin{gathered} 0640 \\ : \\ \text { 05DD } \end{gathered}$ |
| Normal range/LED 0 \% ... 100 \% | $\begin{gathered} 150.0^{\circ} \mathrm{C} \\ : \\ 0.1^{\circ} \mathrm{C} \\ \hline \end{gathered}$ | $\begin{gathered} 1500 \\ : \\ 1 \\ \hline \end{gathered}$ | $\begin{gathered} \hline \text { 05DC } \\ : \\ 0001 \end{gathered}$ |
|  | 0.0000 | 0 | 0 |
|  | $\begin{gathered} -0.1^{\circ} \mathrm{C} \\ : \\ -50.0^{\circ} \mathrm{C} \end{gathered}$ | $\begin{gathered} \hline-1 \\ : \\ -500 \end{gathered}$ | FFFF <br> FEOC |
| Temperature too low/LED OFF | $-50.1^{\circ} \mathrm{C}$ <br> $-60.0^{\circ} \mathrm{C}$ | $\begin{gathered} \hline-501 \\ : \\ -600 \\ \hline \end{gathered}$ | $\begin{gathered} \text { FEOB } \\ : \\ \text { FDA8 } \\ \hline \end{gathered}$ |
| Underflow/LED OFF | $<-60.0{ }^{\circ} \mathrm{C}$ | -32768 | 8000 |
| Short circuit/LED OFF | <<-60.0 ${ }^{\circ} \mathrm{C}$ | -32768 | 8000 |

## Variable measuring range

With the NTC, a resistance is measured. Depending on the values $R, T$ and $B$ a temperature value can be calculated, from which the digital values (dec. and hex.) are derived.
The following table shows an example with the default value:

- $\quad R=10000 \Omega$
- $\quad T=25^{\circ} \mathrm{C}$
- $\quad B=3435 K$

Table 68: Measuring range: NTC
$\begin{array}{|l|c|c|c|c|}\hline \text { Range/ } \\ \text { LED behavior }\end{array}$ Calculated value $\left.\begin{array}{c}\text { Measured value } \\ {[\Omega]}\end{array}\right)$

## Values written in italics

Values written in italics are variable and depend on the parameters $R, T$ and $B$.
Here, the default values are used (R: $10000 \Omega, \mathrm{~T}: 25^{\circ} \mathrm{C}, \mathrm{B}: 3435 \mathrm{~K}$ )

## Technical data

The system data of AC500-eCo V3 apply ${ }^{4}$ ) Chapter 5.1.1 "System data AC500-eCo" on page 159

Only additional details are therefore documented below.

| Parameter | Value |
| :---: | :---: |
| Process voltage |  |
| Connections | Terminals 7 for +24 V (UP) and 8 for 0 V (ZP) |
| Rated value | 24 V DC |
| Protection against reversed voltage | Yes |
| Rated protection fuse on UP | 1 A slow |
| Galvanic isolation | Yes, per module (no isolation between channels) |
| Current consumption |  |
| From 24 V DC power supply at the terminals UP/L+ and ZP/M of the CPU PM50x2 | < 1 mA |
| From UP at normal operation | max. 20 mA |
| Inrush current from UP (at power up) | $0.005 \mathrm{~A}^{2} \mathrm{~s}$ |
| Max. length of analog cables, conductor cross section $\geq 0.2 \mathrm{~mm}^{2}$ | On request |
| Weight | 20 g |
| Mounting position | Horizontal or vertical |
| Cooling | The natural convection cooling must not be hindered by cable ducts or other parts in the control cabinet. |

Technical data of the analog inputs


| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 2 |
| Distribution of channels into groups | 1 group of 2 channels |
| Input type | Unipolar with Pt100, Pt1000, Ni1000 <br> and NTC |
| Galvanic isolation | Yes, per module (no isolation between <br> channels) |


| Parameter | Value |
| :---: | :---: |
| Configurability | $\begin{aligned} & \text { Pt100 (2-wire) }-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C} \\ & \text { Pt100 (3-wire) }-50^{\circ} \mathrm{C} . .+400^{\circ} \mathrm{C} \\ & \text { Pt100 (2-wire) }-50^{\circ} \mathrm{C} \ldots+130^{\circ} \mathrm{C} \\ & \text { Pt100 (3-wire) }-50^{\circ} \mathrm{C} . .+130^{\circ} \mathrm{C} \\ & \text { Pt1000 (2-wire) }-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C} \\ & \text { Pt1000 (3-wire) }-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C} \\ & \text { Ni1000 (2-wire) }-50^{\circ} \mathrm{C} . .+150^{\circ} \mathrm{C} \\ & \text { Ni1000 (3-wire) }-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C} \\ & \text { NTC (2-wire) } \\ & \text { (each input can be configured individu- } \\ & \text { ally) } \end{aligned}$ |
| Indication of the input signals | 1 LED per channel |
| Conversion cycle *) |  |
| Configurations |  |
| 2 activated channels with 3 -wire measurement | 2 s |
| all others | 1 s |
| Resolution $\quad$ Range 16 bits including sign |  |
| Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range |  |
| PT100, PT1000, Ni1000 | $0,3 \%$ max at $25^{\circ} \mathrm{C}$ <br> $0,5 \% \max$ at full temperature range |
| NTC <br> Reference sensor R: $10000 \Omega$, T: $25^{\circ} \mathrm{C}$ B: 3435 K accuracy specified in the range $806 \Omega-100 \mathrm{k} \Omega$ | $0,3 \%$ max at $25^{\circ} \mathrm{C}$ <br> $0,5 \%$ max at full temperature range |
| Temperature coefficient <br> (is related to the max error at $+25^{\circ} \mathrm{C}$ and max error in the full range) | +-0,006 \%/K |
| Temporary deviation during EMC disturbance | 3\% |
| Unused temperature inputs | Must be configured as "Not used" |
| Overvoltage protection | Yes |

[^5]
## Dimensions



The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |  |
| :--- | :--- | :--- | :---: |
| 1SAP 187 100 R0002 | TA5123-2AI-RTD: AC500-eCo V3, <br> analog input option board, 2AI RTD, <br> 16 bits including sign, spring/cable <br> front terminal 3.50 mm pitch | Active |  |
| 1SAP 187 100 R0202 | TA5123-2AI-RTW: AC500-eCo V3, <br> analog input option board, 2AI RTD, <br> 16 bits including sign, spring/cable <br> front terminal 3.50 mm pitch, wide <br> temperature range | Active |  |
| Spare parts |  |  |  |
| 1SAP 187 400 R0015 <br> $* *$ | TA5220-SPF8: spring terminal block, <br> removable, 8-pin, spring front, cable <br> front, 6 pieces per packing unit | Active |  |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.
$\left.{ }^{* *}\right)$ The needed spring terminal block is always delivered with the option board.
The terminal block listed in the table is for spare part only if needed.

## TA5126-2AO-UI - Analog output option board

## Features

- 2 configurable analog outputs ( O 0 and O 1 ) in 1 group

Resolution 12 bits

- Option board is galvanically isolated
- W variant available for use in extended (wide) temperature range


1 Allocation of signal name
22 yellow LEDs to display the signal states at the analog outputs O0 and O1
3 -pin terminal block for analog output signals and power supply (UP, ZP)
4 Output connection diagram for U and I

## Intended purpose

The option board is used as analog output extension module for AC500-eCo V3 CPUs (PM50xx).


The analog option boards TA5120-2AI-UI and TA5126-2AO-UI can only be used from AB 2.5.2, SystemFW 3.5.0_HF-7, BootFW 3.5.1 and higher.
Using the option boards with lower versions will create a configuration error and the CPU will not start.

## Functionality

2 analog outputs, individually configurable for

- Not used (default setting)
- 0 V... 10 V
- $0 \mathrm{~mA} . .20 \mathrm{~mA}$
- $4 \mathrm{~mA} . .20 \mathrm{~mA}$

| Parameter | Value |
| :---: | :---: |
| Resolution of the analog channels |  |
| Voltage 0 V... 10 V | 12 bits |
| Current 0 mA ... 20 mA , | 12 bits |
| Current 4 mA ... 20 mA | 12 bits |
| LED displays | 2 LEDs for signals O0 and O1 |
| Internal power supply | Via the CPU PM50xx |
| External power supply | Via the terminals UP and ZP (process voltage 24 V DC) |
| Required CPU | PM50xx |

## Connections

## WARNING!

## Removal/Insertion under power

The option boards are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug option boards with the power being ON.
Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace an option board.

Disconnecting any powered option board while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The option board TA5126-2AO-UI for analog output extension is plugged into an AC500-eCo V3 CPU PM50xx.
$\Longleftrightarrow$ Chapter 4.3.11.1 "Mounting and demounting the option boards" on page 127
The electrical connection is made via a removable 6-pin terminal block.
The terminal block is included in the scope of delivery of the option board. Further terminal blocks can be ordered separately as spare parts
Chapter 5.2.8.1.2 "TA52xx(-x) - Terminal block sets" on page 1292

Table 69: Assignment of the terminals:

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 1 | O0+ | Positive analog output O0 |
| 2 | O0- | Negative analog output O0 |
| 3 | O1+ | Positive analog output O1 |
| 4 | O1- | Negative analog output O1 |
| 5 | UP | Process voltage UP $=+24$ V DC |
| 6 | ZP | Process voltage ZP = 0 V DC |

## CAUTION!

The negative terminal of the analog outputs (voltage $0 \mathrm{~V} . .10 \mathrm{~V}$ ) are connected internally and form an internal analog ground (AGND). This analog ground is connected to ZP via a PTC resistor. There is no galvanic isolation between the analog circuitry and ZP/UP.

The internal power supply voltage for the module's circuitry is carried out via the connection to CPU. Thus, the current consumption from 24 V DC power supply at the terminals $\mathrm{L}+$ and M of the CPU module increases by $\ll 1 \mathrm{~mA}$ per TA5126-2AO-UI.
The external power supply connection is carried out via the terminals UP (+24 V DC) and ZP (0 V DC).

## NOTICE!

Risk of damaging the PLC modules!
Overvoltages and short circuits might damage the PLC modules.

Generally, analog signals must be laid in shielded cables. The cable shields must be grounded at both sides of the cables. In order to avoid unacceptable potential differences between different parts of the installation, low resistance equipotential bonding conductors must be laid.

The following figure shows the connection of the module:


Fig. 39: Internal construction of the analog outputs

| Parameter | Value |
| :--- | :--- |
| Configurability | $0 \mathrm{~V} . .10 \mathrm{~V}$ |
|  | $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ |
|  | $4 \mathrm{~mA} . . .20 \mathrm{~mA}$ |
|  | (each input can be configured individu- |
| ally $)$ |  |
| Unused voltage outputs | Must be configured as "not used" |
| Unused current outputs | Must be configured as "not used" |

## Firmware update via memory card

The following steps describe the procedure for updating the firmware for the analog option boards using a memory card. Prerequisite is the previous download of the current firmware to the memory card either from the Automation Builder or as online download from ABB.
Direct from $A B B$ Software.
Click this link and on the next web page find the relevant firmware package and download it.

- Unpack this .zip archive file at any location of your hard disc
- Insert empty formatted (FAT16 / FAT32) memory card in the PC card reader
- Execute the unpacked *.exe file
- Select PC card reader as the final destination and confirm.

All directories, files and SDCARD.INI file will be automatically created on memory card and properly configured. After the process is complete, one has the prepared memory card with relevant updates.

Firmware $\checkmark$ Precondition: Prepared memory card with boot project and firmware $\Rightarrow$ Chapter update 6.3.1.4.5.2.2 "Preparation of memory card" on page 1427.

1. Switch off the device.
2. Insert the memory card.
3. Switch on the device.
$\Rightarrow$ The alternate flashing of the RUN and the ERR LED indicates the running update process.

At the end of the update process a reboot is executed and the system firmware is started for the finishing of the update process.
If RUN LED blinks (ERR LED is off), the update was successful and the display shows done.
If ERR LED blinks (RUN LED is off), the update failed and the display shows FAIL.
The text file "SDCARD.RDY" includes the results of the different updates. If the update fails, the file contains the reasons for the abort. Based on this, further steps can be taken to fix the problem.
4. Switch off the device.
5. Remove the memory card.
6. Switch on the device.
$\Rightarrow$ The system starts with the new firmware.

## Parameterization

The arrangement of the parameter data is performed with Automation Builder software.

1. In the device tree, double-click the desired option board.
2. Select the "TA5126-2AO-UI Parameters" tab to edit the parameterization of the desired option board.

| H0 TA5126_2AO_UI $\times$ |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| TA5126-2AO-UI Parameters | Parameter <br> - Run on config fault | Type <br> Enumeration of BYTE | Value No | Default Value <br> No | Unit | Description |
| TA5126-2AO-UI I/O Mapping |  |  |  |  |  |  |
|  | - Check supply | Enumeration of BYTE | On | On |  | Check supply |
| I/0 mapping list | - Behaviour outputs at comm. error | Enumeration of BYTE | off | Off |  | Output value by communication error |
|  | - Output 0 , channel configuration | Enumeration of BYTE | Not used | Not used |  | Analog output 0-Configuration of analog output channel |
| TA5126-2AO-UI IEC Objects | - Output 0, check channel | Enumeration of BYTE | Plausi, Cut wire, Short dircuit | Plausib, Cut wire, Short arait |  | Analog output 0 - Check channel |
|  | - Output subssitute value for channel 0 and 1 | WORD(0.65535) | 0 | 0 |  | Set the subssitute value for channel 0 and 1 |
| Diagnosis | - Output 1, channel configuration | Enumeration of BYTE | Not used | Not used |  | Analog output 1-Configuration of analog output channel |
|  | - Output 1, check channel | Enumeration of BYTE | Plausib, Cut wire, Short dircuit | Plausib, Cut wire, Short dirait |  | Analog output 1-Check channel |

## Diagnosis



1. In the device tree, double-click the desired option board.
2. Select the "Diagnosis" tab to view the diagnosis messages of the desired option board.

| Bus pos. | Type | Channel | Class | Error | Description | Remedy |  |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Module error (Channel 255) |  |  |  |  |  |  |  |
| $0 . .2$ | 31 | 255 | 3 | 3 | Timeout <br> Communication timeout to <br> CPU | lf the process voltage <br> is connected properly, <br> replace the option board. |  |
| $0 . .2$ | 31 | 255 | 3 | 51 | Invalid slot <br> Wrong or missing option <br> board in the appropriate <br> slot | Check the configuration <br> and the hardware setup. |  |
| $0 . .2$ | 31 | 255 | 4 | 11 | Process voltage is too low | Check the process <br> voltage. |  |
| $0 . .2$ | 31 | 255 | 4 | 34 | Data not ready <br> Data synchronization <br> warning | Check PLC program and <br> synchronize. ${ }^{1}$ ) |  |
| $0 . .2$ | 31 | 255 | 3 | 40 | SW-mismatch <br> Hardware does not match <br> the firmware version | Replace the option board. |  |
| $0 . .2$ | 31 | 255 | 3 | 53 | Download failed <br> Power loss during firm- <br> ware update | Replace the option board. <br> 2 |  |


| Bus pos. | Type | Channel | Class | Error | Description | Remedy |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| $0 . .2$ | 31 | 255 | 3 | 9 | DIAG_BUF_OVERFLOW <br> Overflow in Diagnosis <br> buffer | Diagnosis overflow usu- <br> ally means too many <br> repeated warnings or <br> errors. Please check <br> all diagnosis in detail <br> and take the appropriate <br> action.. A restart will clear <br> the diagnosis. |
| $0 . .2$ | 31 | 255 | 3 | 43 | DIAG_INTERNAL_ERR <br> Internal error in the option <br> board | Replace the option board. |
| 0.2 | 31 | 255 | 3 | 26 | DIAG_CFG_PRM_ERR <br> Parameterization error | Check the CPU parame- <br> terization. |
| 0.2 | 31 | 255 | 3 | 19 | DIAG_CRC_ERR <br> Checksum error in option <br> board | Replace the option board. |
| Channel warning | 1 | $0 . .1$ | 4 | 4 | Highest level <br> Output value is greater <br> than the highest level | Check the output value. |
| $0 . .2$ | 1 | $0 . .1$ | 4 | 7 | Lowest level <br> Output value is lower than <br> the lowest level | Check the output value. |
| 0.2 | 1 | $0 . .1$ | 4 | 46 | DIAG_EXT_VOLTAGE_F <br> EEDED | Check the wiring. ${ }^{3}$ ) |
| Output is short circuited to |  |  |  |  |  |  |
| another voltage |  |  |  |  |  |  |

Remarks:

| $\left.{ }^{1}\right)$ | If no other error occurs, the programs are not identical and the param- <br> eter is set to report an error in this case. |
| :--- | :--- |
| ${ }^{2}$ ) | Do not remove the power supply from the option board during the firm- <br> ware update. |
| $\left.{ }^{3}\right)$ | In this case, output will be switched ON /OFF with a 10 sec cycle |
| ${ }^{4}$ ) | in this case, if the output is configured as voltage output, the output will <br> be switched ON /OFF with a 10 sec cycle |

## State LEDs

|  | LED | State | Color | LED = OFF | LED = ON | LED flashes |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | Outputs O0...O1 | Analog output | Yellow | Output is OFF or output value is too low | Output is ON <br> (brightness depends on the value of the analog signal) | -- |

## Measuring ranges

Measuring The represented resolution corresponds to 12 bits.
ranges - Output
ranges of
voltage and cur- Table 70: Measuring range: $0 \mathrm{~V} \ldots 10 \mathrm{~V}$
rent

| Range | Output [V] | Digital value |  |
| :--- | :---: | :---: | :---: |
|  |  | Decimal | Hex. |
| Overflow | 0 V | $>32504$ | $>7 E F 8$ |
| Output value too high | 11.7564 | 32504 | 7 EF 8 |
|  | $:$ | $:$ | $:$ |
|  | 10.0029 | 27656 | 6 C08 |
| Normal range | 10.0000 | 27648 | 6 C00 |
|  | $:$ | $:$ | $:$ |
|  | 0.0029 | 8 | 0008 |
| Output value too low | 0.0000 | 0 | 0 |
|  | -0.0029 | -8 | FFF8 |
|  | $:$ | -4864 | ED00 |
| Underflow | -1.7593 | $<-4864$ | $<$ ED00 |

Table 71: Measuring range: $0 \mathrm{~mA} . .20 \mathrm{~mA}$

| Range | Output [mA] | Digital value |  |
| :--- | :---: | :---: | :---: |
|  |  | Decimal | Hex. |
| Overflow | 0.0000 | $>32504$ | $>7 E F 8$ |
| Output value too high | 23.5127 | 32504 | 7 EF 8 |
|  | $:$ | $:$ | $:$ |
|  | 20.0058 | 27656 | 6 C08 |
| Normal range | 20.0000 | 27648 | 6 C00 |
|  | $:$ | $:$ | 0008 |
| Underflow | 0.0058 | 8 | 0 |

Table 72: Measuring range: $4 \mathrm{~mA} . . .20 \mathrm{~mA}$

| Range | Output [mA] | Digital value |  |
| :--- | :---: | :---: | :---: |
|  |  | Decimal | Hex. |
| Overflow | 0.0000 | $>32504$ | $>7$ EF8 |
| Output value too high | 22.8102 | 32504 | 7 EF8 |
|  | $:$ | $:$ | $:$ |
|  | 20.0046 | 27656 | 6 C08 |
| Normal range | 20.0000 | 27648 | 6 C00 |
|  | $:$ | $:$ | $:$ |
|  | 4.0046 | 8 | 0008 |
| Output value too low | 4.0000 | 0 | 0 |
|  | 3.995 | -8 | FFF8 |
|  | $:$ | -6904 | E508 |
|  | 0.0046 | -6912 | E500 |
| Underflow | 0.0000 | $<-6912$ | $<$ E500 |

## Technical data

The system data of AC500-eCo V3 apply ${ }^{4}$ ) Chapter 5.1.1 "System data AC500-eCo" on page 159
Only additional details are therefore documented below.

| Parameter | Value |
| :---: | :---: |
| Process voltage |  |
| Connections | Terminal 5 for $+24 \mathrm{~V}(\mathrm{UP})$ as well as terminal 6 for 0 V (ZP) |
| Rated value | 24 V DC |
| Protection against reversed voltage | Yes |
| Rated protection fuse on UP | 1 A slow |
| Galvanic isolation | Yes, per module (no isolation between channels) |
| Current consumption |  |
| From 24 V DC power supply at the terminals UP/L+ and ZP/M of the CPU PM50xx | << 1 mA |
| From UP at normal operation | max. 70 mA |
| Inrush current from UP (at power up) | $0.005 \mathrm{~A}^{2} \mathrm{~s}$ |
| Max. length of analog cables, conductor cross section $>0.2 \mathrm{~mm}^{2}$ | On request |
| Weight | 20 g |


| Parameter | Value |
| :--- | :--- |
| Mounting position | Horizontal or vertical with derating <br> (output load reduced to $50 \%$ at $+40^{\circ} \mathrm{C}$ <br> per group) |
| Cooling | The natural convection cooling must not <br> be hindered by cable ducts or other parts <br> in the control cabinet. |

## Technical data of the analog outputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 2 |
| Distribution of channels into groups | 1 group of 2 channels |
| Connections of the channel 0 |  |
| Voltage mode |  |
| Terminal O0+ and O0- | Terminals 1 and 2 |
| Current mode |  |
| Terminal O0+ and ZP | Termnal 1 and 6 |
| Connections of the channel 1 |  |
| Voltage mode |  |
| Terminal O1+ and O1- | Terminals 3 and 4 |
| Current mode |  |
| Terminal O1+ and ZP | Terminals 3 and 6 |
| Output type | Unipolar |
| Galvanic isolation | Yes, per option board (no isolation between channels) |
| Configurability | $\begin{aligned} & 0 \mathrm{~V} \ldots 10 \mathrm{~V} \\ & 0 \mathrm{~mA} \ldots 20 \mathrm{~mA} \\ & 4 \mathrm{~mA} \ldots 20 \mathrm{~mA} \\ & \text { (each output can be configured individu- } \\ & \text { ally) } \end{aligned}$ |
| Output resistance (load), as current output | $0 \Omega . .500 \Omega$ |
| Output loadability, as voltage output | Max. $\pm 10 \mathrm{~mA}$ |
| Indication of the output signals | One LED per channel |
| Conversion cycle |  |
| 1 activated channel | 250 ¢s |
| 2 activated channel | $500 \mu \mathrm{~s}$ |
| Resolution for all configurations | 12 bits |
| Settling time for full range change (resistive load, output signal within specified tolerance) | Typ. 5 ms |


| Parameter | Value |  |
| :--- | :--- | :--- |
| Conversion error of the analog values caused by <br> non-linearity, adjustment error at factory and resolu- <br> tion within the normal range | Max. | $\pm 0.3 \%$ at $+25^{\circ} \mathrm{C}$ |
|  | Max. | $\pm 0.3 \%$ over full temperature <br> range |
| Temperature coefficient <br> (is related to the max error at $+25^{\circ} \mathrm{C}$ and max error <br> in the full range) | $\pm 0.003 \% / \mathrm{K}$ |  |
| Temporary deviation during EMC disturbance |  |  |
| Mapping between output signal and digital value | $\pm 3 \%$ |  |
| Unused voltage outputs | Output ranges of voltage and current |  |
| Unused current outputs | Must be configured as "not used" |  |

## Dimensions



The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |  |  |
| :--- | :--- | :--- | :---: | :---: |
| 1SAP 187 100 R0003 | TA5126-2AO-UI: AC500-eCo V3, <br> analog output option board, 2AO U/I, <br> 12 bits, spring/cable front terminal <br> 3.50 mm pitch | Active |  |  |
| 1SAP 187 100 R0203 | TA5126-2AO-UIW: AC500-eCo V3, <br> analog output option board, 2AO U/I, <br> 12 bits, spring/cable front terminal <br> 3.50 mm pitch, wide range tempera- <br> ture | Active |  |  |
| Spare parts | 1SAP 187 400 R0013 <br> $* *)$TA5220-SPF6: spring terminal block, <br> removable, 6-pin, spring front, cable <br> front, 6 pieces per packing unit |  |  | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.
${ }^{* *}$ ) The needed spring terminal block is always delivered with the option board.
The terminal block listed in the table is for spare part only if needed.

TA5130-KNXPB - Option board KNX address push button

## Features

- W variant available for use in extended (wide) temperature range


1 State LED
2 Allocation of signal name
3 Connector

## Intended purpose



This option board is only intended to be used with PM5072-T-2ETH(W) and PM5082-T-2ETH.

This option board can only be used once on one slot at a time!
The option board is not supported by other AC500-eCo V3 PLCs.
ⓨ Information about the interface to KNX

## Parameterization

The arrangement of the parameter data is performed with Automation Builder software.

1. In the device tree, double-click the desired option board.
2. Select the "TA51xx Parameters" tab to edit the parameterization of the desired option board.


## State LEDs

| Signal | Color | State | Description |
| :--- | :--- | :--- | :--- |
| PRG | Red | ON | Programming state |

## Technical data

The system data of AC500-eCo V3 apply ${ }^{\text {² }}$ Chapter 5.1.1 "System data AC500-eCo" on page 159

Only additional details are therefore documented below.

| Parameter | Value |
| :--- | :--- |
| Usable CPUs | PM5072-T-2ETH(W) and PM5082-T-2ETH |
| Internal power supply | Via internal CPU connection |
| Additional current consumption from 24 V DC <br> power supply at CPU | Max. 25 mA |
| Weight | 14 g |

## Dimensions



The dimensions are in mm and in brackets in inch.

Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 187 200 R0001 | TA5130-KNXPB: AC500-eCo V3, <br> option board KNX address push <br> button for AC500-eCo V3 Pro CPU <br> only | Active |
| 1SAP 187 200 R0201 | TA5130-KNXPBW: AC500-eCo V3, <br> option board KNX address push <br> button for AC500-eCo V3 Pro CPU <br> only, wide temperature range | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

## TA5131-RTC - Option board for real-time clock

## Features

This option board is only for the basic CPUs PM5012-T-ETH and PM5012-R-ETH.
All other AC500-eCo V3 CPUs have the real-time clock already integrated.
(n) Information about the real-time clock


1 TA5131-RTC option board

## Parameterization

The arrangement of the parameter data is performed with Automation Builder software.

1. In the device tree, double-click the desired option board.
2. Select the "TA51xx Parameters" tab to edit the parameterization of the desired option board.


## Technical data

The system data of AC500-eCo V3 apply ${ }^{\wedge}$ Chapter 5.1.1 "System data AC500-eCo" on page 159
Only additional details are therefore documented below.

| Parameter |  |
| :--- | :--- |
| Buffering time (if at least powered for 8 hours) | Value |
|  | 7 days at room tempera- <br> ture |
| Usable CPUs | $\pm 2$ s/day |


| Parameter | Value |
| :--- | :--- |
| Internal power supply | Via internal CPU connec- <br> tion |
| Additional current consumption from 24 V DC power supply at <br> CPU | Max. 25 mA |
| Weight | 16 g |

## Dimensions



The dimensions are in mm and in brackets in inch.

Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 187 200 R0002 | TA5131-RTC:AC500-eCo V3, real- <br> time clock without battery, option <br> board for AC500-eCo V3 Basic CPU <br> only | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

## TA5141-RS232I - Option board for COMx serial communication

## Features

- W variant available for use in extended (wide) temperature range


12 LEDs for communication state display (TxD and RxD)
2 Allocation of signal name
3 5-pin terminal block for communication interface

Option board for COMx serial communication TA5141-RS232I(W) is equipped with 1 RS-232 serial interface with handshake.

The serial interface option boards TA5141-RS232I(W), TA5142-RS485I(W) and TA5142-RS485(W) produced from 2023 can only be used from AB 2.5.2, SystemFW 3.5.0_HF-7, BootFW 3.5.1 and higher. These new produced option boards can be recognized with the Date/Code printed on the sticker or also on the product packaging, the date code looks like $2 x x 3$, where 2 and 3 are for the year and the xx for the production week.
Using the option boards with lower versions will create a configuration error and the CPU will not start.
These new produced option boards cannot be used with lower Automation Builder Version and the CPU BootFW/FW and the application must be upgraded to $A B$ 2.5.2 HF8 or higher.
Application note AC500-eCo V3 - FAQ

## Connections

## Serial interfaces

## NOTICE! <br> Damage to the serial communication interface by using 5-pin terminal block of the TA5101-4DI!

If the 5-pin terminal block of the TA5101-4DI option board is plugged into a option board for COMx serial communication TA5141-RS232I, TA5142-RS485I or TA5142-RS485, the communication interface will be damaged by the 24 V .

Please do not confuse the 5-pin terminal block of the TA5101-4DI with the 5-pin terminal block for serial communication interface of TA5141-RS232I, TA5142RS485I or TA5142-RS485.

Table 73: TA5141-RS232I

| Serial interface | Pin | Signal | Description |
| :--- | :--- | :--- | :--- |
|  | 1 | RTS | Request To Send <br> DCE is ready to accept data from the DTE |
|  | 2 | TxD | Transmit data (output) |
| 2 | 3 | GND | Common Ground |
|  | 4 | RxD | Receive data (input) |

Cable length The maximum possible cable length of a serial connection subnet within a segment depends on the transmission rate.

RS-232 for point-to-point connection:

| Parameter | Value |
| :--- | :--- |
| Transmission rate | $9.6 \mathrm{kBit} / \mathrm{s}$ to $115.2 \mathrm{kBit} / \mathrm{s}$ |
| Maximum cable length | On request |

## Parameterization

The arrangement of the parameter data is performed with Automation Builder software.

1. In the device tree, double-click the desired option board.
2. Select the "TA51xx Parameters" tab to edit the parameterization of the desired option board.

| TA5141-RS232I Parameters | Parameter <br> - Run on config fault | Type | Value | Default Value | Unit | Description |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Information |  | Enumeration of BYTE | No | No |  | Start PLC program even on configuration fault |
|  | - Baudrate | Enumeration of DWORD | 19200 | 19200 | Bits/s | Set the baudrate in Bits per seconds |
|  | - Parity | Enumeration of BYTE | None | None |  | Set the parity Bit type |
|  | - Data Bits | Enumeration of BYTE | 8 | 8 | Bits/character | Set the character size |
|  | - Stop Bits | Enumeration of BYTE | 1 | 1 |  | Set the number of stop Bits per character 2 means 1,5 when character size is 5 Bits |
|  | - Flow control | Enumeration of BYTE | No flow control | No flow control |  | Flow control |

## State LEDs

| Signal | Color | State | Description |
| :--- | :--- | :--- | :--- |
| TxD | Yellow | ON (blinking) | Transmitting |
| RxD | Yellow | ON (blinking) | Receiving |

## Technical data

The system data of AC500-eCo V3 apply ${ }^{\text {² }}$ Chapter 5.1.1 "System data AC500-eCo" on page 159

Only additional details are therefore documented below.

| Parameter | Value |
| :--- | :--- |
| Protocol | Programmable with Automation Builder e.g. <br> Modbus RTU / CAA SerialCom via serial inter- <br> faces |
| Interface | Serial interface |
| Serial interface standard | ElA RS-232 |
| Potential separation | Yes, from the CPU, 500 V DC |
| Serial interface parameters | Configurable via software |
| Modes of operation | Data exchange |
| Transmission rate | 9.6 kbit/s to 115.2 kbit/s |
| Protocol | Programmable |
| Interface connector | $5-$ pin terminal block, male |
| Usable CPUs | PM50xx |
| Internal power supply | Via internal CPU connection |
| Additional current consumption from 24 V DC <br> power supply at CPU | Max. 25 mA |
| Weight | Ca. 15 g |

## Dimensions



The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 187 300 R0001 | TA5141-RS232I: AC500-eCo V3, <br> RS-232 isolated option board for <br> COMx serial communication, spring/ <br> cable front terminal, 3.50 mm pitch | Active |
| 1SAP 187 300 R0201 | TA5141-RS232IW: AC500-eCo V3, <br> RS-232 isolated option board for <br> COMx serial communication, spring/ <br> cable front terminal, 3.50 mm pitch, <br> wide temperature range | Active |
| Spare parts |  |  |
| 1SAP 187 400 R0012 <br> $* *$ | TA5220-SPF5: spring terminal block, <br> removable, 5-pin, spring front, cable <br> front, 3.5 mm pitch, 6 pieces per <br> packing unit | Active |

## *) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

$\left.{ }^{* *}\right)$ The needed spring terminal block is always delivered with the option board. The terminal block listed in the table is for spare part only if needed.

TA5142-RS485I - Option board for COMx serial communication

## Features

- W variant available for use in extended (wide) temperature range


12 LEDs for communication state display (TxD and RxD)
22 LEDs for termination state display
3 Allocation of signal name
4 5-pin terminal block for communication interface

Option boards for COMx serial communication TA5142-RS485I(W) and TA5142-RS485(W) are equipped with 1 RS-485 (2-wire half-duplex) serial interface which can be used for communication via Modbus RTU or CAA SerialCom.

Bus terminations are built-in and configurable.

The serial interface option boards TA5141-RS232I(W), TA5142-RS485I(W) and TA5142-RS485(W) produced from 2023 can only be used from AB 2.5.2, SystemFW 3.5.0_HF-7, BootFW 3.5.1 and higher. These new produced option boards can be recognized with the Date/Code printed on the sticker or also on the product packaging, the date code looks like $2 x x 3$, where 2 and 3 are for the year and the xx for the production week.
Using the option boards with lower versions will create a configuration error and the CPU will not start.
These new produced option boards cannot be used with lower Automation Builder Version and the CPU BootFW/FW and the application must be upgraded to $A B$ 2.5.2 HF8 or higher.
Application note AC500-eCo V3 - FAQ

## Connections

## Serial interfaces

## NOTICE!

Damage to the serial communication interface by using 5-pin terminal block of the TA5101-4DI!
If the 5-pin terminal block of the TA5101-4DI option board is plugged into a option board for COMx serial communication TA5141-RS232I, TA5142-RS485I or TA5142-RS485, the communication interface will be damaged by the 24 V .

Please do not confuse the 5-pin terminal block of the TA5101-4DI with the 5-pin terminal block for serial communication interface of TA5141-RS232I, TA5142RS485I or TA5142-RS485.

Table 74: TA5142-RS485(I)

| Serial interface | Pin | Signal |
| :--- | :--- | :--- |
|  | 1 | A1 <br> internally connected to A2 |
|  | 2 | B1 <br> internally connected to B2 |
|  | 3 | GND |

Protocols

| No. | Protocol | Description |
| :--- | :--- | :--- |
| 1 | Modbus | Modbus RTU, master or slave |
| 2 | CAA SerialCom | Support for blocks contained in the CAA_SerialCom.lib library |

Bus cable

| Bus line |  |
| :--- | :--- |
| Construction | 2 cores, twisted, with common shield |
| Conductor cross section | $>0.22 \mathrm{~mm}^{2}(24$ AWG) |
| Twisting rate | $>10$ per meter (symmetrically twisted) |
| Core insulation | Polyethylene (PE) |
| Resistance per core | $<100 \Omega / \mathrm{km}$ |
| Characteristic impedance | ca. $120 \Omega(100 \Omega \ldots 150 \Omega)$ |
| Capacitance between the cores | $<55 \mathrm{nF} / \mathrm{km}$ (if higher, the max. bus length must be reduced) |
| Terminating resistors | $120 \Omega 1 / 4 \mathrm{~W}$ at both line ends |
| Remarks | Commonly used telephone cables with PE insulation and a <br> core diameter of $>0.8$ mm are usually sufficient. |
|  | Cables with PVC core insulation and core diameter of <br> $0.8 ~ m m ~ c a n ~ b e ~ u s e d ~ u p ~ t o ~ a ~ l e n g t h ~ o f ~ a p p r o x . ~$ 50 m. In |
| this case, the bus terminating resistor is approx. $100 \Omega$. |  |

Cable length The maximum possible cable length of a serial connection subnet within a segment depends on the transmission rate.

RS-485 for point-to-point or bus connection:

| Parameter | Value |
| :--- | :--- |
| Transmission rate | $9.6 \mathrm{kbit} / \mathrm{s}$ to $115.2 \mathrm{kbit} / \mathrm{s}$ |
| Maximum cable length | On request |

Bus termination The line ends of the bus segment must be equipped with bus terminating resistors. These resistors are integrated in the module TA5142-RS485I. The pull-up and pull-down settings must also be made on the circuit board of the module.


1 Termination resistance settings
2 Pull-up and pull-down settings

Table 75: Configuration

| Settings on the module | State of LEDs | Internal wiring diagram | Description |
| :---: | :---: | :---: | :---: |
|  | TXD $\square$ <br> RXD $\square$ <br> 120R $\square$ <br> PUD $\square$ |  | Master at the bus line end, pull-up and pull-down activated, bus termination $120 \Omega$ |
|  | TXD $\square$ <br> RXD $\square$ <br> 120R $\square$ <br> PUD $\square$ |  | Master within the bus line, pull-up and pull-down activated |


| Settings on the module | State of LEDs | Internal wiring diagram | Description |
| :---: | :---: | :---: | :---: |
|  | TXD <br> RXD $\square$ <br> 120R $\square$ <br> PUD |  | Slave at the bus line end, bus termination $120 \Omega$ |
|  | TXD <br> RXD <br> 120R <br> PUD |  | Slave within the bus line |

## Parameterization

The arrangement of the parameter data is performed with Automation Builder software.

1. In the device tree, double-click the desired option board.
2. Select the "TA51xx Parameters" tab to edit the parameterization of the desired option board.


State LEDs

|  | Signal | Color | State | Description |
| :---: | :---: | :---: | :---: | :---: |
| TXD | TxD | Yellow | ON (blinking) | Transmitting |
|  | RxD | Yellow | ON (blinking) | Receiving |
| RXD | 120R | Yellow | ON | Bus termination |
| 120P | PUD | Yellow | ON | Pull-up / Pull-down |
| PUD |  |  |  |  |

## Technical data

The system data of AC500-eCo V3 apply ${ }^{\wedge}$ chapter 5.1.1 "System data AC500-eCo" on page 159
Only additional details are therefore documented below.

Table 76: TA5142-RS485I

| Parameter | Value |
| :--- | :--- |
| Protocol | Programmable with Automation Builder e.g. <br> Modbus RTU / CAA_SerialCom via serial <br> interfaces |
| Interface | Serial interface |
| Serial interface standard | EIA RS-485 |
| Potential separation | Yes, from the CPU, 500 V DC |
| Serial interface parameters | Configurable via software |
| Modes of operation | Data exchange |
| Transmission rate | 9.6 kbit/s to 115.2 kbit/s |
| Protocol | Programmable |
| Interface connector | 5 -pin terminal block, male |
| Usable CPUs | PM50xx |
| Internal power supply | Via internal CPU connection |
| Additional current consumption from 24 V DC <br> power supply at CPU | Max. 25 mA |
| Weight | Ca. 16 g |

## Dimensions



The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 187 300 R0002 | TA5142-RS485I: AC500-eCo V3, <br> RS-485 isolated option board for <br> COMx serial communication, spring/ <br> cable front terminal, 3.50 mm pitch | Active |
| 1SAP 187 300 R0202 | TA5142-RS485IW: AC500-eCo V3, <br> RS-485 isolated option board for <br> COMx serial communication, spring/ <br> cable front terminal, 3.50 mm pitch, <br> wide range temperature | Active |
| Spare parts |  |  |
| 1SAP 187 400 R0012 <br> $* *$ | TA5220-SPF5: spring terminal block, <br> removable, 5-pin, spring front, cable <br> front, 3.5 mm pitch, 6 pieces per <br> packing unit | Active |

## *) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

${ }^{* *}$ ) The needed spring terminal block is always delivered with the option board. The terminal block listed in the table is for spare part only if needed.

## TA5142-RS485- Option board for COMx serial communication

## Features

- W variant available for use in extended (wide) temperature range


12 LEDs for communication state display (TxD and RxD)
22 LEDs for termination state display
3 Allocation of signal name
4 5-pin terminal block for communication interface

Option boards for COMx serial communication TA5142-RS485I(W) and TA5142-RS485(W) are equipped with 1 RS-485 (2-wire half-duplex) serial interface which can be used for communication via Modbus RTU or CAA SerialCom.

Bus terminations are built-in and configurable.

The serial interface option boards TA5141-RS232I(W), TA5142-RS485I(W) and TA5142-RS485(W) produced from 2023 can only be used from AB 2.5.2, SystemFW 3.5.0_HF-7, BootFW 3.5.1 and higher. These new produced option boards can be recognized with the Date/Code printed on the sticker or also on the product packaging, the date code looks like $2 x x 3$, where 2 and 3 are for the year and the xx for the production week.
Using the option boards with lower versions will create a configuration error and the CPU will not start.

These new produced option boards cannot be used with lower Automation Builder Version and the CPU BootFW/FW and the application must be upgraded to $A B$ 2.5.2 HF8 or higher.
Application note AC500-eCo V3-FAQ

## Connections

## Serial interfaces

## NOTICE!

## Damage to the serial communication interface by using 5-pin terminal block of the TA5101-4DI!

If the 5-pin terminal block of the TA5101-4DI option board is plugged into a option board for COMx serial communication TA5141-RS232I, TA5142-RS485I or TA5142-RS485, the communication interface will be damaged by the 24 V .

Please do not confuse the 5-pin terminal block of the TA5101-4DI with the 5-pin terminal block for serial communication interface of TA5141-RS232I, TA5142RS485I or TA5142-RS485.

Table 77: TA5142-RS485(I)

| Serial interface | Pin | Signal |
| :--- | :--- | :--- |
|  | 1 | A1 <br> internally connected to A2 |
|  | 2 | B1 <br> internally connected to B2 |
|  | 3 | GND |

Protocols

| No. | Protocol | Description |
| :--- | :--- | :--- |
| 1 | Modbus | Modbus RTU, master or slave |
| 2 | CAA SerialCom | Support for blocks contained in the CAA_SerialCom.lib library |

Bus cable

| Bus line |  |
| :--- | :--- |
| Construction | 2 cores, twisted, with common shield |
| Conductor cross section | $>0.22 \mathrm{~mm}^{2}(24$ AWG) |
| Twisting rate | $>10$ per meter (symmetrically twisted) |
| Core insulation | Polyethylene (PE) |
| Resistance per core | $<100 \Omega / \mathrm{km}$ |
| Characteristic impedance | ca. $120 \Omega(100 \Omega \ldots 150 \Omega)$ |
| Capacitance between the cores | $<55 \mathrm{nF} / \mathrm{km}$ (if higher, the max. bus length must be reduced) |
| Terminating resistors | $120 \Omega 1 / 4 \mathrm{~W}$ at both line ends |
| Remarks | Commonly used telephone cables with PE insulation and a <br> core diameter of $>0.8$ mm are usually sufficient. |
|  | Cables with PVC core insulation and core diameter of <br> $0.8 ~ m m ~ c a n ~ b e ~ u s e d ~ u p ~ t o ~ a ~ l e n g t h ~ o f ~ a p p r o x . ~$ 50 m. In |
| this case, the bus terminating resistor is approx. $100 \Omega$. |  |

Cable length The maximum possible cable length of a serial connection subnet within a segment depends on the transmission rate.

RS-485 for point-to-point or bus connection:

| Parameter | Value |
| :--- | :--- |
| Transmission rate | $9.6 \mathrm{kbit} / \mathrm{s}$ to $115.2 \mathrm{kbit} / \mathrm{s}$ |
| Maximum cable length | On request |

Bus termination The line ends of the bus segment must be equipped with bus terminating resistors. These resistors are integrated in the module TA5142-RS485. The pull-up and pull-down settings must also be made on the circuit board of the module.


1 Termination resistance settings
2 Pull-up and pull-down settings

Table 78: Configuration

| Settings on the module | State of LEDs | Internal wiring diagram | Description |
| :---: | :---: | :---: | :---: |
|  | TXD $\square$ <br> RXD $\square$ <br> 120R $\square$ <br> PUD $\square$ |  | Master at the bus line end, pull-up and pull-down activated, bus termination $120 \Omega$ |
|  | TXD <br> RXD <br> 120R <br> PUD $\square$ |  | Master within the bus line, pull-up and pull-down activated |



Parameterization
The arrangement of the parameter data is performed with Automation Builder software.

1. In the device tree, double-click the desired option board.
2. Select the "TA51xx Parameters" tab to edit the parameterization of the desired option board.


## State LEDs

|  | Signal | Color | State | Description |
| :--- | :--- | :--- | :--- | :--- |
| TXD | TxD | Yellow | ON (blinking) | Transmitting |
|  | RxD | Yellow | ON (blinking) | Receiving |
| RXD | 120R | Yellow | ON | Bus termination |
| $120 R \square$ | PUD | Yellow | ON | Pull-up / Pull-down |
|  |  |  |  |  |

## Technical data

The system data of AC500-eCo V3 apply ${ }^{\text {² }}$ Chapter 5.1.1 "System data AC500-eCo" on page 159

Only additional details are therefore documented below.

Table 79: TA5142-RS485

| Parameter | Value |
| :--- | :--- |
| Protocol | Programmable with Automation Builder e.g. <br> Modbus RTU / CAA_SerialCom via serial <br> interfaces |
| Interface | Serial interface |
| Serial interface standard | ElA RS-485 |
| Potential separation | No |
| Serial interface parameters | Configurable via software |
| Modes of operation | Programming or data exchange |
| Transmission rate | 9.6 kbit/s to 115.2 kbit/s |
| Protocol | Programmable |
| Interface connector | $5-$ pin terminal block, male |
| Usable CPUs | PM50xx |
| Internal power supply | Via internal CPU connection |
| Additional current consumption from 24 V DC <br> power supply at CPU | Max. 25 mA |
| Weight | Ca. 15 g |

## Dimensions



The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 187 300 R0003 | TA5142-RS485: AC500-eCo V3, <br> RS-485 non isolated option board for <br> COMx serial communication, spring/ <br> cable front terminal, 3.50 mm pitch | Active |
| 1SAP 187 300 R0203 | TA5142-RS485W: AC500-eCo V3, <br> RS-485 non isolated option board for <br> COMx serial communication, spring/ <br> cable front terminal, 3.50 mm pitch, <br> wide range temperature | Active |
| Spare parts |  |  |
| 1SAP 187 400 R0012 <br> $* *$ | TA5220-SPF5: spring terminal block, <br> removable, 5-pin, spring front, cable <br> front, 3.5 mm pitch, 6 pieces per <br> packing unit | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.
${ }^{* *}$ ) The needed spring terminal block is always delivered with the option board.
The terminal block listed in the table is for spare part only if needed.

## FAQs

Frequently asked and important questions about the AC500-eCo V3 serial communication option boards from 2023 are collected and answered in the application note. AC500 V3-eCo FAQ - TA514X-RSXXX Serial Option Boards.

### 5.2.1.2 AC500 and AC500-XC

### 5.2.1.2.1 Product overview and comparison

## Comparison of AC500 V3 terminal bases

With the latest Automation Builder version the following terminal bases are compatible with the AC500 V3 processor modules:

Terminal base compatibility


The number of slots that are available on a terminal base for connecting communication modules or AC500-S modules differs within the terminal base range.
Table 80: Combination of TB56xx-2ETH(-XC) and PM56xx (-XC)

| Processor module | PM5630 | PM5650 | PM5670 | PM5675 |
| :--- | :--- | :--- | :--- | :--- |
| TB5600-2ETH | 0 slot | 0 slot | 0 slot | 0 slot |
| TB5610-2ETH | 1 slot | 1 slot | 1 slot | 1 slot |
| TB5620-2ETH | 2 slots | 2 slots | 2 slots | 2 slots |
| TB5640-2ETH | - | 4 slots | 4 slots | 4 slots |
| TB5660-2ETH | - | - | 6 slots $\left.{ }^{1}\right)$ | 6 slots $\left.{ }^{1}\right)$ |

## Remarks:

The slots can be used for connecting communication modules or AC500-S modules. Note that only one AC500-S module can be connected at one terminal base.
${ }^{1}$ ) PM567x must have an index $\geq \mathrm{C} 0$.

Supported devices

The AC500 V3 terminal bases can be equipped with the following supported devices:
Table 81: Comparison: TB56xx

| Processor module | PM5630 | PM5650 | PM5670 | PM5675 |
| :---: | :---: | :---: | :---: | :---: |
| Max. number of variables allowed for each communication module supported |  |  |  |  |
| Input variables | 4 kB | 4 kB | 5 kB | 5 kB |
| Output variables | 4 kB | 4 kB | 5 kB | 5 kB |
| Type of communication module supported |  |  |  |  |
| CM574-RS/RCOM - serial interface | No | No | No | No |
| CM5610-2RS - 2 serial interfaces | X | X | X | X |
| CM582-DP - PROFIBUS DP V0/V1 slave | X | X | X | X |
| CM592-DP - PROFIBUS DP V0/V1 master | X | X | X | X |
| CM579-ETHCAT - EtherCAT master | X | X | X | X |
| CM579-PNIO - PROFINET IO RT controller | X | X | X | X |
| CM589-PNIO - PROFINET IO RT device | x | x | X | X |
| CM589-PNIO-4 - PROFINET IO RT with 4 devices | X | X | X | X |
| CM597-ETH - Ethernet interface | No | No | No | No |
| CM5640-2ETH - 2 Ethernet interfaces | X | X | X | X |
| CM588-CN - CAN, CANopen slave | No | No | No | No |
| CM598-CN - CAN, CANopen master | only CAN 2A/2B | only CAN 2A/2B | only CAN 2A/2B | only CAN 2A/2B |
| Type of AC500-S module supported |  |  |  |  |
| SM560-S - safety module | X | X | X | X |
| SM560-S-FD-1 - safety module with F-Device functionality for 1 PROFIsafe network | X | X | X | X |
| SM560-S -FD-4 - safety module with F-Device functionality for 1 PROFIsafe network | X | X | X | X |

Memory size and performance

Table 82: Comparison: PM56xx

| Processor module |  | PM5630 | $\begin{array}{\|l} \hline \text { PM565 } \\ 0 \end{array}$ | $\begin{array}{\|l\|} \hline \text { PM567 } \\ 0 \end{array}$ | PM5675 |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Total maximum downloadable application size ${ }^{\text {1) }}$ |  | 9 MB | 84 MB | $\begin{aligned} & \hline 176 \\ & \text { MB } \end{aligned}$ | 176 MB |
|  | Thereof user program code and data (dynamically allocated) | 2 MB | 8 MB | 32 MB | 32 MB |
|  | Thereof user webserver data max. | 6 MB | 72 MB | $\begin{aligned} & 128 \\ & \text { MB } \end{aligned}$ | 128 MB |
| Flash memory for User data |  |  |  |  |  |
|  | Remaining for all other usage (project save, infrastructure...) | 30 MB | $\begin{aligned} & 285 \\ & \text { MB } \end{aligned}$ | $\begin{aligned} & 643 \\ & \text { MB } \end{aligned}$ | 643 MB |
| Buffered (SRAM) |  | 256 kB | 256 kB | 1.5 MB | 1.5 MB |
|  | Thereof VAR retain persistent | 128 kB | 128 kB | $\begin{aligned} & 1024 \\ & \text { kB } \end{aligned}$ | 1024 kB |

$\left.\begin{array}{|l|l|l|l|l|l|l|l|}\hline \text { Processor module } & \text { PM5630 } & \begin{array}{l}\text { PM565 } \\ \mathbf{0}\end{array} & \begin{array}{l}\text { PM567 } \\ \mathbf{0}\end{array} & \text { PM5675 } \\ \hline \text { Modbus register) }\end{array}\right)$

| Processor module |  | PM5630 | PM565 | $\begin{aligned} & \text { PM567 } \\ & 0 \end{aligned}$ | PM5675 |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Program execution: |  |  |  |  |  |
|  | Cyclic | x |  |  |  |
|  | Time-controlled | x |  |  |  |
|  | Multitasking | X |  |  |  |
|  | Minimum cycle time configurable for cyclical task | 1 ms | 1 ms | $0,5 \mathrm{~ms}$ | 0,5 ms |
| User program protection by password |  | x (user management) |  |  |  |
| Internal interfaces for communication: |  |  |  |  |  |
| Serial interface COM1: |  |  |  |  |  |
|  | Physical link | Configurable for RS-232 or RS-485 ( $9.6 \mathrm{~kb} / \mathrm{s}, 19.2 \mathrm{~kb} / \mathrm{s}, 38.4 \mathrm{~kb} / \mathrm{s}, 57.6$ $\mathrm{kb} / \mathrm{s}$ and $115.2 \mathrm{~kb} / \mathrm{s}$ ) |  |  |  |
|  | Connection | Pluggable terminal block, spring connection |  |  |  |
|  | Usage | Serial ASCII communication,Modbus RTU |  |  |  |
| CAN interface: |  |  |  |  |  |
|  | Physical link | CAN 2A/2B (from $50 \mathrm{~kb} / \mathrm{s}$ to $1 \mathrm{Mb} / \mathrm{s}$ ) |  |  |  |
|  | Connection | Pluggable terminal block, spring connection |  |  |  |
|  | Usage | CANopen master communication, CAN 2A/2B, J1939 protocol, CAN sync |  |  |  |
|  | Max. number of variables allowed |  |  |  |  |
|  | Input variables | 2 kB | 4 kB | 5 kB | 5 kB |
|  | Output variables | 2 kB | 4 kB | 5 kB | 5 kB |
| Network interface ETH1, ETH2: |  |  |  |  |  |
|  | Usage | Ethernet |  |  |  |
|  | Physical link | 10/100 base-TX, configurable as internal switch or independent Interfaces |  |  |  |
|  | Connection | 2x RJ45 socket, provided on TB56xx-2ETH |  |  |  |
| LEDs, LCD display, function keys |  | RUN / STOP, status, diagnosis, settings |  |  |  |
| Number of timers |  | Unlimited |  |  |  |
| Number of counters |  | Unlimited |  |  |  |
| Programming languages: |  |  |  |  |  |
|  | Structured Text ST | x |  |  |  |
|  | Instruction list IL | X |  |  |  |
|  | Function Block Diagram FBD | X |  |  |  |
|  | Ladder Diagram LD | X |  |  |  |
|  | Sequential function chart SFC | X |  |  |  |
|  | Continuous function chart (CFC) | x |  |  |  |

Remarks:
1): The values are for information only and cannot be fulfilled altogether. The available resources are limited at the end by the maximal downloadable application size for each CPU.

## Comparison of features and protocols

Communication and onboard protocols

Table 83: OPC UA server / OPC DA server

| Processor module |  | PM5630 | PM5650 | PM5670 | PM5675 |
| :--- | :--- | :--- | :--- | :--- | :--- |
| OPC UA server |  | x | $x$ | $x$ | $x$ |
|  | Number of free tags | 3.000 | 10.000 | 30.000 | 30.000 |
|  | Number of connections | 10 | 20 | 50 | 50 |
|  | Min. sampling rate (limit) | 500 ms | 100 ms | 50 ms | 50 ms |
| OPC DA server AE |  | x | x | x | x |
| Number of connections |  |  |  |  |  |

Table 84: Modbus, Telecontrol

| Processor module | PM5630 | PM5650 | PM5670 | PM5675 |
| :--- | :--- | :--- | :--- | :--- |
| Modbus TCP client / server |  | x | x | x |
| Number of Modbus clients ModMast <br> in parallel on a CPU master (server) | 30 | 50 | 120 | 120 |
|  | Number of Modbus server in parallel <br> (e.g. for SCADA access) | 15 | 25 | 50 |
| IEC 60870-5-104 telecontrol protocol | x | x | x |  |
|  | Number of free tags | 1.000 | 5.000 | 10.000 |
|  | Control station (number of connec- <br> tions) | 5 | 10 | 20 |
|  | Sub-station (number of connections) | 5 | 10 | 20 |

## Ethernet protocols and ports for AC500 V3 products

## General

The communication module CM5640-2ETH acts as a port extender of the used AC500 V3 CPU. It is not offloading the CPU for the protocols.

- the stacks are still executed in CPU.
- the performance is slightly lower or the load is slightly higher than directly from the main CPU, because each communication module and the communication module bus must also be handled.
Each CM5640-2ETH provides:
- 2 additional independent onboard Ethernet interfaces controlled by the CPU and not switched.
- all Ethernet ports, wherever located - on the communication module or on the CPU - must be in different subnets.
Addressing is not done via slot numbers but via IP addresses.

The Ethernet limitations of the CPU used together with CM5640-2ETH apply to all onboard and additional interfaces.

All interfaces of the onboard and CM5640-2ETH modules need to be in different subnets. Overlapping of subnets must be avoided. The selection of the used interface (ETH port) is done automatically according the match of the IP-address of the server and the subnet of the interfaces.

| Description | $\geq$ CPU firmware | $\begin{gathered} \text { PM563 } \\ \text { O-2ET } \\ H \end{gathered}$ | $\begin{gathered} \text { PM565 } \\ \text { 0-2ET } \\ H \end{gathered}$ | $\begin{gathered} \text { PM567 } \\ 0-2 E T \\ H \end{gathered}$ | $\begin{gathered} \text { PM567 } \\ 5-2 E T \\ H \end{gathered}$ | $\begin{gathered} \geq \text { CPU } \\ \text { firm- } \\ \text { ware } \end{gathered}$ | $\begin{aligned} & \text { CM564 } \\ & \text { 0-2ETH } \end{aligned}$ |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| ABB netConfig | V3.0.0 | x | x | x | X | V3.6.0 | x |
| Online access with driver 3S TCP/IP BlkDrvTcp | V3.0.0 | X | X | X | X | V3.6.0 | X |
| Modbus TCP server | V3.0.3 | X | X | X | X | V3.6.0 | x |
| Modbus TCP client with POU ModTcpMast | V3.0.1 | x | X | x | x | V3.6.0 | X |
| UDP out of user program with library netBaseService.lib | V3.0.0 | X | X | X | X | V3.6.0 | X |
| UDP data exchange, Network variables | V3.0.0 | X | x | X | X | V3.6.0 | x |
| TCP/IP out of user program with library netBaseService.lib | V3.0.0 | X | X | X | X | V3.6.0 | X |
| Web server on PLC with web visualization | V3.0.0 | x | x | x | X | V3.6.0 | X |
| NTP/SNTP ((Simple) Network Time Protocol) client with 3 S licenced store package SNTPService.package. <br> Library container: SNTPService | V3.0.0 | X | X | X | X | V3.6.0 | X |
| IEC60870-5-104 control station incl. $2^{\text {nd }}$ connection and $2^{\text {nd }}$ port | V3.0.0 | X | X | X | X | V3.6.0 | X |
| IEC60870-5-104 substation incl. $2^{\text {nd }}$ port | V3.0.0 | X | X | X | X | V3.6.0 | X |
| FTP server | V3.0.0 | X | x | X | X | V3.6.0 | X |
| CODESYS network variables | V3.0.0 | X | X | X | X | V3.6.0 | X |
| OPC DA server | V3.0.0 | X | x | x | X | V3.6.0 | X |
| OPC UA server | V3.0.0 | X | X | X | X | V3.6.0 | X |
| ICMP - ping out of user project with POU ETHx_ICMP_PING | V3.0.0 | X | X | X | X | V3.6.0 | X |
| DHCP client | V3.1.0 | X | X | X | X | V3.6.0 | X |
| NTP/SNTP ((Simple) Network Time Protocol) client system solution <br> * Chapter 6.3.4.4.2 "(S)NTP client configuration" on page 1718 | V3.1.0 | X | X | X | X | V3.6.0 | X |
| NTP/SNTP ((Simple) Network Time Protocol) server system solution <br> ③ Chapter 6.3.4.4.3 "(S)NTP server configuration" on page 1720 | V3.1.0 | X | X | X | X | V3.6.0 | X |
| Maximum number of Input/output allowed variable on Ethernet for the protocol | V3.4.0 | $\begin{gathered} 2 \mathrm{kB} / 2 \\ \mathrm{kB} \end{gathered}$ | $\begin{gathered} 4 \mathrm{kB} / 4 \\ \mathrm{kB} \end{gathered}$ | $\begin{gathered} 5 \mathrm{kB} / 5 \\ \mathrm{kB} \end{gathered}$ | $\begin{gathered} 5 \mathrm{kB} / 5 \\ \mathrm{kB} \end{gathered}$ | V3.6.0 | Depen ds on used CPU |


| Description | $\begin{aligned} & \geq \text { CPU } \\ & \text { firm- } \\ & \text { ware } \end{aligned}$ | $\begin{gathered} \text { PM563 } \\ 0-2 \mathrm{ET} \\ \mathrm{H} \end{gathered}$ | $\begin{gathered} \text { PM565 } \\ \text { O-2ET } \\ H \end{gathered}$ | $\begin{gathered} \text { PM567 } \\ \text { 0-2ET } \\ H \end{gathered}$ | $\begin{gathered} \text { PM567 } \\ 5-2 E T \\ H \end{gathered}$ | $\begin{gathered} \geq \text { CPU } \\ \text { firm- } \\ \text { ware } \end{gathered}$ | $\begin{aligned} & \text { CM564 } \\ & \text { 0-2ETH } \end{aligned}$ |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| IEC 61850 (MMS server, GOOSE) ${ }^{2}$ ) | V3.1.0 | X | X | x | x | V3.6.0 | X |
| EthernetIP Scanner ${ }^{1,2}$ ) | AB <br> 2.4.1/ <br> FW <br> 3.4.1 | X | X | X | X | $\begin{gathered} \mathrm{AB} \\ 2.6 .0 / \\ \text { FW } \\ 3.6 .0 \end{gathered}$ |  |
| EthernetIP Adapter ${ }^{1,2}$ ) | $\begin{aligned} & \text { AB } \\ & 2.4 .1 / \\ & \text { FW } \\ & 3.4 .1 \end{aligned}$ | X | X | X | X | $\begin{gathered} \text { AB } \\ 2.6 .0 / \\ \text { FW } \\ 3.6 .0 \end{gathered}$ | X |
| KNX - Building communication ${ }^{2}$ ) | V3.2.x | X | X | X | X | V3.6.0 | X |
| BACnet-BC - Infrastructure communication ${ }^{2}$ ) | V3.3.1 | X | x | X | X | V3.6.0 | X |
| HTTPS - secure web server on PLC with CODESYS web visualization <br> « Chapter 6.3.4.7.3.2 "Secure web server" on page 1727 | V3.1.0 | X | X | X | X | V3.6.0 | X |
| WebVisu for data visualisation on web server HTML5 | V3.0.0 | X | X | X | X | V3.6.0 | X |
| FTPS - secure FTP <br> « Chapter 6.3.4.7.3.3 "Secure FTP" on page 1728 | V3.1.0 | X | X | X | X | V3.6.0 | X |
| Secure online access with driver 3S UDP BlkDrvUdp | V3.1.0 | X | X | X | X | V3.6.0 | X |
| Secure online access with driver 3S TCP/IP BlkDrvTcp | V3.1.0 | X | X | X | X | V3.6.0 | X |
| ICMP - ping out of user project with POU ETHx_ICMP_PING or EthIcmpPing (PLCopen style) | V3.1.0 | X | X | X | X | V3.6.0 | X |
| Modbus TCP client (master) with ModTcpMast (PLCopen style) | V3.1.0 | X | X | X | X | V3.6.0 | X |
| Remarks: <br> ${ }^{1}$ ): in preparation <br> ${ }^{2}$ ): feature is licensed |  |  |  |  |  |  |  |

## Default open Ethernet ports of PM56xx-2ETH

After startup without a PLC project the PM56xx-2ETH contains the following Ethernet ports and sockets:

| Protocol | Port |
| :--- | :--- |
| ABB NetConfig ${ }^{1}$ ) | UDP 24576 |
| Online access with driver 3S Tcp/lp BlkDrvTcp (no scan) | TCP 11740 |


| Protocol | Port |
| :--- | :--- |
| OPC UA server ${ }^{2}$ ) | TCP 4840 |
| Remarks: |  |
| 1): The port 24576 for ABB NetConfig protocol can be disabled via PLC configuration by <br> deleting the protocol node from configuration tree of Ethernet interfaces ETH1 and ETH2. <br> ${ }^{2}$ ): The port 4840 for OPC UA server is closed by default as of SystemFW V3.6.0. |  |

All other ports are closed by default.

## Overview of protocols, sockets and ports

| Protocol | Port | Sockets |
| :---: | :---: | :---: |
| ABB netConfig | 24576 | 1 permanent socket per interface |
| 3 gateway client (e.g. CODESYS) to gateway server | 1217 | 1 permanent socket |
| Online access with driver 3S block driver TCP/IP (no scan) | 11740 | 1 socket per connection + 1 listen |
| Modbus TCP server | 502 or configurable | 1 socket listen +1 socket per server connection, number of server connections is configurable in AB |
| Modbus TCP client with POU ModTcpMast | Random | 1 socket per connection with POU ModTcpMast |
| UDP out of user program with library SysLibSockets.lib | 1 ... 65535 | 1 socket per connection |
| TCP/IP out of user program with library SysLibSockets.lib | 1... 65535 | 1 socket per connection |
| Web server on PLC with web visualization | 80 | 1 listen and 1 per connection |
| NTP/SNTP client | 123 | 1 permanent socket |
| IEC60870-5-104 control station | Random | 1 per connection |
| IEC60870-5-104 substation | 2404 | 1 per connection |
| FTP server | Command port = 21 <br> Data active mode $=20$ <br> Data passive mode = random | 1 per session, max. 4 allowed |
| CODESYS network variables | 1202 | (UDP broadcast) |
| OPC DA server (default 3S block driver) | $\begin{aligned} & \begin{array}{l} \text { UDP }=1740 \\ \text { or } \\ \text { TCP/IP } \\ =11740 \end{array} \end{aligned}$ | 1 socket per connection |
| OPC UA server | 4840 | 1 permanent socket |
| ICMP - ping out of user project with POU ETHx_ICMP_PING DHCP | none | No socket |
| DHCP | 67 | 1 socket during startup |


| Protocol | Port | Sockets |
| :--- | :--- | :--- |
| NTP／SNTP（（Simple）Network Time Pro－ <br> tocol）client system solution <br> 乡 Chapter 6．3．4．4．2＂（S）NTP client config－ <br> uration＂on page 1718 | 123 | 1 permanent socket |
| NTP／SNTP（（Simple）Network Time Pro－ <br> tocol）server system solution <br> 乡 Chapter 6．3．4．4．3＂（S）NTP server config－ <br> uration＂on page 1720 | 123 | 1 permanent socket |
| HTTPS－secure web server on PLC with <br> CODESYS web visualization <br> 乡 Chapter 6．3．4．7．3．2＂Secure web server＂ <br> on page 1727 | 443 | 1 listen and 1 per connection |
| FTPS－secure FTP <br> 乡 Chapter 6．3．4．7．3．3＂Secure FTP＂ <br> on page 1728 | Command <br> port $=21$ <br> Data active <br> mode $=20$ <br> Data passive | 1 per session，max．4 allowed |
| mode $=$ <br> random | 11740 | 1 socket per connection＋1 <br> listen |
| Secure online access with driver 3S <br> TCP／IP BIkDrvTcp | No socket |  |
| ICMP－ping out of user project with <br> POU ETHx＿ICMP＿PING or EthlcmpPing <br> （PLCopen style） | None | Random <br> Modbus TCP client（master）with POU <br> ModTcpMast（PLCopen style） |

## Limitation of connections per protocol

The limitation for each CPU apply to all onboard Ethernet interfaces either directly on the CPU itself or also on all CM5640－2ETH interfaces that are added to this CPU．

| Protocol | PM5630 <br> －2ETH | PM565 <br> 0－2ETH | PM5670－ <br> 2ETH | PM5675－ <br> 2ETH | $\geq$ CPU <br> firm－ <br> ware |
| :--- | :--- | :--- | :--- | :--- | :--- |
| Modbus TCP server（e．g．for SCADA <br> access） | 30 | 100 | 100 | 100 | 3.0 .3 |
|  | 40 | 40 | 40 | 40 | 3.1 .0 |
| Modbus TCP client with POU ModTcpMast <br> （PLCopen style） | 30 | 15 | 25 | 50 | 50 |


| Protocol | $\begin{aligned} & \text { PM5630 } \\ & \text {-2ETH } \end{aligned}$ | $\begin{aligned} & \text { PM565 } \\ & \text { 0-2ETH } \end{aligned}$ | $\begin{aligned} & \text { PM5670- } \\ & \text { 2ETH } \end{aligned}$ | $\begin{array}{\|l\|} \hline \text { PM5675- } \\ \text { 2ETH } \end{array}$ | $\geq$ CPU firmware |
| :---: | :---: | :---: | :---: | :---: | :---: |
| FTP server | 4 | 4 | 4 | 4 | 3.1 .0 |
| Online access with driver 3S TCP/IP BlkDrvTcp | $\begin{aligned} & \mathrm{n} / \mathrm{a} \\ & 8 \end{aligned}$ | $\begin{array}{\|l\|} \hline 4 \\ 8 \end{array}$ | $\mathrm{n} / \mathrm{a}$ $8$ | n/a 8 | $\begin{aligned} & 3.0 .0 \\ & 3.1 .0 \end{aligned}$ |
| OPC DA server (number of connections) | $\begin{aligned} & \mathrm{n} / \mathrm{a} \\ & 8 \end{aligned}$ | $\begin{array}{\|l\|} \hline 4 \\ 8 \end{array}$ | $\begin{aligned} & \mathrm{n} / \mathrm{a} \\ & 8 \end{aligned}$ | $\begin{aligned} & \mathrm{n} / \mathrm{a} \\ & 8 \end{aligned}$ | $\begin{aligned} & 3.0 .0 \\ & 3.1 .0 \end{aligned}$ |
| OPC UA server (number of connections) | $\begin{aligned} & 50 \\ & 10 \end{aligned}$ | $\begin{aligned} & 50 \\ & 20 \end{aligned}$ | $\begin{aligned} & 50 \\ & 50 \end{aligned}$ | $\begin{aligned} & 50 \\ & 50 \end{aligned}$ | $\begin{array}{\|l\|} \hline 3.1 .0 \\ 3.4 .0 \end{array}$ |
| No. of free tags <br> + additional license for extension ${ }^{1}$ ) | 1.000 | 5.000 | 30.000 | 30.000 | 3.4.0 |
| min sampling rate (limit) | 500 ms | 100 ms | 50 ms | 50 ms | 3.4 .0 |
| Secure online access with driver 3S TCP/IP BIkDrvTcp | 8 | 8 | 8 | 8 | 3.1.0 |
| FTPS - secure FTP server | 4 | 4 | 4 | 4 | 3.1 .0 |
| HTTPS - Integrated webserver - number of connections | 4 | 8 | 12 | 12 | 3.4.0 |
| Ethernet/IP <br> Maximum number of Input/Output | $\begin{aligned} & 0.5 \\ & \mathrm{kB} / 0.5 \\ & \mathrm{kB} \end{aligned}$ | 0.5 <br> kB/0.5 kB | $\begin{aligned} & 0.5 \\ & \mathrm{kB} / 0.5 \\ & \mathrm{kB} \end{aligned}$ | $\begin{array}{\|l\|} \hline 0.5 \\ \mathrm{kB} / 0.5 \mathrm{kB} \end{array}$ | 3.4.1 |
| Remarks: <br> ${ }^{1}$ ): in preparation |  |  |  |  |  |

The PLC types PM5630-2ETH, PM5670-2ETH and PM5675-2ETH are available as of SystemFW 3.1.0.

## Default Ethernet configuration

| Module | IP Address | Netmask | Comment |
| :--- | :--- | :--- | :--- |
| PM5xx2-x-ETH | ETH: 192.168.0.10 | 255.255 .255 .0 |  |
| PM50x2-T-2ETH | ETH1: 192.168.0.10 | 255.255 .255 .0 | The Ethernet ports <br> must be configured <br> in different sub net- <br> works. |
| PM56xx-2ETH | ETH1: 192.168.0.10 | 255.255 .255 .0 | The Ethernet ports <br> must be configured <br> in different sub net- <br> works. |
| CM5640-2ETH | ETH2: 192.168.1.10 |  | The Ethernet ports <br> must be configured <br> in different sub net- <br> works. |
|  | 192.168 .11 .10, | 255.255 .255 .0 |  |
|  | SLO.168.12.10 <br> SLOT2: <br> 192.168 .21 .10, <br> 192.168 .22 .10 |  |  |

* Changing the IP addresses using the Automation Builder
* Changing the IP addresses using the CFG key


## Online access

3S TCP/IP block driver. This driver requires at least 2 sockets:

- 1x driver "BIkDrvTcp" on port 11740
- $1 x$ listen on port 11740 if PLC has established online connection


Online access can be established from:

- Automation Builder command 'Login' $\Leftrightarrow$ Chapter 6.4.1.21.3.7.2 "Command 'Login'" on page 2624
- CODESYS OPC DA server
- Panel CP600 series

Each established connection needs one socket. In addition one socket on port 11740 is listening.

1. Startup the PLC.
$\Rightarrow$ One socket on port 11740 (listen).
2. Login from Automation Builder via driver "BIKDrvTcp".
$\Rightarrow 2$ sockets on port 11740 ( 1 x online, 1 x listen)
3. Additional login out of OPC server with the same driver.
$\Rightarrow 3$ sockets on port 11740 ( $2 x$ online, $1 x$ listen)
4. Additional connect CP600 via driver "BlkDrvTcp".
$\Rightarrow 4$ sockets on port 11740 ( $3 x$ online, $1 x$ listen)

### 5.2.1.2.2 PM56xx-2ETH for AC500 V3 products

Features
Processor modules with onboard interfaces:

- PM5630-2ETH: processor module, memory 8 MB, with Ethernet support (onboard Ethernet) - 2 network interfaces RJ45, CAN and COM1 on the terminal base.
- PM5650-2ETH: processor module, memory 80 MB , with Ethernet support (onboard Ethernet) - 2 network interfaces RJ45, CAN and COM1 on the terminal base.
- PM5670-2ETH: processor module, memory 160 MB, with Ethernet support (onboard Ethernet) - 2 network interfaces RJ45, CAN and COM1 on the terminal base.
- PM5675-2ETH: processor module, $160 \mathrm{MB}, 8$ GB flash disk, with Ethernet support (onboard Ethernet) - 2 network interfaces RJ45, CAN and COM1 on the terminal base.
- XC version for use in extreme ambient conditions available


16 7-segment state displays with backlight
2 "Triangle" displays for "item"
3 "Square" displays for "state"
3 state LEDs
8 function keys
Slot for memory card
Label
8 Compartment for lithium battery TA521
9 Lithium battery TA521
10 Memory card
11 I/O bus for connection of I/O modules
12 Slot for processor module (processor module mounted on terminal base)

13 Slots for communication modules (multiple, depending on terminal base; unused slots must be covered with TA524)
14 Interface for CAN (5-pin terminal block, removable)
15 Power supply (5-pin terminal block, removable)
16 Serial interface COM1 (9-pin terminal block, removable)
17 RJ45 female connector for ETHERNET1 connection
18 RJ45 female connector for ETHERNET2 connection
19 DIN rail
${ }_{*}^{*_{*} \psi_{k}}$ Sign for XC version

## Short description

The processor modules are the central units of the control system AC500. The types differ in their performance (memory size, speed etc.). Each processor module must be mounted on a suitable terminal base.

The terminal base type (TB56xx) depends on the number of communication modules which are used together with the processor module.

Table 85: Comparison: TB56xx

| Processor module | PM5630 | PM5650 | PM5670 | PM5675 |
| :---: | :---: | :---: | :---: | :---: |
| Max. number of variables allowed for each communication module supported |  |  |  |  |
| Input variables | 4 kB | 4 kB | 5 kB | 5 kB |
| Output variables | 4 kB | 4 kB | 5 kB | 5 kB |
| Type of communication module supported |  |  |  |  |
| CM574-RS/RCOM - serial interface | No | No | No | No |
| CM5610-2RS - 2 serial interfaces | X | X | X | X |
| CM582-DP - PROFIBUS DP V0/V1 slave | X | X | X | X |
| CM592-DP - PROFIBUS DP V0/V1 master | X | X | X | X |
| CM579-ETHCAT - EtherCAT master | X | x | X | X |
| CM579-PNIO - PROFINET IO RT controller | X | X | X | X |
| CM589-PNIO - PROFINET IO RT device | X | X | X | X |
| CM589-PNIO-4 - PROFINET IO RT with 4 devices | X | X | X | X |
| CM597-ETH - Ethernet interface | No | No | No | No |
| CM5640-2ETH - 2 Ethernet interfaces | X | X | X | X |
| CM588-CN - CAN, CANopen slave | No | No | No | No |
| CM598-CN - CAN, CANopen master | $\begin{aligned} & \text { only CAN } \\ & \text { 2A/2B } \end{aligned}$ | $\begin{aligned} & \text { only CAN } \\ & \text { 2A/2B } \end{aligned}$ | $\begin{array}{\|l} \text { only CAN } \\ \text { 2A/2B } \end{array}$ | $\begin{aligned} & \text { only CAN } \\ & \text { 2A/2B } \end{aligned}$ |
| Type of AC500-S module supported |  |  |  |  |
| SM560-S - safety module | X | X | X | x |
| SM560-S-FD-1 - safety module with F-Device functionality for 1 PROFIsafe network | X | X | X | X |
| SM560-S -FD-4 - safety module with F-Device functionality for 1 PROFIsafe network | X | X | X | X |

All terminal bases (TB56xx) provide the same communication interfaces (ETH1, ETH2, CAN and COM1) $\Rightarrow$ Chapter 5.2.3.1.4 "Technical data" on page 383.
All other V3 processor modules can operate multiple communication modules via their communication module interface.
The communication modules are mounted on the left side of the processor module on the same terminal base.

On the right side of the processor module, up to 10 digital or analog I/O expansion modules can be connected to the I/O bus. Each I/O module requires a suitable terminal unit depending on the module type.

Terminal bases, terminal units, I/O modules, communication modules and accessories have their own technical descriptions.

Each processor module can be used as:

- Stand-alone processor module
- Stand-alone processor module with local I/Os
- Remote IO server
- Remote IO client

The processor modules are powered with 24 V DC.

## WARNING!

## Removal/Insertion under power

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.
Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.
Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.
The devices must not be opened when in operation. The same applies to the network interfaces.

## Connections

All terminals for connection are available on the terminal base. For information on connection and available interfaces see the descriptions for

- 帠 Chapter 5.2.3.1 "TB56xx for AC500 V3 products" on page 377.


Table 86: Combination of TB56xx-2ETH(-XC) and PM56xx(-XC)

| Processor module | PM5630 | PM5650 | PM5670 | PM5675 |
| :--- | :--- | :--- | :--- | :--- |
| TB5600-2ETH | 0 slot | 0 slot | 0 slot | 0 slot |
| TB5610-2ETH | 1 slot | 1 slot | 1 slot | 1 slot |
| TB5620-2ETH | 2 slots | 2 slots | 2 slots | 2 slots |
| TB5640-2ETH | - | 4 slots | 4 slots | 4 slots |
| TB5660-2ETH | - | - | 6 slots $\left.{ }^{1}\right)$ | 6 slots $\left.{ }^{1}\right)$ |

Remarks:
The slots can be used for connecting communication modules or AC500-S modules. Note that only one AC500-S module can be connected at one terminal base.
${ }^{1}$ ) PM567x must have an index $\geq$ C0.

## Storage elements

## Lithium battery

The processor modules are supplied without lithium battery. It must be ordered separately. The TA521 lithium battery is used for data (SRAM) and RTC buffering while the processor module is not powered.
${ }^{\text {r }}$ Chapter 6.8.2.3.2 "AC500 battery" on page 4445

The CPU monitors the discharge degree of the battery. A warning is issued before the battery condition becomes critical (about 2 weeks before). Once the warning message appears, the battery should be replaced as soon as possible.

The technical data, handling instructions and the insertion/replacement of the battery is described in detail in the chapter TA521 lithium battery ${ }^{\star}$ Chapter 5.2.8.2.3 "TA521-Battery" on page 1319.

Memory card AC500 processor modules are supplied without memory card. It must be ordered separately. The memory card can be used

- to read and write user files,
- to download a user program,
- for firmware updates,
- for program source code storage.

AC500 processor modules can be operated with and without memory cards. The processor module uses a standard file system (FAT). This allows standard card readers to read and write the memory cards.
$\Leftrightarrow$ Further information

LEDs, display and function keys on the front panel

\& Detailed information on using the LEDs, display and the function keys such as startup procedure and error coding

## Technical data

The system data of AC500 and S500 are applicable to the standard version. ${ }^{\mu}$ Chapter 5.1.2 "System data AC500" on page 166

The system data of AC500-XC are applicable to the XC version. \& Chapter 5.1.3 "System data AC500-XC" on page 169
Only additional details are therefore documented below.
The technical data are also applicable to the XC version.
Processor
module and
terminal base

| Parameter | Value |
| :--- | :--- |
| Connection of the supply voltage 24 V <br> DC at the terminal base of the processor <br> module | Removable 5-pin terminal block with spring con- <br> nection |
| Current consumption on 24 V DC | PM5630-2ETH: 110 mA |
| Min. typ. (module alone) | PM5650-2ETH: 120 mA <br>  <br>  <br> Max. typ. (all communication modules <br> and I/Os) <br> PM5675-2ETH: 140 mA <br> Num5630-2ETH: 850 mA <br> Number of slots for processor modules |
| Processor module interfaces at the ter- <br> minal bases TB56xx | PM5650-2ETH: 900 mA |
| Weight (processor module without ter- <br> minal base) | PM5670-2ETH: 950 mA |
| Mounting position | PM5675-2ETH: 950 mA |

Table 87: Comparison: PM56xx

| Processor module | PM5630 | PM565 <br> 0 | PM567 $0$ | PM5675 |
| :---: | :---: | :---: | :---: | :---: |
| Total maximum downloadable application size ${ }^{1)}$ | 9 MB | 84 MB | $\begin{array}{\|l\|} \hline 176 \\ \text { MB } \end{array}$ | 176 MB |
| Thereof user program code and data (dynamically allocated) | 2 MB | 8 MB | 32 MB | 32 MB |
| Thereof user webserver data max. | 6 MB | 72 MB | $\begin{aligned} & 128 \\ & \text { MB } \end{aligned}$ | 128 MB |
| Flash memory for User data |  |  |  |  |
| Remaining for all other usage (project save, infrastructure...) | 30 MB | $\begin{aligned} & 285 \\ & \text { MB } \end{aligned}$ | $\begin{aligned} & 643 \\ & \text { MB } \end{aligned}$ | 643 MB |
| Buffered (SRAM) | 256 kB | 256 kB | 1.5 MB | 1.5 MB |
| Thereof VAR retain persistent | 128 kB | 128 kB | $\begin{aligned} & 1024 \\ & k R \end{aligned}$ | 1024 kB |
| Thereof \%M memory (e.g. Modbus register) | 128 kB | 128 kB | 512 kB | 512 kB |
| Expandable memory | None | None | None | None |
| Integrated mass storage memory (FLASH) | None | None | None | 8 GB |
| Slot for pluggable memory card | X | X | X | X |



| Processor module |  | PM5630 | PM565 | PM567 | PM5675 |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  | Minimum cycle time configurable for cyclical task | 1 ms | 1 ms | 0,5 ms | $0,5 \mathrm{~ms}$ |
| User program protection by password |  | x (user management) |  |  |  |
| Internal interfaces for communication: |  |  |  |  |  |
| Serial interface COM1: |  |  |  |  |  |
|  | Physical link | Configurable for RS-232 or RS-485 ( $9.6 \mathrm{~kb} / \mathrm{s}, 19.2 \mathrm{~kb} / \mathrm{s}, 38.4 \mathrm{~kb} / \mathrm{s}, 57.6$ $\mathrm{kb} / \mathrm{s}$ and $115.2 \mathrm{~kb} / \mathrm{s}$ ) |  |  |  |
|  | Connection | Pluggable terminal block, spring connection |  |  |  |
|  | Usage | Serial ASCII communication,Modbus RTU |  |  |  |
| CAN interface: |  |  |  |  |  |
|  | Physical link | CAN 2A/2B (from $50 \mathrm{~kb} / \mathrm{s}$ to $1 \mathrm{Mb} / \mathrm{s}$ ) |  |  |  |
|  | Connection | Pluggable terminal block, spring connection |  |  |  |
|  | Usage | CANopen master communication, CAN 2A/2B, J1939 protocol, CAN sync |  |  |  |
|  | Max. number of variables allowed |  |  |  |  |
|  | Input variables | 2 kB | 4 kB | 5 kB | 5 kB |
|  | Output variables | 2 kB | 4 kB | 5 kB | 5 kB |
| Network interface ETH1, ETH2: |  |  |  |  |  |
|  | Usage | Ethernet |  |  |  |
|  | Physical link | 10/100 base-TX, configurable as internal switch or independent Interfaces |  |  |  |
|  | Connection | 2x RJ45 socket, provided on TB56xx-2ETH |  |  |  |
| LEDs, LCD display, function keys |  | RUN / STOP, status, diagnosis, settings |  |  |  |
| Number of timers |  | Unlimited |  |  |  |
| Number of counters |  | Unlimited |  |  |  |
| Programming languages: |  |  |  |  |  |
|  | Structured Text ST | x |  |  |  |
|  | Instruction list IL | x |  |  |  |
|  | Function Block Diagram FBD | x |  |  |  |
|  | Ladder Diagram LD | x |  |  |  |
|  | Sequential function chart SFC | x |  |  |  |
|  | Continuous function chart (CFC) | x |  |  |  |

[^6]Table 88: Combination of TB56xx-2ETH(-XC) and PM56xx (-XC)

| Processor module | PM5630 | PM5650 | PM5670 | PM5675 |
| :--- | :--- | :--- | :--- | :--- |
| TB5600-2ETH | 0 slot | 0 slot | 0 slot | 0 slot |
| TB5610-2ETH | 1 slot | 1 slot | 1 slot | 1 slot |
| TB5620-2ETH | 2 slots | 2 slots | 2 slots | 2 slots |
| TB5640-2ETH | - | 4 slots | 4 slots | 4 slots |
| TB5660-2ETH | - | - | 6 slots $\left.{ }^{1}\right)$ | 6 slots $\left.{ }^{1}\right)$ |

Remarks:
The slots can be used for connecting communication modules or AC500-S modules. Note that only one AC500-S module can be connected at one terminal base.
${ }^{1}$ ) PM567x must have an index $\geq$ C0.
Table 89: Comparison: TB56xx

| Processor module | PM5630 | PM5650 | PM5670 | PM5675 |
| :---: | :---: | :---: | :---: | :---: |
| Max. number of variables allowed for each communication module supported |  |  |  |  |
| Input variables | 4 kB | 4 kB | 5 kB | 5 kB |
| Output variables | 4 kB | 4 kB | 5 kB | 5 kB |
| Type of communication module supported |  |  |  |  |
| CM574-RS/RCOM - serial interface | No | No | No | No |
| CM5610-2RS - 2 serial interfaces | x | X | X | X |
| CM582-DP - PROFIBUS DP V0/V1 slave | x | X | X | X |
| CM592-DP - PROFIBUS DP V0/V1 master | X | X | X | X |
| CM579-ETHCAT - EtherCAT master | x | X | X | X |
| CM579-PNIO - PROFINET IO RT controller | X | X | X | X |
| CM589-PNIO - PROFINET IO RT device | X | X | X | X |
| CM589-PNIO-4 - PROFINET IO RT with 4 devices | x | X | X | X |
| CM597-ETH - Ethernet interface | No | No | No | No |
| CM5640-2ETH - 2 Ethernet interfaces | x | x | X | x |
| CM588-CN - CAN, CANopen slave | No | No | No | No |
| CM598-CN - CAN, CANopen master | $\begin{aligned} & \text { only CAN } \\ & \text { 2A/2B } \end{aligned}$ | $\begin{aligned} & \text { only CAN } \\ & \text { 2A/2B } \end{aligned}$ | only CAN 2A/2B | $\begin{aligned} & \text { only CAN } \\ & \text { 2A/2B } \end{aligned}$ |
| Type of AC500-S module supported |  |  |  |  |
| SM560-S - safety module | X | x | X | X |
| SM560-S-FD-1 - safety module with F-Device functionality for 1 PROFIsafe network | X | X | X | X |
| SM560-S -FD-4 - safety module with F-Device functionality for 1 PROFIsafe network | X | X | X | X |

Communication and onboard protocols

Table 90: OPC UA server / OPC DA server

| Processor module |  | PM5630 | PM5650 | PM5670 | PM5675 |
| :--- | :--- | :--- | :--- | :--- | :--- |
| OPC UA server |  | $x$ | $x$ | $x$ | $x$ |
|  | Number of free tags | 3.000 | 10.000 | 30.000 | 30.000 |


| Processor module |  | PM5630 | PM5650 | PM5670 | PM5675 |
| :--- | :--- | :--- | :--- | :--- | :--- |
|  | Number of connections | 10 | 20 | 50 | 50 |
|  | Min. sampling rate (limit) | 500 ms | 100 ms | 50 ms | 50 ms |
| OPC DA server AE |  | x | x | x | x |
|  | Number of connections | 8 | 8 | 8 | 8 |

Table 91: Modbus, Telecontrol

| Processor module |  | PM5630 | PM5650 | PM5670 |
| :--- | :--- | :--- | :--- | :--- |
| Modbus TCP client / server | x | x | PM 565 |  |
|  | Number of Modbus clients ModMast <br> in parallel on a CPU master (server) | 30 | 50 | 120 |
|  | Number of Modbus server in parallel <br> (e.g. for SCADA access) | 15 | 25 | 50 |
| IEC 60870-5-104 telecontrol protocol | x | x | 50 |  |
|  | Number of free tags | 1.000 | 5.000 | 120 |
|  | Control station (number of connec- <br> tions) | 5 | 10 | 20 |
|  | Sub-station (number of connections) | 5 | 10 | 20 |

## Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm

The dimensions are in mm and in brackets in inch.

## Ordering data

Processor modules for AC500
(Standard) V3 products

To enable better product availability into the production and to provide some new features, a revision 3 of the existing AC500 V3 processor module was necessary. The existing AC500 V3 processor module revision 2 with rubric R0278 will move to classic and will be replaced by compatible new AC500 V3 processor module revision 3 with rubric R0378.

For example: The processor module revision 3 PM5630-2ETH (1SAP131 000 R0378) replaces the existing processor module revision 2 PM5630-2ETH (1SAP 131000 R0278) and provides the same features or functionality of the previous ones.

Following points must be considered with the processor module revision 3:

- The processor module revision 3 (R037x) requires a new BootFW / CPUFW from V3.6.x and higher.
- It cannot be downgraded and used with lower FW versions than V3.6.0.
- The processor module revision $3(R 037 x)$ provides the same features as the processor module revision 2 (R027x) existing today and is fully compatible.
- An existing application using a processor module revision 2 (R027x) built with Automation Builder < 2.6 can run on a processor module revision 3 (R037x) but the application must be upgraded to at least $A B 2.6 .0$ or higher.

What must be done using a new processor module revision 3 (R037x)?

- On a new application?
- Just use the new processor module revision 3 (R037x)
- Use the latest Automation Builder software from 2.6.0 or higher.
- On an existing application using an Automation Builder software version smaller than 2.6.0?
- To use a new processor module revision 3 in an existing application (e.g., replacement of the processor module revision 2), the application must be upgraded to at least $A B$ 2.6.0 or higher.
- If several processor module (revision 3 and revision 2) are used within the same project, all the processor modules used in the same application must be upgraded to the FW Version V3.6.x and higher.

Table 92: Processor modules for AC500 (Standard) V3

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 131 000 R0278 <br> (processor module <br> revision 2) | PM5630-2ETH, processor module, <br> memory 8 MB, 24 V DC, memory card <br> slot, interface 1 RS-232/485, display, <br> 2 RJ45 independent onboard Ethernet <br> TCP/IP interfaces with Modbus TCP, <br> web server, IEC60870-5-104 or <br> selectable Ethernet based protocols | Classic <br> (replaced by processor <br> module revision 3) |
| 1SAP 131 000 R0378 <br> (processor module <br> revision 3) | PM5630-2ETH, processor module, <br> memory 8 MB, 24 V DC, memory card <br> slot, interface 1 RS-232/485, display, <br> 2 RJ45 independent onboard Ethernet <br> TCP/IP interfaces with Modbus TCP, <br> web server, IEC60870-5-104 or <br> selectable Ethernet based protocols | Active |
| 1SAP 331 000 R0278 <br> (processor module <br> revision 2) | PM5630-2ETH-XC, processor module, <br> memory 8 MB, 24 V DC, memory card <br> slot, interface 1 RS-232/485, display, <br> 2 RJ45 independent onboard Ethernet <br> TCP/IP interfaces with Modbus TCP, <br> web server, IEC60870-5-104 or <br> selectable Ethernet based protocols, <br> XC version | Classic <br> (replaced by processor <br> module revision 3) |
| 1SAP 341 000 R0278 <br> (processor module | PM5650-2ETH-XC, processor module, <br> memory 80 MB, 24 V DC, memory <br> card slot, interface 1 RS-232/485, <br> display, 2 RJ45 independent <br> onboard Ethernet TCP/IP interfaces <br> with Modbus TCP, web server, <br> IEC60870-5-104 or selectable <br> Ethernet based protocols, XC version | Classic |
| (replaced by processor |  |  |
| module revision 3) |  |  |


| Part no. | Description | Product life cycle phase *) |
| :---: | :---: | :---: |
| 1SAP 341000 R0378 <br> (processor module revision 3) | PM5650-2ETH-XC, processor module, memory $80 \mathrm{MB}, 24 \mathrm{~V}$ DC, memory card slot, interface 1 RS-232/485, display, 2 RJ45 independent onboard Ethernet TCP/IP interfaces with Modbus TCP, web server, IEC60870-5-104 or selectable Ethernet based protocols, XC version | Active |
| 1SAP 151000 R0278 <br> (processor module revision 2) | PM5670-2ETH, processor module, memory $160 \mathrm{MB}, 24 \mathrm{~V}$ DC, memory card slot, interface 1 RS-232/485, display, 2 RJ45 independent onboard Ethernet TCP/IP interfaces with Modbus TCP, web server, IEC60870-5-104 or selectable Ethernet based protocols | Classic <br> (replaced by processor module revision 3) |
| 1SAP 151000 R0378 (processor module revision 3) | PM5670-2ETH, processor module, memory $160 \mathrm{MB}, 24 \mathrm{~V}$ DC, memory card slot, interface 1 RS-232/485, display, 2 RJ45 independent onboard Ethernet TCP/IP interfaces with Modbus TCP, web server, IEC60870-5-104 or selectable Ethernet based protocols | Active |
| 1SAP 351000 R0278 (processor module revision 2) | PM5670-2ETH-XC, processor module, memory $160 \mathrm{MB}, 24 \mathrm{~V}$ DC, memory card slot, interface 1 RS-232/485, display, 2 RJ45 independent onboard Ethernet TCP/IP interfaces with Modbus TCP, web server, IEC60870-5-104 or selectable Ethernet based protocols, XC version | Classic <br> (replaced by processor module revision 3) |
| 1SAP 351000 R0378 <br> (processor module revision 3) | PM5670-2ETH-XC, processor module, memory $160 \mathrm{MB}, 24 \mathrm{~V}$ DC, memory card slot, interface 1 RS-232/485, display, 2 RJ45 independent onboard Ethernet TCP/IP interfaces with Modbus TCP, web server, IEC60870-5-104 or selectable Ethernet based protocols, XC version | Active |
| 1SAP 151500 R0278 <br> (processor module revision 2) | PM5675-2ETH, processor module, memory $160 \mathrm{MB}, 8$ GB flash disk, 24 V DC, memory card slot, interface 1 RS-232/485, display, 2 RJ45 independent onboard Ethernet TCP/IP interfaces with Modbus TCP, web server, IEC60870-5-104 or selectable Ethernet based protocols | Classic <br> (replaced by processor module revision 3) |
| 1SAP 151500 R0378 <br> (processor module revision 3) | PM5675-2ETH, processor module, memory 160 MB, 8 GB flash disk, 24 V DC, memory card slot, interface 1 RS-232/485, display, 2 RJ45 independent onboard Ethernet TCP/IP interfaces with Modbus TCP, web server, IEC60870-5-104 or selectable Ethernet based protocols | Active |


| Part no. | Description | Product life cycle phase *) |
| :---: | :---: | :---: |
| 1SAP 351500 R0278 (processor module revision 2) | PM5675-2ETH-XC, processor module, memory $160 \mathrm{MB}, 8$ GB flash disk, 24 V DC, memory card slot, interface 1 RS-232/485, display, 2 RJ45 independent onboard Ethernet TCP/IP interfaces with Modbus TCP, web server, IEC60870-5-104 or selectable Ethernet based protocols, XC version | Classic (replaced by processor module revision 3) |
| 1SAP 351500 R0378 (processor module revision 3) | PM5675-2ETH-XC, processor module, memory 160 MB, 8 GB flash disk, 24 V DC, memory card slot, interface 1 RS-232/485, display, 2 RJ45 independent onboard Ethernet TCP/IP interfaces with Modbus TCP, web server, IEC60870-5-104 or selectable Ethernet based protocols, XC version | Active |
| 1SAP 131000 R0379 | PM5630-MC-KIT: AC500, Machine Controller Kit with PM5630-2ETH, CM579-ETHCAT, TB5610-ETH, PS5611-MC | Active |
| 1SAP 141000 R0379 | PM5650-MC-KIT: AC500, Machine Controller Kit with PM5650-2ETH, CM579-ETHCAT, TB5610-ETH, PS5611-MC | Active |
| 1SAP 151000 R0379 | PM5670-MC-KIT: AC500, Machine Controller Kit with PM5670-2ETH, CM579-ETHCAT, TB5610-ETH, PS5611-MC | Active |
| Required accessories |  |  |
| 1SAP xxx 300 R0278 | TB56xx, terminal base <br> * Chapter 5.2.3.1 "TB56xx for AC500 V3 products" on page 377 | Active |
| Optional accessories |  |  |
| 1SAP 180300 R0001 | TA521, lithium battery <br> * Chapter 5.2.8.2.3 "TA521 - Battery" on page 1319 | Active |
| 1SAP 180100 R00xx | MC51xx, memory card <br> ⓢ Chapter 5.2.8.1.1"MC5102 Micro memory card with adapter" on page 1286 <br> ② Chapter 5.2.8.2.2 "MC5141 Memory card" on page 1315 | Active |

[^7]
### 5.2.2 Communication modules for AC500(-XC) processor modules

### 5.2.2. Features



AC500 communication modules are required for

- a connection to standard fieldbus systems and
- for integration into existing networks.

AC500 communication modules

- enable communication on different fieldbuses.
- are mounted on the left side of the processor module on the same terminal base.
- are directly powered via the internal communication module bus of the terminal base. A separate voltage source is not required.


## WARNING!

## Removal/Insertion under power

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.
Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.
Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.
The devices must not be opened when in operation. The same applies to the network interfaces.

The communication between the processor module and the communication modules takes place via the communication module bus, which is integrated in the terminal base. Depending on the used terminal base up to 6 communication modules can be connected.

- Hy Chapter 5.2.3.1 "TB56xx for AC500 V3 products" on page 377

There are no restrictions concerning which communication modules can be arranged for a processor module.
Within the AC500 control system, the communication modules can be used as

- bus master or
- slave.

It depends on the

- selected protocol,
- the functionality of the communication module and
- the several field buses and networks.

The following name extensions of the device names describe the supported field bus/protocol:

- CM5640-2ETH: Ethernet
- CM5x2-DP: PROFIBUS
- CM5x9-PNIO: PROFINET
- CM579-ETHCAT: EtherCAT
- CM598-CN: CANopen
- CM5610-2RS: 2 serial interfaces (COM1/COM2)

If a XC version of the device is available, for use in extreme ambient conditions (e.g. wider temperature and humidity range), this is indicated with a snowflake sign.

### 5.2.2.2 Compatibility of communication modules and communication interface modules

Table 93: Modbus TCP

| Communication <br> module | Communication <br> interface <br> module | I/O expansion <br> module <br> S500 | l/O expansion <br> module <br> S500-eCo | l/O expansion <br> module <br> S500-S | Applications |
| :--- | :--- | :--- | :--- | :--- | :--- |
| Onboard <br> Ethernet inter- <br> face | CI521-MODTCP <br> CI522-MODTCP | $x$ | $x$ | -- | high availability, <br> remote I/O |
| Onboard <br> Ethernet inter- <br> face | CI521-MODTCP <br> CI522-MODTCP | $x$ | -- | -- | hot-swap I/O |
| CM5640-2ETH | CI521-MODTCP <br> CI522-MODTCP | $x$ | $x$ | -- | high availability, <br> remote I/O |
| CM5640-2ETH | CI521-MODTCP <br> CI522-MODTCP | $x$ | -- | hot-swap I/O |  |

Table 94: PROFIBUS DP

| Communication <br> module | Communication <br> interface <br> module | I/O expansion <br> module <br> S500 | l/O expansion <br> module <br> S500-eCo | l/O expansion <br> module <br> S500-S | Applications |
| :--- | :--- | :--- | :--- | :--- | :--- |
| CM592-DP <br> master | CI541-DP <br> CI542-DP | x | x | -- | remote I/O |
| CM592-DP <br> master | CI541-DP <br> CI542-DP | x | -- | -- | hot-swap I/O |

Table 95: PROFINET IO RT

| Communication <br> module | Communication <br> interface <br> module | I/O expansion <br> module <br> S500 | I/O expansion <br> module <br> S500-eCo | I/O expansion <br> module <br> S500-S | Applications |
| :--- | :--- | :--- | :--- | :--- | :--- |
| CM579-PNIO <br> controller | CI501-PNIO <br> CI502-PNIO | $x$ | $x$ | $x$ | remote I/O, <br> safety I/O |
| CM579-PNIO <br> controller | CI501-PNIO <br> CI502-PNIO | $x$ | -- | hot-swap I/O |  |

Table 96: CANopen

| Communication <br> module | Communication <br> interface <br> module | l/O expansion <br> module <br> S500 | l/O expansion <br> module <br> S500-eCo | I/O expansion <br> module <br> S500-S | Applications |
| :--- | :--- | :--- | :--- | :--- | :--- |
| Onboard CAN <br> interface | $\mathrm{CI} 581-\mathrm{CN}$ <br> CI582-CN | -- | -- | remote I/O |  |

Table 97: EtherCAT

| Communication <br> module | Communication <br> interface <br> module | l/O expansion <br> module <br> S500 | l/O expansion <br> module <br> S500-eCo | I/O expansion <br> module <br> S500-S | Applications |
| :--- | :--- | :--- | :--- | :--- | :--- |
| CM579-ETHCAT <br> master | CI511-ETHCAT <br> CI512-ETHCAT | x | x | -- | remote I/O |

### 5.2.2.3 Technical data (Overview)

| Communication module | Field bus | Transmission rate | Field bus connector | Processor | Communication module interface | Current consumption from 24 V DC power supply at the terminal base of the CPU |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| CM5610-2RS | Serial (ASCII/ Modbus) | $\begin{aligned} & 9.6 \ldots 187.5 \\ & \mathrm{kBit} / \mathrm{s} \end{aligned}$ | $2 \times \mathrm{MC}$ 0.5/9-G-2.5, 9-pin, male | TI ARM Cortex-A9 | Dual-port RAM | Typ. 40 mA |
| CM579ETHCAT | EtherCAT | 10 or 100 MBit/s | $2 \times \mathrm{RJ45}$ | Hilscher NETX $100$ | Dual-port RAM, 16 kB | Typ. 85 mA |
| CM582-DP CM592-DP | PROFIBUS DP | $\begin{aligned} & 9.6 \mathrm{kBit} / \mathrm{s} . . .12 \\ & \mathrm{MBit} / \mathrm{s} \end{aligned}$ | D-sub, 9-pin, female, bended | Hilscher NETX $100$ | Dual-port RAM, 16 kB | Typ. 65 mA |
| CM598-CN | CANopen | 10 ... 1 MBit/s | $\begin{aligned} & \text { COMBICON } \\ & 2 \times 5 \text {-pin, } \\ & \text { bended } \end{aligned}$ | Hilscher NETX $100$ | Dual-port RAM, 16 kB | Typ. 65 mA |
| CM579-PNIO | PROFINET | $100 \mathrm{MBit} / \mathrm{s}$ | $2 \times \mathrm{RJ45}$ | Hilscher NETX $100$ | Dual-port RAM, 16 kB | Typ. 85 mA |
| $\begin{aligned} & \text { CM5640-2ET } \\ & \mathrm{H} \end{aligned}$ | 2 x Ethernet | $10 \text { or } 100$ MBit/s | $2 \times \mathrm{RJ} 45$ | TI ARM Cortex-A9 | Dual-port RAM | Typ. 47 mA |

### 5.2.2.4 Serial

### 5.2.2.4.1 CM5610-2RS with 2 serial interfaces

## Features



[^8]

Only use terminal blocks approved by ABB to avoid injury and damage to the module.

The communication modules with 2 serial interfaces are delivered with two 9-pin terminal blocks TA532 (1SAP 182000 R0001).

The terminal block listed in the ordering data is for spare part only if needed.

## Purpose

Communication module CM5610-2RS is equipped with 2 serial interfaces (COM1 and COM2) which can be used as serial interface extension of the proocessor module or for communication e.g. for communication via Modbus or ASCII.

Depending on the connection, the physical interface of COM1 and COM2 is either RS-232 or RS-485.

## Connections

## Serial interfaces

Pin assignment


## Bus cable for RS-485

Bus cable

| Bus line | 2 cores, twisted, with common shield |
| :--- | :--- |
| Construction | $>0.22 \mathrm{~mm}^{2}(24 \mathrm{AWG})$ |
| Conductor cross section | $>10$ per meter (symmetrically twisted) |
| Twisting rate | Polyethylene (PE) |
| Core insulation | $<100 \Omega / \mathrm{km}$ |
| Resistance per core | ca. $120 \Omega(100 \Omega \ldots 150 \Omega)$ |
| Characteristic impedance | $<55 \mathrm{nF} / \mathrm{km}$ (if higher, the max. bus length must be reduced) |
| Capacitance between the cores | $<120 \Omega 1 / 4 \mathrm{~W}$ at both line ends |
| Terminating resistors | Commonly used telephone cables with PE insulation and a <br> core diameter of $>0.8$ mm are usually sufficient. |
| Remarks | Cables with PVC core insulation and core diameter of <br> 0.8 mm can be used up to a length of approx. $250 \mathrm{~m} . \mathrm{In}$ <br> this case, the bus terminating resistor is approx. $100 \Omega$. |

## Cable lengths

The maximum possible cable length of a serial connection subnet within a segment depends on the transmission rate.

RS-232 (for point-to-point connection):

| Parameter | Value |
| :--- | :--- |
| Transmission rate | $9.6 \mathrm{kbit} / \mathrm{s}$ to $115.2 \mathrm{kbit} / \mathrm{s}$ |
| Maximum cable length | On request |

RS-485 (for point-to-point or bus connection):

| Parameter | Value |
| :--- | :--- |
| Transmission rate | $9.6 \mathrm{kbit} / \mathrm{s}$ to $115.2 \mathrm{kbit} / \mathrm{s}$ |
| Maximum cable length | On request |

## Bus termination (RS-485 only)

The line ends of the bus segment must be equipped with bus terminating resistors. Normally, these resistors are integrated in the interface connectors.


1 Term. P
2 RxD/TxD-P
3 RxD/TxD-N
4 Term. N
A Master at the bus line end, pull-up resistor and pull-down resistor activated, bus terminating resistor $120 \Omega$
B Slave within the bus line
C Slave at the bus line end, bus terminating resistor $120 \Omega$

## State LEDs

| LED |  | Color | State | Description |
| :---: | :---: | :---: | :---: | :---: |
| AHB CM5610 | PWR | Green | ON (light) | Voltage is present |
| $\square$ Pwr $\square$ RDY $\square$ Run $\square$ STA - ERR |  |  | OFF (dark) | Voltage is missing |
|  | RDY | Yellow | Programmable | Depends on user program |
|  | RUN | Green | Programmable | Depends on user program |
|  | STA | Yellow | Programmable | Depends on user program |
|  | ERR | Red | Programmable | Depends on user program |

## Technical data

The system data of AC500 and S500 are applicable to the standard version ${ }^{\xi}$ Chapter 5.1.2 "System data AC500" on page 166.

The system data of AC500-XC are applicable to the XC version $\Leftrightarrow$ Chapter 5.1.3 "System data AC500-XC" on page 169.
Only additional details are therefore documented below.

The technical data are also applicable to the XC version.

| Parameter |  | Value |
| :---: | :---: | :---: |
| Protocols |  |  |
|  | Serial COM ASCII | Yes |
|  | Modbus RTU (master and slave) | Yes |
| State LED |  |  |
|  | PWR | 1 |
|  | RUN | 1 |
|  | ERR | 1 |
|  | RDY | 1 |
|  | STA | 1 |
| Rotary switch |  |  |
|  | ADDR 00...FFhex | 2, for station identification |
| Serial interface 1 |  |  |
|  | Physical layer |  |
|  | RS232 / RS485 configurable | Isolated |
|  | Connection name | COM input of funktion blocks: $11,21, \ldots 61$ depending on which slot CM5610-2RS is plugged. Counting starts from right to left on left side of CPU. |
|  | Connection type |  |
|  | Terminal block | Pluggable spring terminals |
|  | Transmission rate | Configurable from 9.6 kbit/s to $115.2 \mathrm{kbit} / \mathrm{s}$ |
|  | Usage |  |
|  | Programming | No |
|  | Modbus RTU (master and slave) | Yes |
|  | Serial ASCII communication | Yes |
| Serial interface 2 |  |  |
|  | Physical layer |  |
|  | RS232 / RS485 configurable | Isolated |
|  | Connection name | COM input of funktion blocks:12, 22, ... 62 depending on which slot CM5610-2RS is plugged. Counting starts from right to left on left side of CPU. |
|  | Connection type |  |
|  | Terminal block | Pluggable spring terminals |
|  | Transmission rate | Configurable from 9.6 kbit/s to $115.2 \mathrm{kbit} / \mathrm{s}$ |
|  | Usage |  |
|  | Programming | No |
|  | Modbus RTU (master and slave) | Yes |
|  | Serial ASCII communication | Yes |
| Communication module interface/bus |  | Dual-port RAM (simulated) |


| Parameter |  | Value |
| :---: | :---: | :---: |
| Net weight [kg] |  | 0.12 kg |
| Power supply |  |  |
|  | Nominal supply voltage |  |
|  | Supply current at nominal supply voltage (depending on system architecture) | Typ. 40 mA per communication module at 24 V <br> With maximum load to both COM ports: 46 mA |
|  | Internal power supply source | Communication module bus |
| Hardware |  |  |
|  | Usable CPUs | PM56xx \& Chapter 5.2.1.2.2 "PM56xx-2ETH for AC500 V3 products" on page 313 |
|  | Usable terminal bases | TB56xx \& Chapter 5.2.3.1 "TB56xx for AC500 V3 products" on page 377 |

## Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm

The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 176 000 R0020 | CM5610-2RS, communication module, <br> 2 serial RS232/485 | Active |
| 1SAP 376 000 R0020 | CM5610-2RS-XC, communication <br> module, 2 serial RS232/485, XC ver- <br> sion | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

Table 98: Spare parts for communication modules with 2 serial interfaces

| Part no. | Description |
| :--- | :--- |
| 1SAP 182 000 R0001 | TA532: 9-pin terminal block set for communication module <br> CM5610-2RS(-XC), 10 pieces, spring type terminal |

The communication modules with 2 serial interfaces are delivered with two 9-pin terminal blocks TA532 (1SAP 182000 R0001).
The terminal block listed in the ordering data is for spare part only if needed.

### 5.2.2.5 CANopen

### 5.2.2.5.1 CM598-CN - CANopen master

## Features

- CANopen master 1 Mbit/s
- XC version for use in extreme ambient conditions available


15 LEDs for state display
2 Label
3 Communication interface, 5-pin, Combicon, male, removable plug with spring terminals
${ }_{*}^{*}+{ }_{*}^{*}$ Sign for XC version

## Purpose

Communication module CM598-CN enables communication over the CANopen field bus.
For use in extreme ambient conditions (e.g. wider temperature and humidity range), a special $X C$ version of the device is available.


The AC500 V3 CPUs only support CAN 2A/2B protocol on the communication module CM598-CAN $\Leftrightarrow$ Chapter 6.3.2.11.1.1.5 "Configuration of the protocols CAN 2.0 A / CAN 2.0 B" on page 1533.
Support of CANopen protocol is in preparation.

## Connections

Field bus interface

| Interface socket | 5-pin COMBICON |
| :--- | :--- |
| Transmission standard | ISO 11898, potential-free |
| Transmission protocol | CANopen (CAN), 1 Mbaud max. |
| Transfer rate (transmis- <br> sion rate) | $10 \mathrm{kbit} / \mathrm{s}, 20 \mathrm{kbit} / \mathrm{s}, 50 \mathrm{kbit} / \mathrm{s}, 100 \mathrm{kbit} / \mathrm{s}, 125 \mathrm{kbit} / \mathrm{s}, 250 \mathrm{kbit} / \mathrm{s}, 500$ <br> $\mathrm{kbit} / \mathrm{s}, 800 \mathrm{kbit} / \mathrm{s}$ and $1 \mathrm{Mbit} / \mathrm{s}$, |

Pin assignment Table 99: Pin assignment of the CANopen connector

| Interface |  | PIN | Signal | Description |
| :---: | :---: | :---: | :---: | :---: |
|  | ®我 | 1 | CAN_GND | CAN reference potential |
|  |  | 2 | CAN_L | Bus line, receive/transmit line, LOW |
|  | (1) 3 - (1) D | 3 | CAN_SHLD | Shield of the bus line |
|  |  | 4 | CAN_H | Bus line, receive/transmit line, HIGH |
|  | $\boxed{\square}$ | 5 | NC | Not connected |
| Terminal block removed | Terminal block inserted |  |  |  |

## NOTICE

U Unused connector!
Make sure that the terminal block is always connected to the terminal base or communication module, even if you do not use the interface.

Bus length The maximum possible bus length of a CAN network depends on bit rate (transmission rate) and cable type. The sum of all bus segments must not exceed the maximum bus length

| Bit Rate (speed) | Bus Length |
| :--- | :--- |
| $1 \mathrm{Mbit} / \mathrm{s}$ | 40 m |
| $800 \mathrm{kbit} / \mathrm{s}$ | 50 m |
| $500 \mathrm{kbit} / \mathrm{s}$ | 100 m |
| $250 \mathrm{kbit} / \mathrm{s}$ | 250 m |
| $125 \mathrm{kbit} / \mathrm{s}$ | 500 m |
| $50 \mathrm{kbit} / \mathrm{s}$ | 1000 m |

Types of bus For CANopen, only bus cables with characteristics as recommended in ISO 11898 are to be cables used. The requirements for the bus cables depend on the length of the bus segment. Regarding this, the following recommendations are given by ISO 11898:

| Length of seg- <br> ment $[\mathrm{m}]$ | Bus cable (shielded, twisted pair) |  |  | Max. transmis- <br> sion rate $[\mathrm{kbit} / \mathrm{s}]$ |
| :--- | :--- | :--- | :--- | :--- |
|  | Conductor <br> cross section <br> $\left[\mathrm{mm}^{2}\right]$ | Line resistance <br> $[\Omega / \mathrm{km}]$ | Wave impe- <br> dance $[\Omega]$ |  |
| $0 \ldots . .40$ | $0.25 \ldots 0.34 /$ <br> AWG23, AWG22 | 70 | 120 | 1000 at 40 m |
| $40 \ldots 300$ | $0.34 \ldots 0.60 /$ <br> AWG22, AWG20 | $<60$ | 120 | $<500$ at 100 m |
| $300 \ldots 600$ | $0.50 \ldots 0.60 /$ <br> AWG20 | $<40$ | 120 | $<100$ at 500 m |
| $600 \ldots 1000$ | $0.75 \ldots . .0 .80 /$ <br> AWG18 | $<26$ | 120 | $<50$ at 1000 m |

Bus terminating The ends of the data lines have to be terminated with a $120 \Omega$ bus terminating resistor. The bus resistors terminating resistor is usually installed directly at the bus connector.


Fig. 40: CANopen interface, bus terminating resistors connected to the line ends

| 1 | CAN_GND |
| :--- | :--- |
| 2 | CAN_L |
| 3 | Shield |
| 4 | CAN_H |
| 5 | Data line, shielded twisted pair |
| 6 | COMBICON connection, CANopen interface |



Fig. 41: DeviceNet interface, bus terminating resistors connected to the line ends

| 6 | DeviceNet power supply |
| :--- | :--- |
| 7 | COMBICON connection, DeviceNet interface |
| 8 | Data lines, twisted pair cables |
| 9 | red |
| 10 | black |
| 11 | white |
| 12 | blue |
| 13 | bare |

The grounding of the shield should take place at the switchgear $\Longleftrightarrow$ Chapter 5.1.2 "System data AC500" on page 166.

Table 100: Meaning of the diagnosis LEDs

| LED |  | Color | State | Description |
| :---: | :---: | :---: | :---: | :---: |
| AM Cms98 | PWR | Green | ON (light) | Power supply available |
| pmen |  |  | OFF (dark) | Power supply not available or defective hardware |
| Rum] | RDY | Yellow | ON | Boot procedure |
| $\xrightarrow{\text { Pr }}$ |  |  | Blinking | Boot failure |
|  |  |  | OFF | --- |
|  | RUN | Green | ON | Communication module is operational |
|  |  |  | Blinking | --- |
|  |  |  | OFF | Communication module is not operational |
|  | CAN-RUN | Green | ON | Operational: Device is in the OPERATIONAL state |
|  |  |  | Single Flash | Stopped: Device is in STOPPED state |
|  |  |  | Blinking | Pre-operational: Device is in the PREOPERATIONAL state |
|  |  |  | OFF | No communication or no power supply |
|  | CAN-ERR | Red | ON | CANopen bus is off |
|  |  |  | Single flash | Warning limit reached: At least one of the error counters of the CAN controller has reached or exceeded the warning level (too many error frames) |
|  |  |  | Double flash | Error control event: A guard event (NMT Slave or NMTmaster) or a heartbeat event (Heartbeat consumer) has occurred |
|  |  |  | OFF | No Error: Device is in working condition |
|  | CAN-RUN | Yellow | Blinking | No production data available, |
|  | CAN-ERR | Yellow | (synchronously) | no bus communication possible. |
| LED state | CAN-RUN | Green | Blinking | Firmware file transfers durin |
| firmware | CAN-ERR | Red | (synchronously) |  |
| update | CAN-RUN | Green | Blinking | Communication module writes the |
|  | CAN-ERR | Red | (alternately) | firmware file to the internal flash. <br> Do not power off the PLC! |

## Technical data

The system data of AC500 and S500 are applicable to the standard version $\Leftrightarrow$ Chapter 5.1.2 "System data AC500" on page 166.
The system data of AC500-XC are applicable to the XC version ${ }^{\aleph}$ Chapter 5.1.3 "System data AC500-XC" on page 169.
Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter | Value |
| :--- | :--- |
| Protocol | CANopen master (in preparation), CAN2A, <br> CAN2B |
| Transmission rate | 10 kbit/s to 1 Mbit/s |
| Ambient temperature | see: <br> System data AC500 Chapter 5.1.2 "System <br> data AC500" on page 166 <br> System Data AC500-XC Chapter 5.1.3 <br> "System data AC500-XC" on page 169 |
| Usable terminal bases | All TB5xx |
| Field bus connector | Pluggable connector COMBICON, 5-pin |
| Technology | Hilscher NETX 100 |
| Indicators | 5 LEDs |
| Internal power supply | Via the communication module interface of the <br> terminal base |
| Current consumption from 24 V DC power <br> supply at the Terminal Base of the CPU | Typ. 65 mA |
| Number of Slaves | Max. 126 |
| Number of receive/transmit PDOs | Max. 512 (respectively for receive and <br> transmit) |
| Total quantity of input and output data | Max. 3584 byte (respectively for input and <br> output) |
| Weight | Ca. 150 g |

## Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm

The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 173 800 R0001 | CM598-CN, communication module <br> CANopen master | Active |
| 1SAP 373 800 R0001 | CM598-CN-XC, communication <br> module CANopen master, XC version | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.2.6 EtherCAT

## Two alternatives are available for using EtherCAT:

- Onboard EtherCAT for the AC500-eCo
- EtherCAT for the AC500 with the CM579 communication module

Further information can be found here:
Onboard EtherCAT
*) EtherCAT configurator
« Communication schema onboard Ethernet/EtherCAT
\& Motion Control
EtherCAT
« CM579-ETHCAT
« Communication schema onboard Ethernet/EtherCAT
© Motion Control

### 5.2.2.6.1 CM579-ETHCAT - EtherCAT master

Features


15 LEDs for state display
22 rotary switches for address setting (not used)
3 Label
42 communication interfaces RJ45 (ETHCAT1 and ETHCAT2)

## Intended purpose

Communication module CM579-ETHCAT is for EtherCAT communication.
The comunication module is configured via the dual-port memory by means of a system configurator. The configuration is saved on a non-volatile Flash EPROM memory.
② Chapter 6.9.3.6.2 "CM579-ETHCAT" on page 4665

## Connections

Field bus interfaces

The EtherCAT communication module provides 2 RJ45 interfaces with the following pin assignment. The pin assignment is used for the EtherCAT slaves (communication interface modules CI5xy-ETHCAT) as well.

| Pin assignment | Interface | Pin | Signal | Description |
| :---: | :---: | :---: | :---: | :---: |
|  |  | 1 | TxD+ | Transmit data + |
|  |  | 2 | TxD- | Transmit data - |
|  |  | 3 | RxD+ | Receive data + |
|  |  | 4 | NC | Not connected |
|  |  | 5 | NC | Not connected |
|  |  | 6 | RxD- | Receive data - |
|  |  | 7 | NC | Not connected |
|  |  | 8 | NC | Not connected |
|  |  | Shield | Cable shield | Functional earth |

In corrosive environment, please protect unused connectors using the TA535 accessory.
Not supplied with this device.
(3) Further information about wiring and cable types

The EtherCAT network differentiates between input-connectors (IN) and outputconnectors (OUT):
At the EtherCAT slaves (communication interface modules), the ETH1-connector is IN and the ETH2-connector is OUT.
At the EtherCAT master (communication module), the ETHCAT1 connector has to be used. The ETHCAT2 connector is reserved for future extensions.

## State LEDs

The EtherCAT state is shown by the EtherCAT communication module's LEDs. Some LEDs are two-colored.

Table 101: Meaning of the diagnosis LEDs

| LED |  | Color | State | Description |
| :---: | :---: | :---: | :---: | :---: |
| ATH CM579 | PWR | Green | On | Power supply available |
|  |  |  | Blinking | --- |
|  |  |  | Off | Power supply not available or defective hardware |
|  | RDY | Yellow | On | Boot procedure |
|  |  |  | Blinking | Boot failure |
|  |  |  | Off | --- |
|  | RUN | Green | On | Communication module is operational |
|  |  |  | Blinking | --- |
|  |  |  | Off | Communication module is not operational |
|  | STA1 | Green | On | No bus error, communication running |
|  |  |  | Blinking | Establishing communication |
|  |  |  | Off | System error |
|  | STA2 | Red | On | Configuration error |
|  |  |  | Blinking | --- |
|  |  |  | Off | No error |
|  | STA1 | Yellow | Blinking | No production data available, |
|  | STA2 | Yellow | (synchronously) | no bus communication possible. |
| LED state during firmware update | STA1 | Green | Blinking (synchronously) | Firmware file transfers during communication module firmware update. |
|  | STA2 | Red |  |  |
|  | STA1 | Green | Blinking (alternately) | Communication module writes the firmware file to the internal flash. <br> Do not power off the PLC! |
|  | STA2 | Red |  |  |

The RJ45 Ethernet connector contains two LEDs showing the current Ethernet port connection state.

Table 102: Meaning of the diagnosis LEDs

| LED |  | Color | State | Description |
| :---: | :---: | :---: | :---: | :---: |
|  | ETHCAT1 LED "Link" | Green | On | Ethernet connection established |
|  |  |  | Off | No Ethernet connection |
|  | ETHCAT1 LED "RX/TX" | Yellow | On | Device sends/receives frames |
|  |  |  | Off | No Ethernet connection |
|  | ETHCAT2 LED "Link" | Green |  | Connector ETHCAT2 is not used |
|  | ETHCAT2 LED "RX/TX" | Yellow |  |  |

The system data of AC500 and S500 are applicable to the standard version ${ }^{*} \Rightarrow$ Chapter 5.1.2 "System data AC500" on page 166.
The system data of AC500-XC are applicable to the XC version ${ }^{\aleph}$ Chapter 5.1.3 "System data AC500-XC" on page 169.
Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter | Value |
| :---: | :---: |
| Internal Supply | Via the communication module interface of the terminal base |
| Protocol | EtherCAT |
| Field bus connector | $2 \times \mathrm{RJ45}$ (ETHCAT1 and ETHCAT2) |
| Technology | Hilscher NETX 100 |
| Transfer rate | 10/100 Mbit/s (full-duplex) |
| Transfer method | According to Ethernet II, IEEE 802.3 |
| Ethernet | 100 base-TX, internal switch, 2x RJ45 socket |
| Bus length (segment length max.) | 100 m at $100 \mathrm{Mbit} / \mathrm{s}$ |
| Indicators | 5 LEDs |
| Usable CPUs | PM56xx 出 Chapter 5.2.1.2.2 "PM56xx-2ETH for AC500 V3 products" on page 313 |
| Usable terminal bases | All TB56xx (not TB5600) $\Leftarrow$ Chapter 5.2.3.1 "TB56xx for AC500 V3 products" on page 377 |
| Ambient temperature | System data AC500 $\Rightarrow$ Chapter 5.1.2 "System data AC500" on page 166 <br> System Data AC500 XC $\stackrel{y}{ }{ }^{〔}$ Chapter 5.1.3 "System data AC500-XC" on page 169 |
| Current consumption from 24 V DC power supply at the terminal base of the CPU | Typ. 85 mA |
| Internal supply | Via the communication module interface of the terminal base |
| Number of slaves | Limited to 200 |
| Quantity of input and output data for a single slave | Max. 5760 bytes (respectively for input and output) |
| Total quantity of input and output data | Max. 5760 bytes (only valid for asynchronous operation, for synchronous operation the reachable values depends on the additional load of SoE, CoE and EoE, typical reachable values are 1024 bytes). |
| Supported protocols | RTC - Real-time cyclic protocol, class 1 <br> RTA - Real-time acyclic protocol |
| Acyclic services | - CoE upload <br> - CoE download (1500 bytes max.) <br> - Emergency |
| Min. bus cycle | 1 ms |
| Max. size of the bus configuration file | 2 MB |
| Weight | Ca. 170 g |

## Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm

The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 170 902 R0101 | CM579-ETHCAT, EtherCAT <br> communication module | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.2.7 Ethernet

### 5.2.2.7.1 CM5640-2ETH - Communication module Ethernet

## Features

- TCP/IP with 2 independent channels
- XC version for use in extreme ambient conditions available


15 LEDs for state display
22 rotary switches for station identification
3 Label
42 communication interfaces Ethernet RJ45
${ }_{x}^{*}+\ldots$ Sign for XC version

## Purpose

The communication module provides additional communication ports for the CPU. The Ethernet communication module is a port extender that provides two additional, separate, non-programmable Ethernet interfaces for the CPU processor module used.
Each communication module and port used results in a slightly higher CPU load on the CPU compared to using the same protocol via the onboard port, since the communication module and the transmission of the communication packets must be additionally processed by the CPU. The complete TCP/IP protocol and the application layers are supported.


- Each communication module port must be addressed via its IP address (not slot no.) and must be in a different subnet than the port(s) of the CPU.
- Each used communication module and port causes a slightly higher CPU load compared to using the same protocol onboard.

It is not possible to close a RSTP ring by using the two ports of the communication module.

Applications:

- TCP/IP for PC/Automation Builder (programming)
- UDP functions using SysSocket or NetBaseServices libraries
- Modbus on TCP/IP (Modbus on TCP/IP, client and server and all standard protocols from the PLC)

For use in extreme ambient conditions (e.g. wider temperature and humidity range), a special $X C$ version of the device is available.

## Connections

Field bus inter- The Ethernet communication module has 2 RJ45 interfaces: faces

Pin assignment

| Interface | Pin | Signal | Description |
| :---: | :---: | :---: | :---: |
|  | 1 | TxD+ | Transmit data + |
|  | 2 | TxD- | Transmit data - |
|  | 3 | RxD+ | Receive data + |
|  | 4 | NC | Not connected |
|  | 5 | NC | Not connected |
|  | 6 | RxD- | Receive data - |
|  | 7 | NC | Not connected |
|  | 8 | NC | Not connected |
|  | Shield | Cable shield | Functional earth |

In corrosive environment, please protect unused connectors using the TA535 accessory.
Not supplied with this device.

## ③) Further information about wiring and cable types

## State LEDs

The Ethernet state is shown by the Ethernet communication module's LEDs.

Table 103: Meaning of the diagnosis LEDs

| LED |  | Color | State | Description |
| :---: | :---: | :---: | :---: | :---: |
|  | PWR |  | On | Power supply available |
|  |  |  | Off | Power supply not available or defective module |
|  | RDY | Yellow | On | Boot procedure |
|  |  |  | Off | Awaiting boot |
|  |  |  | Blinking | Boot failure |
|  | RUN | Green | On | Communication module is operational |
|  |  |  | Off | Communication module is not operational |
|  | STA1 | Green | On | Communication module is activated |
|  |  |  | Off | Communication module awaiting activation |
|  | STA2 | Red | On | Diagnosis available not fetched by PLC |
|  |  |  | Off | No error |
| $\begin{array}{\|l} \hline \text { LED state } \\ \text { during } \\ \text { firmware } \\ \text { update } \end{array}$ | STA1 | Green | Blinking (alternately) | Communication module is in firmware update state. <br> Do not power off the PLC! |
|  | STA2 | Red |  |  |

The RJ45 Ethernet connector contains two LEDs showing the current Ethernet port connection state.

Table 104: Meaning of the diagnosis LEDs

| LED |  | $\begin{array}{\|l\|} \hline \text { Color } \\ \hline \text { Green } \\ \hline \end{array}$ | State | Description |
| :---: | :---: | :---: | :---: | :---: |
|  | ETH1 LED "Link" |  | On | Ethernet connection established |
|  |  |  | Off | No Ethernet connection |
|  | ETH1 LED "RX/TX" | Yellow | On | --- |
|  |  |  | Blinking | Device sends/receives frames |
|  |  |  | Off | --- |
|  | ETH2 LED "Link" | Green | On | Ethernet connection established |
|  |  |  | Off | No Ethernet connection |
|  | ETH2 LED "RX/TX" | Yellow | On | --- |
|  |  |  | Blinking | Device sends/receives frames |
|  |  |  | Off | --- |

## Technical data

The system data of AC500 and S500 are applicable to the standard version ${ }^{\aleph}$ Chapter 5.1.2 "System data AC500" on page 166.
The system data of AC500-XC are applicable to the XC version ${ }^{\mu}$ Chapter 5.1.3 "System data AC500-XC" on page 169.

Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter |  | Value |
| :---: | :---: | :---: |
| Protocols |  |  |
|  | Ethernet |  |
|  | TCP/P | Yes |
|  | UPD/IP | Yes |
|  | us TCP | Yes |
|  | (Ping) | Yes |
| DN |  | Yes |
|  | $P$ (email) | Yes |
|  |  | Yes |
| State LED |  |  |
|  | PWR | 1 |
|  | RUN | 1 |
|  | RDY | 1 |
|  | STA | 2 |
| Rotary switch |  |  |
|  | ADDR 00...FFhex | 2, for station identification |
| Ethernet interfaces |  |  |
| Physical layer <br> Connection name |  | 10/100Base-TX |
|  |  | ETH <Slot>1, ETH <Slot>2 |
| Connection type |  |  |
|  | Independent channels | Yes, 2 |
|  | Internal switch | No |
|  | Auto negotiation | Yes, not adjustable/changeable |
|  | MAC address | Not configurable |
|  | Transmission rate | 10/100 Mbit/s |
|  | Max. cable length | 100 m at $100 \mathrm{Mbit} / \mathrm{s}$ |
|  | Usage |  |
|  | Programming | 3S Online Protocol |
|  | TCP/IP | Yes |
|  | Modbus TCP | Yes |
|  | Online access | 3S Online Protocol |
| Communication module interface/bus |  | Dual-port RAM |
| Net weight |  | 0.12 kg |
| Power supply |  |  |
|  | Nominal supply voltage |  |
|  | Supply current at nominal supply voltage (depending on system architecture) | 47 mA per communication module at 24 V |
|  | Internal power supply source | Communication module bus |
| Hardware |  |  |


| Parameter | Value |
| :---: | :---: |
| Usable CPUs | PM56xx 乡y Chapter 5.2.1.2.2 "PM56xx-2ETH for AC500 V3 products" on page 313 |
| Usable terminal bases | TB56xx $\Leftarrow$ Chapter 5.2.3.1 "TB56xx for AC500 V3 products" on page 377 |

## Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm

The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 176 000 R0080 | CM5640-2ETH, <br> communication module Ethernet <br> TCP/IP with 2 RJ45 CPU port exten- <br> sions | Active |
| 1SAP 376 000 R0080 | CM5640-2ETH-XC, <br> communication module Ethernet <br> TCP/IP with 2 RJ45 CPU port exten- <br> sions, XC version | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.2.8 PROFIBUS

### 5.2.2.8.1 CM582-DP - PROFIBUS DP slave

## Features

- PROFIBUS DP slave $12 \mathrm{Mbit} / \mathrm{s}$
- Compatible with Automation Builder version starting from V2.0.2, and with CPU firmware version starting from V2.6
- XC version for use in extreme ambient conditions available


15 LEDs for state display
2 Label
3 Communication interface PROFIBUS DP D-sub, 9-pin, female
${ }_{x}^{*}+{ }_{*}^{*}$ Sign for XC version

## Purpose

Communication module CM582-DP enables communication over the PROFIBUS DP field bus.
For use in extreme ambient conditions (e.g. wider temperature and humidity range), a special XC version of the device is available.

## Connections

Field bus inter- The PROFIBUS DP connector (9-pin, female) has the following pin assignment: face

| Pin |  | Signal | Description |
| :---: | :---: | :---: | :---: |
|  | 1 | NC | Not connected |
|  | 2 | NC | Not connected |
|  | 3 | RxD/TxD-P | Receive/Transmit positive |
|  | 4 | CNTR-P | Control signal for repeater, positive |
|  | 5 | DGND | Reference potential for data exchange and +5 VI |
|  | 6 | VP | +5 V (power supply for the bus terminating resistors) |
|  | 7 | NC | Not connected |
|  | 8 | RxD/TxD-N | Receive/Transmit negative |
|  | 9 | NC | Not connected |

Table 105: Correlation of transmission rate, bit time and cable length:

| Tranmission rate in [kbit/s] | Bit time [tBit] | Max. cable length in [m] |
| :--- | :--- | :--- |
| 9.6 | $104.2 \mu \mathrm{~s}$ | 1200 |
| 19.2 | $52.1 \mu \mathrm{~s}$ | 1200 |
| 31.25 | $32 \mu \mathrm{~s}$ | 1200 |
| 45.45 | $22 \mu \mathrm{~s}$ | 1200 |
| 93.75 | $10.7 \mu \mathrm{~s}$ | 1200 |
| 187.5 | $5.3 \mu \mathrm{~s}$ | 1000 |
| 500 | $2 \mu \mathrm{~s}$ | 400 |
| 1500 | 666.7 ns | 200 |
| 3000 | 333.3 ns | 100 |
| 6000 | 166.7 ns | 100 |
| 12000 | 83.3 ns | 100 |

## State LEDs

The PROFIBUS state is shown by state LEDs.

Table 106: Meaning of the diagnosis LEDs

| LED |  | Color | State | Description |
| :---: | :---: | :---: | :---: | :---: |
|  | PWR | Green | ON (light) | Power supply available. |
|  |  |  | OFF (dark) | Power supply not available or defective hardware |
|  | RDY | Yellow | ON | Boot procedure |
|  |  |  | Blinking | Boot failure |
|  |  |  | OFF | --- |
|  | RUN | Green | ON | Communication module is operational |
|  |  |  | Blinking | --- |
|  |  |  | OFF | Communication module is not operational |
|  | STA | Green | ON | Communication to all slaves is established |
|  |  |  | Flashes cyclic | --- |
|  |  |  | Flashes noncyclic | No configuration or stack error |
|  |  |  | OFF | No communication |
|  | ERR | Red | Blinking | No data exchange to the master module or the cable is disconnected |
|  |  |  | OFF | No error |
|  | STA | Yellow | Blinking | No production data available, |
|  | ERR | Yellow | (synchronously) | no bus communication possible. |
| LED state during firmware update | STA | Green | Blinking <br> (synchronously) | Firmware file transfers during communication module firmware update. |
|  | ERR | Red |  |  |
|  | STA | Green | Blinking <br> (alternately) | Communication module writes the firmware file to the internal flash. <br> Do not power off the PLC! |
|  | ERR | Red |  |  |

## Technical data

The system data of AC500 and S500 are applicable to the standard version $\left.{ }^{*}\right\rangle$ Chapter 5.1.2 "System data AC500" on page 166.

The system data of AC500-XC are applicable to the XC version ${ }^{\&}$ Chapter 5.1.3 "System data AC500-XC" on page 169.

Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter | Value |
| :--- | :--- |
| State indication | By 5 LEDs <br> PWR, RDY, RUN, STA, ERR |
| Usable CPUs | PM57x, PM58x, PM59x |


| Parameter | Value |
| :--- | :--- |
| Usable terminal bases | All TB5xx |
| Current consumption from 24 V DC power <br> supply at the terminal base of the CPU | Typ. 65 mA |
| Internal power supply | Through the communication module interface <br> of the terminal base |
| Maximum number of cyclic input data | 244 bytes |
| Maximum number of cyclic output data | 244 bytes |
| Maximum number of acyclic read/write | 240 bytes |
| Configuration data | max. 244 bytes |
| Parameter data | 237 bytes application specific parameters |
| Processor | Hilscher NETX 100 |
| Internal RAM memory | 8 MB |
| External Flash memory | 8 MB |
| Weight | Ca. 150 g |

## Technical data of the interface

| Parameter | Value |
| :--- | :--- |
| Interface socket | 9-pin, D-sub socket |
| Transmission standard | EIA RS-485 acc. to IEC 61158/61784, poten- <br> tial-free |
| Transmission protocol | PROFIBUS DP |
| Transmission rate | $9.6 \mathrm{kbit} / \mathrm{s}$ up to $12 \mathrm{Mbit} / \mathrm{s}$ |

## Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm

The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 172 200 R0001 | CM582-DP, communication module <br> PROFIBUS DP slave, 12 MBit/s | Active |
| 1SAP 372 200 R0001 | CM582-DP-XC, <br> communication module PROFIBUS <br> DP slave, 12 MBit/s, XC version | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.2.8.2 CM592-DP - PROFIBUS DP master

## Features

- Master $12 \mathrm{Mbit} / \mathrm{s}$
- XC version for use in extreme ambient conditions available


15 LEDs for state display
2 Label
3 Communication interface PROFIBUS DP D-sub, 9-pin, female


## Purpose

Communication module CM592-DP enables communication over the PROFIBUS DP field bus.
For use in extreme ambient conditions (e.g. wider temperature and humidity range), a special XC version of the device is available.

## Connections

Field bus inter- The PROFIBUS DP connector (9-pin, female) has the following pin assignment: face

| Pin |  | Signal | Description |
| :---: | :---: | :---: | :---: |
|  | 1 | NC | Not connected |
|  | 2 | NC | Not connected |
|  | 3 | RxD/TxD-P | Receive/Transmit positive |
|  | 4 | CNTR-P | Control signal for repeater, positive |
|  | 5 | DGND | Reference potential for data exchange and +5 VI |
|  | 6 | VP | +5 V (power supply for the bus terminating resistors) |
|  | 7 | NC | Not connected |
|  | 8 | RxD/TxD-N | Receive/Transmit negative |
|  | 9 | NC | Not connected |

Table 107: Correlation of transmission rate, bit time and cable length:

| Tranmission rate in [kbit/s] | Bit time [tBit] | Max. cable length in [m] |
| :--- | :--- | :--- |
| 9.6 | $104.2 \mu \mathrm{~s}$ | 1200 |
| 19.2 | $52.1 \mu \mathrm{~s}$ | 1200 |
| 31.25 | $32 \mu \mathrm{~s}$ | 1200 |
| 45.45 | $22 \mu \mathrm{~s}$ | 1200 |
| 93.75 | $10.7 \mu \mathrm{~s}$ | 1200 |
| 187.5 | $5.3 \mu \mathrm{~s}$ | 1000 |
| 500 | $2 \mu \mathrm{~s}$ | 400 |
| 1500 | 666.7 ns | 200 |
| 3000 | 333.3 ns | 100 |
| 6000 | 166.7 ns | 100 |
| 12000 | 83.3 ns | 100 |

## State LEDs

The PROFIBUS state is shown by state LEDs.

Table 108: Meaning of the diagnosis LEDs

| LED |  | Color | State | Description |
| :---: | :---: | :---: | :---: | :---: |
|  | PWR |  | ON (light) | Power supply available |
|  |  |  | OFF (dark) | Power supply not available or defective hardware |
|  | RDY | Yellow | ON | Boot procedure |
|  |  |  | Blinking | Boot failure |
|  |  |  | OFF | --- |
|  | RUN | Green | ON | Communication module is operational |
|  |  |  | Blinking | --- |
|  |  |  | OFF | Communication module is not operational |
|  | STA | Green | ON | Communication to all slaves is established |
|  |  |  | Flashes cyclic | --- |
|  |  |  | Flashes noncyclic | No configuration or stack error |
|  |  |  | OFF | No communication |
|  | ERR | Red | ON | Communication to one/all slaves is disconnected |
|  |  |  | Flashes cyclic | Communication to at least one slave is disconnected |
|  |  |  | OFF | No error |
|  | STA | Yellow | Blinking | No production data available, |
|  | ERR | Yellow | (synchronously) | no bus communication possible. |
| LED state during firmware update | STA | Green | Blinking (synchronously) | Firmware file transfers during communication module firmware update. |
|  | ERR | Red |  |  |
|  | STA | Green | Blinking (alternately) | Communication module writes the firmware file to the user flash memory. <br> Do not power off the PLC! |
|  | ERR | Red |  |  |

## Technical data

The system data of AC500 and S500 are applicable to the standard version $\Leftrightarrow$ Chapter 5.1.2 "System data AC500" on page 166.
The system data of AC500-XC are applicable to the XC version ${ }^{\mu}$ Chapter 5.1.3 "System data AC500-XC" on page 169.
Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter | Value |
| :--- | :--- |
| State indication | By 5 LEDs <br> PWR, RDY, RUN, STA, ERR |
| Usable CPUs | PM57x, PM58x, PM59x |
| Usable terminal bases | All TB5xx |
| Current consumption from 24 V DC power <br> supply at the terminal base of the CPU | Typ. 65 mA |
| Internal power supply | Through the communication module interface <br> of the terminal base |
| Maximum number of supported slaves | 125 (DPV0/DPV1) |
| Maximum number of total cyclic input data | 5712 bytes |
| (Status information is separately managed) |  |, | Maximum number of total cyclic output data | 5760 bytes |
| :--- | :--- |
| Maximum number of cyclic intput data | 244 bytes/slave |
| Maximum number of cyclic output data | 244 bytes/slave |
| Configuration data | max. 244 bytes per slave |
| Parameterization data per slave | 7 bytes/slave standard parameters |
|  | 237 bytes/slave application specific parame- <br> ters |
| Maximum number of acyclic read/write | 240 bytes per slave and telegram |
| Processor | Hilscher NETX 100 |
| Internal RAM memory | 8 MB |
| External user flash memory | 8 MB |
| Weight | Ca. 150 g |


| Technical data <br> of the interface |  |  |
| :--- | :--- | :--- |
|  | Parameter | Value |
|  | Interface socket | 9-pin, D-sub socket |
|  | Transmission standard | EIA RS-485 acc. to IEC 61158/61784, poten- <br> tial-free |
|  | Transmission protocol | PROFIBUS DP |
| Transmission rate | 9.6 kbit/s up to $12 \mathrm{Mbit} / \mathrm{s}$ |  |
|  |  |  |

## Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm

The dimensions are in mm and in brackets in inch.

Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 173 200 R0001 | CM592-DP, communication module <br> PROFIBUS DP master, 12 MBit/s | Active |
| 1SAP 373 200 R0001 | CM592-DP-XC, <br> communication module PROFIBUS <br> DP master, 12 MBit/s, XC version | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.2.8.3 PROFIBUS connection details

Attachment plug 9-pin D-sub connector, male for the bus
cable

| Parameter | Value |
| :--- | :--- |
| Fastening torque | 0.4 Nm |

Assignment

| Pin | Signal | Description |
| :--- | :--- | :--- |
| 1 | Shield | Shielding, protective ground |
| 2 | not used | - |
| 3 | RxD/TxD-P | Reception / transmission line, <br> positive |
| 4 | CBTR-P | Control signal for repeater, <br> positive (optional) |
| 5 | DGND | Reference potential for data <br> lines and +5 V |
| 6 | VP | +5 V, supply voltage for bus <br> terminating resistors |
| 7 | not used | - |
| 8 | CNTR-N | Reception / transmission line, <br> negative |
| 9 |  | Control signal for repeater, <br> negative (optional) |

Bus cable

| Parameter | Value |
| :--- | :--- |
| Type | Twisted pair (shielded) |
| Characteristic impedance | $135 \Omega \ldots 165 \Omega$ |
| Cable capacitance | $<30 \mathrm{pF} / \mathrm{m}$ |
| Conductor diameter of the cores | $\geq 0.64 \mathrm{~mm}$ |
| Conductor cross section of the cores | $\geq 0.34 \mathrm{~mm}^{2}$ |
| Cable resistance per core | $\leq 55 \Omega / \mathrm{km}$ |
| Loop resistance (resistance of two cores) | $\leq 110 \Omega / \mathrm{km}$ |

Cable lengths The maximum possible cable length of a PROFIBUS subnet within a segment depends on the tranmission rate (baud rate).

| Transmission Rate | Maximum Cable Length |
| :--- | :--- |
| $9.6 / 19.2 / 93.75$ kBaud | 1200 m |
| 187.5 kBaud | 1000 m |
| 500 kBaud | 400 m |
| 1.5 MBaud | 200 m |
| 3 MBaud to 12 MBaud | 100 m |

Branch lines are generally permissible for transmission rates of up to $1500 \mathrm{kbit} / \mathrm{s}$. But in fact they should be avoided for transmission rates higher than $500 \mathrm{kbit} / \mathrm{s}$.

Bus terminating The line ends (of the bus segments) have to be terminated using bus terminating resistors resistors according to the drawing below. The bus terminating resistors are usually placed inside the bus connector.


Repeaters One bus segment can have up to 32 subscribers. Using repeaters a system can be expanded to up to 126 subscribers. Repeaters are also required for longer transfer lines. Please note that a repeater's load to the bus segment is the same as the load of a normal bus subscriber. The sum of normal bus subscribers and repeaters in one bus segment must not exceed 32 .


Fig. 42: Principle example for a PROFIBUS-DP system with repeaters (1500 kbit/s baud rate)

### 5.2.2.9 PROFINET

### 5.2.2.9.1 CM579-PNIO - PROFINET IO RT controller

## Features

- PROFINET IO controller
- Integrated 2-port switch
- XC version for use in extreme ambient conditions available


15 LEDs for state display
22 rotary switches for address setting (not used)
3 Label
42 communication interfaces RJ45 (PNIO1 and PNIO2)
${ }_{x}^{*}+{ }_{*}^{*}$ Sign for XC version

Intended purpose
The communication module is for PROFINET RT communication.
The PROFINET communication module includes an internal Ethernet switch. The connection to the Ethernet can be established directly to the communication module. An additional switch is not necessary.

The communication module is configured via the dual-port memory by means of a system configurator. The configuration is saved on a non-volatile Flash EPROM memory.

For use in extreme ambient conditions (e.g. wider temperature and humidity range), a special XC version of the device is available.

Functionality

| Parameter | Value |
| :--- | :--- |
| Protocol | PROFINET IO RT |
| Usable CPUs | PM57x, PM58x, PM59x <br> そ Chapter 5.2.1.2.2 "PM56xx-2ETH for <br> AC500 V3 products" on page 313 |
| Usable terminal bases | All TB56xx (not TB5600) 乡 Chapter 5.2.3.1 <br> "TB56xx for AC500 V3 products" on page 377 |
| Field bus connector | 2 RJ45 (PNIO1 and PNIO2), with integrated <br> 2-port switch |
| Internal supply | Via the communication module interface of the <br> terminal base |

## Connections

Field bus inter- The communication module provides 2 RJ45 interfaces.

## faces

| Pin assignment | Interface | Pin | Signal | Description |
| :---: | :---: | :---: | :---: | :---: |
|  |  | 1 | TxD+ | Transmit data + |
|  |  | 2 | TxD- | Transmit data - |
|  |  | 3 | RxD+ | Receive data + |
|  |  | 4 | NC | Not connected |
|  | $8$ | 5 | NC | Not connected |
|  |  | 6 | RxD- | Receive data - |
|  |  | 7 | NC | Not connected |
|  |  | 8 | NC | Not connected |
|  |  | Shield | Cable shield | Functional earth |

In corrosive environment, please protect unused connectors using the TA535 accessory.

Not supplied with this device.

## ③ Further information about wiring and cable types

## State LEDs

The PROFINET state is shown by the state LEDs.

Table 109: Meaning of the diagnosis LEDs

| LED |  | Color | State | Description |
| :---: | :---: | :---: | :---: | :---: |
| AHP CM579 | PWR | Green | On | Power supply available |
|  |  |  | Blinking | --- |
|  |  |  | Off | Power supply not available or defective hardware |
|  | RDY | Yellow | On | Boot procedure |
|  |  |  | Blinking | Boot failure |
|  |  |  | Off | --- |
|  | RUN | Green | On | Communication module is operational |
|  |  |  | Blinking | --- |
|  |  |  | Off | Communication module is not operational |
|  | STA1 | Red | On | Diagnosis alarm reported. At least one device is having a diagnosis alarm. In incorporation with STA2 PNIO: License fault. |
|  |  |  | Blinking | System error |
|  |  |  | Off | No system error |
|  | STA2 | Red | On | No connection; in incorporation with STA1 PNIO: license fault |
|  |  |  | Blinking | Configuration fault: some configured I/O modules are not connected |
|  |  |  | Off | No bus error, communication is running |
|  | STA1 | Yellow | Blinking | No production data available, |
|  | STA2 | Yellow | (synchronously) | no bus communication possible. |
| LED state during firmware update | STA1 | Green | Blinking <br> (synchronously) | Firmware file transfers during communication module firmware update. |
|  | STA2 | Red |  |  |
|  | STA1 | Green | Blinking (alternately) | Communication module writes the firmware file to the internal flash. <br> Do not power off the PLC! |
|  | STA2 | Red |  |  |

The RJ45 Ethernet connector contains two LEDs showing the current Ethernet port connection state.

Table 110: Meaning of the diagnosis LEDs

| LED |  | Color | State | Description |
| :---: | :---: | :---: | :---: | :---: |
|  | PNIO1 LED "Link" | Green | On | Ethernet connection established |
|  |  |  | Off | No Ethernet connection |
|  | PNIO1 LED "RX/TX" | Yellow | On | --- |
|  |  |  | Blinking | PROFINET device sends/receives frames |
|  |  |  | Off | --- |
|  | PNIO2 LED "Link" | Green | On | Ethernet connection established |


| LED |  | Color | State | Description |
| :--- | :--- | :--- | :--- | :--- |
|  |  |  | Off | No Ethernet connection |
|  |  | PNIO2 LED "RX/TX" | Yellow | On |
|  |  |  | Blinking | PROFINET device sends/receives <br> frames |
|  |  |  | Off | --- |

## Technical data

The system data of AC500 and S500 are applicable to the standard version ${ }^{\mu}$ Chapter 5.1.2 "System data AC500" on page 166.
The system data of AC500-XC are applicable to the XC version $\Leftrightarrow$ Chapter 5.1.3 "System data AC500-XC" on page 169.

Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter | Value |
| :--- | :--- |
| Protocol | PROFINET IO RT |
| Bus connection | 2 RJ45 (PNIO1 and PNIO2), with integrated 2- <br> port switch |
| Switch | Integrated |
| Technology | Hilscher NETX 100 |
| Transfer rate | 100 Mbit/s (full-duplex) |
| Transfer method | According to Ethernet II, IEEE 802.3 |
| Ethernet | 100 base-TX, internal switch, 2x RJ45 socket |
| Bus length (segment length max.) | 100 m |
| Indicators | 5 LEDs |
| Usable terminal bases | All TB56xx (not TB5600) „ Chapter 5.2.3.1 <br> "TB56xx for AC500 V3 products" on page 377 |
| Supported alarm types | Process alarm, diagnostic alarm, return of Sub- <br> Module, plug alarm, pull alarm |
| Alarm processing | Requires handling in application program |
| Current consumption from 24 V DC power <br> supply at the terminal base of the CPU | Typ. 85 mA |
| Internal supply | Via the communication module interface of the <br> terminal base |
| Weight | Ca. 170 g |
| Maximum number of remote I/O stations <br> connected | 128 |


| Parameter | Value |
| :--- | :--- |
| Supported protocols | RTC - real-time cyclic protocol, class 1 <br> RTA - real-time acyclic protocol <br> DCP - discovery and configuration protocol *) <br> CL-RPC - connectionless remote procedure call <br> Since revision FW 2.4.8.0 additionally <br> LLDP - link layer discovery protocol <br> SNMP - simply network management protocol <br> (SNMP v1) |
| Acyclic services | PNIO read / write (max. 1392 bytes per telegram, <br> max. 4096 bytes per service request) |
| Total quantity of input and output data | CM579-PNIO < FW 2.4.8.0 <br> CM579-PNIO = FW 2.4.8.0 <br> CM579-PNIO > FW 2.4.8.0 <br> 3072 bytes in total |
| 1024 bytes per I/O module <br> 4096 bytes in total |  |
| Min. bus cycle | 1440 bytes per I/O module <br> PM5630, PM5650: 4096 bytes in total <br> PM567x: 5120 bytes in total |
| Conformance class | 1 ms |

*) CM579-PNIO does not allow setting "Station name" by using PROFINET service "DCP SET NameOfStation".

## Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm

The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 170 901 R0101 | CM579-PNIO, PROFINET <br> communication module | Active |
| 1SAP 370 901 R0101 | CM579-PNIO-XC, PROFINET <br> communication module, XC version | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.2.9.2 CM589-PNIO(-4) - PROFINET IO RT with 4 devices

Features

- PROFINET IO device
- Integrated 2-port switch
- XC version for use in extreme ambient conditions available


15 LEDs for state display
22 rotary switches for setting the IO device identifier
3 Label
42 communication interfaces RJ45 (PNIO1 and PNIO2)
䊀 Sign for XC version

The communication module is for PROFINET RT communication.
The PROFINET communication module includes an internal Ethernet switch. The connection to the Ethernet can be established directly to the communication module. An additional switch is not necessary.
The communication module is configured via the dual-port memory by means of a system configurator. The configuration is saved on a non-volatile Flash EPROM memory.

For use in extreme ambient conditions (e.g. wider temperature and humidity range), a special $X C$ version of the device is available.

## CM589-PNIO(-4)

CM589-PNIO supports one application relation to communicate to one single PROFINET IO controller.

CM589-PNIO-4 supports 4 application relations to communicate to up to 4 PROFINET IO controllers in parallel using PROFINET Shared Device technology.

Functionality

| Parameter | Value |
| :--- | :--- |
| Protocol | PROFINET IO RT |
| Usable CPUs | PM57x, PM58x, PM59x <br> そ Chapter 5.2.1.2.2 "PM56xx-2ETH for <br> AC500 V3 products" on page 313 |
| Usable terminal bases | All TB56xx (not TB5600) 乡 Chapter 5.2.3.1 <br> "TB56xx for AC500 V3 products" on page 377 |
| Field bus connector | 2 RJ45 (PNIO1 and PNIO2), with integrated <br> 2-port switch |
| Internal supply | Via the communication module interface of the <br> terminal base |

## Connections

Field bus inter- The PROFINET communication module provides 2 RJ45 interfaces: faces

| Pin assignment | Interface | Pin | Signal | Description |
| :---: | :---: | :---: | :---: | :---: |
|  |  | 1 | TxD+ | Transmit data + |
|  |  | 2 | TxD- | Transmit data - |
|  |  | 3 | RxD+ | Receive data + |
|  |  | 4 | NC | Not connected |
|  |  | 5 | NC | Not connected |
|  |  | 6 | RxD- | Receive data - |
|  |  | 7 | NC | Not connected |
|  |  | 8 | NC | Not connected |
|  |  | Shield | Cable shield | Functional earth |

In corrosive environment, please protect unused connectors using the TA535 accessory.

Not supplied with this device.
(3) Further information about wiring and cable types

## Addressing

The module reads the position of the rotary switches only during power-up, i.e. changes of the switch position during operation will have no effect until the next module initialization.

## State LEDs

The PROFINET state is shown by the state LEDs.

Table 111: Meaning of the diagnosis LEDs

| LED |  | Color | State | Description |
| :---: | :---: | :---: | :---: | :---: |
| AHP CM589 | PWR | Green | On | Power supply available |
| PWR [ <br> Ror 1 <br> RuN $[$ <br> STA1 $\mathbb{D}$ PNo <br> STR $\quad$ PNO |  |  | Blinking | --- |
|  |  |  | Off | Power supply not available or defective hardware |
|  | RDY | Yellow | On | Boot procedure |
|  |  |  | Blinking | Boot failure |
|  |  |  | Off | --- |
|  | RUN | Green | On | Communication module is operational |
|  |  |  | Blinking | --- |
|  |  |  | Off | Communication module is not operational |
|  | STA1 | Red | On | System error; watchdog timeout |
|  |  |  | Blinking |  |
|  |  |  | Off | No system error |
|  | STA2 | Red | On | No connection; no configuration |
|  |  |  | Blinking | No data exchange |
|  |  |  | Off | No bus error, communication is running |
|  | STA1 | Yellow | Blinking | No production data available, |
|  | STA2 | Yellow | (synchronously) | no bus communication possible. |
| LED state during firmware update | STA1 | Green | Blinking <br> (synchronously) | Firmware file transfers during communication module firmware update. |
|  | STA2 | Red |  |  |
|  | STA1 | Green | Blinking (alternately) | Communication module writes the firmware file to the internal flash. <br> Do not power off the PLC! |
|  | STA2 | Red |  |  |

The RJ45 Ethernet connector contains two LEDs showing the current Ethernet port connection state.

Table 112: Meaning of the diagnosis LEDs

| LED |  | Color | State | Description |
| :---: | :---: | :---: | :---: | :---: |
|  | PNIO1 LED "Link" | Green | On | Ethernet connection established |
|  |  |  | Off | No Ethernet connection |
|  | PNIO1 LED "RX/TX" | Yellow | On | PROFINET device sends/receives frames |
|  |  |  | Blinking | PROFINET device sends/receives frames |
|  |  |  | Off | --- |
|  | PNIO2 LED "Link" | Green | On | Ethernet connection established |
|  |  |  | Off | No Ethernet connection |


| LED |  |  |  | Color |
| :--- | :--- | :--- | :--- | :--- |
| State | Description |  |  |  |
|  | PNIO2 LED "RX/TX" | Yellow | On | PROFINET device sends/receives <br> frames |
|  |  |  | Blinking | PROFINET device sends/receives <br> frames |
|  |  | Off | --- |  |

## Technical data

The system data of AC500 and S500 are applicable to the standard version ${ }^{*} \Rightarrow$ Chapter 5.1.2 "System data AC500" on page 166.
The system data of AC500-XC are applicable to the XC version ${ }^{\mu}$ Chapter 5.1.3 "System data AC500-XC" on page 169.

Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter | Value |
| :--- | :--- |
| Protocol | PROFINET IO RT |
| Bus connection | 2 RJ45 (PNIO1 and PNIO2), with integrated <br> 2-port switch |
| Switch | Integrated |
| Technology | Hilscher NETX 100 |
| Transfer rate | 100 Mbit/s (full-duplex) |
| Transfer method | According to Ethernet II, IEEE 802.3 |
| Ethernet | 100 base-TX, internal switch, 2x RJ45 socket |
| Bus length (segment length max.) | 100 m |
| Indicators | 5 LEDs |
| Usable terminal bases | All TB56xx (not TB5600) \& Chapter 5.2.3.1 <br> "TB56xx for AC500 V3 products" on page 377 |
| Supported alarm types | Process alarm, diagnostic alarm, return of <br> SubModule, plug alarm, pull alarm |
| Current consumption from 24 V DC power <br> supply at the terminal base of the CPU | Typ. 85 mA |
| Internal supply | Via the communication module interface of the <br> terminal base |
| Setting of the I/O device identifier | With 2 rotary switches at the front side of the <br> module |
| Weight | Ca. 170 g |


| Parameter | Value |
| :--- | :--- |
| Supported protocols | RTC - real-time cyclic protocol, class 1 |
|  | RTA - real-time acyclic protocol |
|  | DCP - discovery and configuration protocol *) |
|  | CL-RPC - connectionless remote procedure <br> call |
|  | LLDP - link layer discovery protocol |
|  | SNMP - simply network management protocol |
|  | MRP - MRP Client |

*) Setting NameOfStation via service "DCP SET NameOfStation" is enabled only if rotary switches are adjusted to position "00".

## Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm
$\square$

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 172 900 R0011 | CM589-PNIO, PROFINET <br> communication module | Active |
| 1SAP 372 900 R0011 | CM589-PNIO-XC, PROFINET <br> communication module, XC version | Active |
| 1SAP 172 900 R0111 | CM589-PNIO-4, PROFINET <br> communication module | Active |
| 1SAP 372 900 R0111 | CM589-PNIO-4-XC, PROFINET <br> communication module, XC version | Active |

${ }^{\text {*) }}$ ) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.3 Terminal bases for AC500(-XC) processor modules and communication modules

### 5.2.3.1 TB56xx for AC500 V3 products

### 5.2.3.1.1 Features

- TB5600-2ETH: 1 processor module, with network interface 2 Ethernet RJ45, 1 CAN and 1 COM1
- TB5610-2ETH: 1 processor module, 1 communication module, with network interface 2 Ethernet RJ45, 1 CAN and 1 COM1
- TB5620-2ETH: 1 processor module, 2 communication modules, with network interface 2 Ethernet RJ45, 1 CAN and 1 COM1
- TB5640-2ETH: 1 processor module, 4 communication modules, with network interface 2 Ethernet RJ45, 1 CAN and 1 COM1
- TB5660-2ETH: 1 processor module, 6 communication modules, with network interface 2 Ethernet RJ45, 1 CAN and 1 COM1
- XC version for use in extreme ambient conditions available


Table 113: Combination of TB56xx-2ETH(-XC) and PM56xx (-XC)

| Processor module | PM5630 | PM5650 | PM5670 | PM5675 |
| :--- | :--- | :--- | :--- | :--- |
| TB5600-2ETH | 0 slot | 0 slot | 0 slot | 0 slot |
| TB5610-2ETH | 1 slot | 1 slot | 1 slot | 1 slot |
| TB5620-2ETH | 2 slots | 2 slots | 2 slots | 2 slots |
| TB5640-2ETH | - | 4 slots | 4 slots | 4 slots |
| TB5660-2ETH | - | - | 6 slots $\left.{ }^{1}\right)$ | 6 slots $\left.{ }^{1}\right)$ |

## Remarks:

The slots can be used for connecting communication modules or AC500-S modules. Note that only one AC500-S module can be connected at one terminal base.
${ }^{1}$ ) PM567x must have an index $\geq$ C0.
The following figure shows the TB5620-2ETH as example.


I/O bus (10-pin, female) to connect the I/O terminal units
One available slot for the processor module
Slots for communication modules
Interface for CAN (5-pin terminal block, removable)
Power supply ( 5 -pin terminal block, removable)
Serial interface COM1 (9-pin terminal block, removable)
RJ45 female connector for Ethernet connection
Holes for screw mounting

XC version
$\mathbf{X C}=\mathrm{e}$ Xtreme Conditions


## Extreme conditions

Terminal bases for use in extreme ambient conditions have no ${ }_{*_{+}^{*}}^{{ }^{*}}$ sign for XC version.

The figure 3 in the Part no. 1SAP3... (label) identifies the XC version.

### 5.2.3.1.2 Short description

Terminal bases TB56xx are used as sockets for processor modules PM56xx and communication modules.
Up to 10 I/O terminal units for I/O expansion modules can be added to these terminal bases.
The terminal bases have slots for one processor module and for communication modules as well as terminals and interfaces for power supply, expansion and networking.

Table 114: Combination of TB56xx-2ETH $(-X C)$ and PM56xx $(-X C)$

| Processor module | PM5630 | PM5650 | PM5670 | PM5675 |
| :--- | :--- | :--- | :--- | :--- |
| TB5600-2ETH | 0 slot | 0 slot | 0 slot | 0 slot |
| TB5610-2ETH | 1 slot | 1 slot | 1 slot | 1 slot |
| TB5620-2ETH | 2 slots | 2 slots | 2 slots | 2 slots |
| TB5640-2ETH | - | 4 slots | 4 slots | 4 slots |
| TB5660-2ETH | - | - | 6 slots $\left.{ }^{1}\right)$ | 6 slots $\left.{ }^{1}\right)$ |

Remarks:
The slots can be used for connecting communication modules or AC500-S modules. Note that only one AC500-S module can be connected at one terminal base.
${ }^{1}$ ) PM567x must have an index $\geq \mathrm{C} 0$.

## NOTICE!

## Risk of malfunctions!

Unused slots for communication modules are not protected against accidental physical contact.

- Unused slots for communication modules must be covered with dummy communication modules to achieve IP20 rating ${ }^{\star} \Rightarrow$ Chapter 5.2.8.2.4 "TA524 - Dummy communication module" on page 1323.
- I/O bus connectors must not be touched during operation.


### 5.2.3.1.3 Connections

I/O bus
The I/O bus is the I/O data bus for the I/O modules. Through this bus, I/O and diagnosis data are transferred between the processor module and the I/O modules. Up to 10 I/O modules can be added.
② Chapter 4.2.3 "I/O bus - Communication within the PLC" on page 97
« $\leadsto$ Chapter 5.2.4.2 "I/O bus - Data transfer" on page 387

## Power supply

The supply voltage of 24 V DC is connected to a removable 5-pin terminal block. L+/M exist twice. It is therefore possible to feed e.g. external sensors (up to 8 A max. with $1.5 \mathrm{~mm}^{2}$ conductor) via these terminals, when the ambient temperature never exceeds $+60^{\circ} \mathrm{C}$.

Pin assignment

| Pin Assignment |  | Label | Function | Description |
| :---: | :---: | :---: | :---: | :---: |
| Terminal block removed | Terminal block inserted | L+ | +24 V DC | Positive pin of the power supply voltage |
|  |  | L+ | +24 V DC | Positive pin of the power supply voltage |
|  |  | M | 0 V | Negative pin of the power supply voltage |
|  |  | M | 0 V | Negative pin of the power supply voltage |
|  |  | $\stackrel{1}{=}$ | FE | Functional earth |

## NOTICE!

## Risk of damaging the PLC due to improper voltage levels!

- Never exceed the maximum tolerance values for process and supply voltages.
- Never fall below the minimum tolerance values for process and supply voltages.
Observe the system data ${ }^{\circledR}$ Chapter 5.1.2 "System data AC500" on page 166 and the technical data of the module used.


## NOTICE!

## Risk of malfunction!

To ensure reliability and proper functionality of processor modules below index C0, the supply voltage must ramp-up from 0 V to 24 V within max. 2.5 s .

## NOTICE!

Risk of damaging the terminal base and power supply!
Short circuits might damage the terminal base and power supply.
Make sure that the four clamps $L+$ and $M$ (two of each) are not wrongly connected (e. g. +/- of power supply is connected to both L+/L+ or both M/M).

## NOTICE!

Risk of damaging the terminal base!
Terminal base can be damaged by connecting the power supply terminal block (L+/M) to COM1.

Make sure that the COM1 terminal block is always connected to the terminal base even if you do not use COM1 to prevent this.

## NOTICE!

Risk of damaging the terminal base!
Excessive current might damage the clamp and terminal base.
Make sure that the current flowing through the removable clamps never exceeds 8 A (with $1.5 \mathrm{~mm}^{2}$ conductor).

## Serial interface COM1

The serial interface COM1 is connected to a removable 9-pin terminal block.
From firmware version V3.1 it is configurable for RS-232 or RS-485 (V3.0 RS-232 only).

Pin assignment (RS-485 I RS-232)

|  |  | Pin | Signal | Interface | Description |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | 1 | Terminator P | RS-485 | Terminator P |
|  |  | 2 | RxD/TxD-P | RS-485 | Receive/Transmit, positive |
|  |  | 3 | RxD/TxD-N | RS-485 | Receive/Transmit, negative |
|  |  | 4 | Terminator N | RS-485 | Terminator N |
|  |  | 5 | RTS | RS-232 | Request to send (output) |
| Terminal block removed | Terminal block inserted | 6 | TxD | RS-232 | Transmit data (output) |
|  |  | 7 | SGND | Signal Ground | Signal Ground |
|  |  | 8 | RxD | RS-232 | Receive data (input) |
|  |  | 9 | CTS | RS-232 | Clear to send (input) |

## - NOTICE! <br> Unused connector! <br> Make sure that the terminal block is always connected to the terminal base or communication module, even if you do not use the interface.

## Ethernet interface

This interface is the connection to a processor module with onboard Ethernet e.g. PM56xx-2ETH.

TB56xx-2ETH for processor modules PM56xx-2ETH provide 2 independent Ethernet interfaces.

The two Ethernet interfaces can be configured as independent interfaces or with switch functionality.

In case of two independent interfaces they must be configured to different subnets.

For structured Ethernet cabling only use cables according to TIA/EIA-568-A, ISO/IEC 11801 or EN 50173.

Pin assignment

| Interface | Pin | Signal | Description |
| :---: | :---: | :---: | :---: |
|  | 1 | TxD+ | Transmit data + |
|  | 2 | TxD- | Transmit data - |
|  | 3 | RxD+ | Receive data + |
|  | 4 | NU | Not used |
|  | 5 | NU | Not used |


| Interface | Pin | Signal | Description |
| :--- | :--- | :--- | :--- |
|  | 6 | RxD- | Receive data - |
|  | 7 | NU | Not used |
|  | 8 | NU | Not used |
|  | Shield | Cable shield | Functional earth |

## 0 <br> NOTICE! <br> Risk of corrosion!

Unused connectors and slots may corrode if XC devices are used in salt-mist environments.

Protect unused connectors and slots with TA535 protective caps for XC devices Chapter 5.2.8.3.6 "TA535-Protective caps for XC devices" on page 1332.
« Supported protocols and used Ethernet ports
${ }^{4}$ Communication via Modbus TCP/IP
${ }^{\star}$ Supported protocols and used Ethernet ports

## CAN interface

This interface is the connection to a processor module with onboard CAN e.g. PM56xx-2ETH.

| Interface socket | COMBICON, 5-pin, female, removable plug with spring terminals |
| :--- | :--- |
| Transmission standard | ISO 11898, potential-free |
| Transmission protocol | CANopen (CAN), 1 Mbaud max. |
| Transfer rate (transmis- <br> sion rate) | $50 \mathrm{kbit} / \mathrm{s}, 100 \mathrm{kbit} / \mathrm{s}, 125 \mathrm{kbit} / \mathrm{s}, 250 \mathrm{kbit} / \mathrm{s}, 500 \mathrm{kbit} / \mathrm{s}, 800 \mathrm{kbit} / \mathrm{s}$ <br> and $1 \mathrm{Mbit} / \mathrm{s}$, |

Pin assignment Table 115: Pin assignment of the CANopen connector

| Interface |  | PIN | Signal | Description |
| :---: | :---: | :---: | :---: | :---: |
| $\xrightarrow[3]{2}$ <br> Terminal block removed | QR | 1 | CAN_GND | CAN reference potential |
|  |  | 2 | CAN_L | Bus line, receive/transmit line, LOW |
|  | (1) 3 - (1) $D$ | 3 | CAN_SHLD | Shield of the bus line |
|  |  | 4 | CAN_H | Bus line, receive/transmit line, HIGH |
|  | $\otimes$ | 5 | NC | Not connected |
|  | Terminal block inserted |  |  |  |

NOTICE!
Unused connector!
Make sure that the terminal block is always connected to the terminal base or communication module, even if you do not use the interface.

Bus length The maximum possible bus length of a CAN network depends on bit rate (transmission rate) and cable type. The sum of all bus segments must not exceed the maximum bus length

| Bit Rate (speed) | Bus Length |
| :--- | :--- |
| $1 \mathrm{Mbit} / \mathrm{s}$ | 40 m |
| $800 \mathrm{kbit} / \mathrm{s}$ | 50 m |
| $500 \mathrm{kbit} / \mathrm{s}$ | 100 m |
| $250 \mathrm{kbit} / \mathrm{s}$ | 250 m |
| $125 \mathrm{kbit} / \mathrm{s}$ | 500 m |
| $50 \mathrm{kbit} / \mathrm{s}$ | 1000 m |

Types of bus Only bus cables with characteristics as recommended in ISO 11898 are to be used. The cables requirements for the bus cables depend on the length of the bus segment.

Bus terminating Both ends of the CAN bus have to be terminated with a $120 \Omega(\geq 1 / 4 \mathrm{~W}, \leq 5 \%)$ bus terminating resistors resistor, to minimize signal reflection. The bus terminating resistor should be connected directly at the bus connector between the CAN signals (CAN_H and CAN_L).

### 5.2.3.1.4 Technical data

The system data of AC500 and S500 are applicable to the standard version. $\Leftrightarrow$ Chapter 5.1.2 "System data AC500" on page 166

The system data of AC500-XC are applicable to the XC version. $\Leftrightarrow$ Chapter 5.1.3 "System data AC500-XC" on page 169
Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter | Value |
| :--- | :--- |
| Connection of the supply voltage 24 V <br> DC at the terminal base of the processor <br> module | Removable 5-pin terminal block spring type |
| Max. current consumption from 24 V DC | TB5600: $0.25 \mathrm{~A}^{1}$ ) |
|  | TB5610: $0.35 \mathrm{~A}^{1}$ ) |
|  | TB5620: $0.4 \mathrm{~A}^{1}$ ) |
|  | TB5640: $0.6 \mathrm{~A}^{1}$ ) |
|  | TB5660: $0.8 \mathrm{~A}^{1}$ ) |
| Melting integral of a fuse at 24 V DC | Min. $1 \mathrm{~A}^{2} \mathrm{~s}^{2}$ ) |
| Peak inrush current from 24 V DC | $55 \mathrm{~A}^{2}$ ) |
| Number of slots for processor modules | 1 (on all terminal bases) |
| Processor module interfaces at TB56xx | I/O bus, ETH1, ETH2, CAN, COM1 |


| Parameter | Value |
| :--- | :--- |
| Net weight (terminal base without pro- | TB5600: 155 g |
| cessor module) | TB5610: 180 g |
|  | TB5620: 210 g |
|  | TB5640: 260 g |
|  | TB5660: 310 g |
| Mounting position | Horizontal or vertical |

${ }^{1}$ ) Including processor modules, communication modules and communication interface modules
${ }^{2}$ ) The inrush current and the melting integral depends on the internal power supply of the processor module and the number and type of communication modules and I/O modules connected to the I/O bus.
Table 116: Combination of TB56xx-2ETH $(-X C)$ and PM56xx(-XC)

| Processor module | PM5630 | PM5650 | PM5670 | PM5675 |
| :--- | :--- | :--- | :--- | :--- |
| TB5600-2ETH | 0 slot | 0 slot | 0 slot | 0 slot |
| TB5610-2ETH | 1 slot | 1 slot | 1 slot | 1 slot |
| TB5620-2ETH | 2 slots | 2 slots | 2 slots | 2 slots |
| TB5640-2ETH | - | 4 slots | 4 slots | 4 slots |
| TB5660-2ETH | - | - | 6 slots $\left.{ }^{1}\right)$ | 6 slots $\left.{ }^{1}\right)$ |

Remarks:
The slots can be used for connecting communication modules or AC500-S modules. Note that only one AC500-S module can be connected at one terminal base.
${ }^{1}$ ) PM567x must have an index $\geq$ C0.
5.2.3.1.5 Dimensions


1 Din rail 15 mm
2 Din rail 7.5 mm

The dimensions are in mm and in brackets in inch.

### 5.2.3.1.6 Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 110 300 R0278 | TB5600-2ETH, terminal base AC500, <br> slots: 1 processor module, 2 Ethernet <br> RJ45, 1 CAN connector | Active |
| 1SAP 310 300 R0278 | TB5600-2ETH-XC, terminal base <br> AC500, slots: 1 processor module, <br> 2 Ethernet RJ45, 1 CAN connector, <br> XC version | Active |
| 1SAP 111 300 R0278 | TB5610-2ETH, terminal base AC500, <br> slots: 1 processor module, <br> 1 communication module, 2 Ethernet <br> RJ45, 1 CAN connector | Active |
| 1SAP 311 300 R0278 | TB5610-2ETH-XC, terminal base <br> AC500, slots: 1 processor module, <br> 1 communication module, 2 Ethernet <br> RJ45, 1 CAN connector, XC version | Active |


| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 112 300 R0278 | TB5620-2ETH, terminal base AC500, <br> slots: 1 processor module, <br> 2 communication modules, 2 Ethernet <br> RJ45, 1 CAN connector | Active |
| 1SAP 312 300 R0278 | TB5620-2ETH-XC, terminal base <br> AC500, slots: 1 processor module, <br> 2 communication modules, 2 Ethernet <br> RJ45, 1 CAN connector, XC version | Active |
| 1SAP 114 300 R0278 | TB5640-2ETH, terminal base AC500, <br> slots: 1 processor module, <br> 4 communication modules, 2 Ethernet <br> RJ45, 1 CAN connector | Active |
| 1SAP 314 300 R0278 | TB5640-2ETH-XC, terminal base <br> AC500, slots: 1 processor module, <br> 4 communication modules, 2 Ethernet <br> RJ45, 1 CAN connector, XC version | Active |
| 1SAP 116 300 R0278 | TB5660-2ETH, terminal base AC500, <br> slots: 1 processor module, <br> 6 communication modules, 2 Ethernet <br> RJ45, 1 CAN connector | Active |
| 1SAP 316 300 R0278 | TB5660-2ETH-XC, terminal base <br> AC500, slots: 1 processor module, <br> 6 communication modules, 2 Ethernet <br> RJ45, 1 CAN connector, XC version | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

Table 117: Combination of TB56xx-2ETH(-XC) and PM56xx(-XC)

| Processor module | PM5630 | PM5650 | PM5670 | PM5675 |
| :--- | :--- | :--- | :--- | :--- |
| TB5600-2ETH | 0 slot | 0 slot | 0 slot | 0 slot |
| TB5610-2ETH | 1 slot | 1 slot | 1 slot | 1 slot |
| TB5620-2ETH | 2 slots | 2 slots | 2 slots | 2 slots |
| TB5640-2ETH | - | 4 slots | 4 slots | 4 slots |
| TB5660-2ETH | - | - | 6 slots $\left.{ }^{1}\right)$ | 6 slots $\left.{ }^{1}\right)$ |

Remarks:
The slots can be used for connecting communication modules or AC500-S modules. Note that only one AC500-S module can be connected at one terminal base.
${ }^{1}$ ) PM567x must have an index $\geq$ C0.

Table 118: Accessories

| Part no. | Description |
| :--- | :--- |
| 1SAP 180 800 <br> R0001 | TA526, wall mounting accessory |

### 5.2.4 I/O modules

### 5.2.4.1 Safety instructions

## Hot swap

System requirements for hot swapping of I/O modules:

- Types of terminal units that support hot swapping of I/O modules have the appendix TU5xx-H.
- I/O modules as of index FO.

The following I/O bus masters support hot swapping of attached I/O modules:

- Communication interface modules CI5xx as of index FO.
- Processor modules PM56xx-2ETH with firmware version as of V3.2.0.


## NOTICE!

## Risk of damage to I/O modules!

Hot swapping is only allowed for I/O modules.
Processor modules and communication interface modules must not be removed or inserted during operation.

## Conditions for hot swapping

- Digital outputs are not under load.
- Input/output voltages above safety extra low voltage/ protective extra low voltages (SELV/PELV) are switched off.
- Modules are completely plugged on the terminal unit with both snap fit engaged before switching on loads or input/output voltage.
$\star$ Further information about hot swap.


### 5.2.4.2 I/O bus - Data transfer

The I/O bus is the I/O data bus for the I/O modules. Through this bus, I/O and diagnosis data are transferred between the processor module and the I/O modules. Up to $10 \mathrm{I} / \mathrm{O}$ modules can be added.
Chapter 4.2.3 "//O bus - Communication within the PLC" on page 97

* Chapter 5.2.4.2 "//O bus - Data transfer" on page 387

The synchronized I/O bus is the I/O data bus for the I/O modules connected with the processor modules or communication interface modules. Through this bus, I/O and diagnosis data are transferred.
With its fast data transmission, the I/O bus obtains very low reaction times.
Up to $10 \mathrm{I} / \mathrm{O}$ terminal units (for one I/O module each) can be added to one terminal base or to one AC500-eCo processor module. The I/O terminal units and the AC500-eCo I/O modules, have a bus input at the left side and a bus output at the right side. Thus the length of the I/O bus increases with the number of attached I/O modules $乡 3$ Table 119 "Maximum number of I/O devices which can be connected to the I/O bus" on page 389.


1 I/O bus connection
The connection of the I/O bus is performed automatically by telescoping the modules on the DIN rail. The I/O bus provides the following signals:

- Supply voltage of 3.3 V DC for feeding the electronic interface components
- 3 data lines for the synchronized serial data exchange
- several control signals


## NOTICE!

Except when using hot swap terminal units, the I/O bus is not designed for pulling and plugging modules during operation. If a module is pulled or plugged on a terminal unit that is not hot swap capable while the bus is running, the following consequences are possible

- reset of the station or of the processor module
- system lockup
- damage of the module


## WARNING!

## Removal/Insertion under power

Removal or insertion under power is permissible only if all conditions for hot swapping are fullfilled.
(4) Chapter 4.9.3 "Replace an I/O module with hot swap" on page 153

The devices are not designed for removal or insertion under power when the conditions for hot swap do not apply. Because of unforeseeable consequences, it is not allowed to plug in or unplug devices with the power being ON.
Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while they are energized in a hazardous location could result in an electric arc, which could create an ignition source resulting in fire or explosion.

Prior to proceeding, make sure that power is been disconnected and that the area has been thoroughly checked to ensure that flammable materials are not present.

The devices must not be opened when in operation. The same applies to the network interfaces.

Profibus (master and slave) and CM589-PNIO are available since version 2.5.0 of the Automation Builder.

Table 119: Maximum number of I/O devices which can be connected to the I/O bus

| Device | Version Automation <br> Builder | Version firmware | Max. number of I/O <br> devices |
| :--- | :--- | :--- | :--- |
| CANopen bus <br> modules CI581-CN <br> and CI582-CN | As of V2.1.0 | All | 0 |
| PROFINET bus <br> modules CI501-PNIO <br> and CI502-PNIO | As of V2.1.0 | all | 10 |
| EtherCAT com- <br> munication inter- <br> face module CI511- <br> ETHCAT and <br> CI512-ETHCAT | As of V2.1.0 | As of V2.0.x | 10 |
| Modbus communica- <br> tion interface module <br> CI521 and CI522 | Independent from <br> Automation Builder <br> version | all | 10 |

Profibus (master and slave) and CM589-PNIO are available since version 2.5.0 of the Automation Builder.

Table 120: General data of the I/O bus

| Parameter | Value |
| :--- | :--- |
| Supply voltage, signal level | $3.3 \mathrm{~V} \mathrm{DC} \pm 10 \%$ |
| Max. supply current | On request |
| Type of the data interface | Synchronized serial data exchange |


| Parameter | Value |
| :--- | :--- |
| Bus data transmission speed | $1.8 \mathrm{Mb} / \mathrm{s}$ |
| Minimum bus cycle time | $500 ~ \mu \mathrm{~s}$ <br> This value is valid for all module combinations <br> (from 1 to $10 \mathrm{I} / \mathrm{O}$ modules) |
| Galvanic isolation | I/O bus is galvanic connected to CPU and <br> communication interface logic ciruits. Galvanic <br> isolation of I/O bus is I/O module specific. See <br> each module specification for details. |
| Protection against electrostatic discharge <br> (ESD) | TB5xx, TB56xx: with protection diodes, <br> no ESD discharge allowed on the port. |

Table 121: Wiring (bus connection)

| Parameter | Value |
| :--- | :--- |
| Bus connection | Left-side and right-side connection from <br> module to module via a 10-pole HE plug (male <br> at the left side, female at the right side) |
| Mechanical connection | Established by the terminal units |
| Max. bus length | 1 m |

### 5.2.4.3 Digital I/O modules

### 5.2.4.3.1 S500-eCo

## DC562 - Digital input/output module

## Features

- 16 configurable digital inputs/outputs in 1 group, 24 V DC
- Module-wise galvanically isolated


1 I/O bus
216 yellow LEDs to display the states of the inputs/outputs C0 ... C15
3 Terminal number
4 Allocation of signal name
5 Terminal block for input and output signals (9-pin)
6 Terminal block for input and output signals (11-pin)
72 holes for wall-mounting with screws
DIN rail

## Intended purpose

The device can be used as a decentralized I/O extension module for S 500 communication interface modules (e. g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs.

The inputs/outputs are group-wise galvanically isolated from each other.
All other circuitry of the module is galvanically isolated from the inputs/outputs.

The I/O module must not be used as a decentralized I/O module with CI590-CS31-HA communication interface modules.

## Functionality

| Parameter | Value |
| :--- | :--- |
| LED displays | For signal states |
| Internal power supply | Via I/O bus |
| External power supply | Via the terminals ZP and UP (process voltage 24 V <br> DC) |

## Connections

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

The connection is carried out by using a removable 9-pin and 11-pin terminal block. These terminal blocks differ in their connection system (spring terminals or screw terminals, cable mounting from the front or from the side). The terminal blocks are not included in the module's scope of delivery and must be ordered separately.


Fig. 43: Internal construction of the digital inputs and outputs

Table 122: Assignment of the terminals:

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 1 | --- | Reserved |
| 2 | C0 | Input/output signal C0 |
| 3 | C1 | Input/output signal C1 |
| 4 | C2 | Input/output signal C2 |
| 5 | C3 | Input/output signal C3 |
| 6 | C4 | Input/output signal C4 |


| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 7 | C5 | Input/output signal C5 |
| 8 | C6 | Input/output signal C6 |
| 9 | C7 | Input/output signal C7 |
| 10 | --- | Reserved |
| 11 | C8 | Input/output signal C8 |
| 12 | C9 | Input/output signal C9 |
| 13 | C10 | Input/output signal C10 |
| 14 | C11 | Input/output signal C11 |
| 15 | C12 | Input/output signal C12 |
| 16 | C13 | Input/output signal C13 |
| 17 | C14 | Input/output signal C14 |
| 18 | C15 | Input/output signal C15 |
| 19 | UP | Process voltage UP +24 V DC |
| 20 | ZP | Process voltage ZP 0 V DC |

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a communication interface module or a CPU). Thus, the current consumption from 24 V DC power supply at the terminals L+/UP and M/ZP of the CPU/communication interface module increases by 10 mA per DC562.

The external power supply connection is carried out via the UP (+24 V DC) and ZP ( 0 V DC) terminals.

## WARNING!

## Removal/Insertion under power

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.
Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.
Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.
The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

Risk of damaging the PLC modules!
Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

Process supply voltage must be connected to UP/ZP of the module. The inputs and UP/ZP must use the same power supply.


Fig. 44: Connection of the digital input/output module DC562
In this connection example, the inputs/outputs $\mathrm{C} 0 \ldots \mathrm{C} 7$ are connected as inputs and the inputs/ outputs C8 ... C15 are connected as outputs.

The module provides several diagnosis functions $\Leftrightarrow$ Chapter 5.2.4.3.1.1.7 "Diagnosis" on page 397.
The meaning of the LEDs is described in the section State LEDs ${ }^{\mu} \Rightarrow$ Chapter 5.2.4.3.1.1.8 "State LEDs" on page 397.

## I/O configuration

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or communication interface module) during power-up of the system.

Hence, replacing I/O modules is possible without any re-parameterization via software.
$\qquad$
If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.

## Parameterization

The arrangement of the parameter data is performed with Automation Builder software.
The parameter data directly influences the functionality of modules.
For non-standard applications, it is necessary to adapt the parameters to your system configuration.

| Name | Value | Internal <br> Value | Internal <br> Value, <br> Type | Default | Min. | Max. | EDS SIot <br> Index |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Module ID | Internal | $6155^{1}$ ) | WORD | 6155 <br> $0 x 180 B$ | 0 | 65535 | $x x 01$ |
| Ignore <br> module | No <br> Yes | 0 | BYTE | No <br> $(0 x 00)$ |  |  |  |
| Parameter <br> length 2) | Internal | 1 - CPU | BYTE | 0 | 0 | 255 | $\left.x x 02^{3}\right)$ |

${ }^{1}$ ) with CS31 and addresses less than 70, the value is increased by 1
${ }^{2}$ ) the module has no additional user-configurable parameters
${ }^{3}$ ) Value is hexadecimal: HighByte is slot (xx: $0 \ldots 7$ ), LowByte is index (1 ... n) GSD file:

| Ext_User_Prm_Data_Len $=$ | $0 \times 06$ |
| :--- | :--- |
| Ext_User_Prm_Data_Const $(0)=$ | $0 \times 18,0 \times 0 \mathrm{C}, 0 \times 00,0 \times 02,0 \times 00,0 \times 00 ;$ |

Diagnosis

| E1 ... E4 | d1 | d2 | d3 | d4 | Identifier $000 \text {... } 063$ | $\|$AC500- <br> Display | <- Display in |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err | PS501 <br> PLC <br> Browser |  |
| $\begin{array}{\|l\|} \hline \text { Byte } 6 \\ \text { Bit } 6 \ldots 7 \end{array}$ | - | Byte 3 | Byte 4 | Byte 5 | Byte 6 <br> Bit 0 ... 5 | PNIO diagnosis block |  |
| Class | Inter- <br> face | Device | Module | Channel | Error- <br> Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | $\left.{ }^{3}\right)$ | $\left.{ }^{4}\right)$ |  |  |  |
| Module error DC562 |  |  |  |  |  |  |  |
| 3 | 14 | 1 ... 10 | 31 | 31 | 19 | Checksum error in the I/O module | Replace I/O module |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |
| 3 | 14 | 1 ... 10 | 31 | 31 | 43 | Internal error in the module | Replace I/O module |
|  | 11/12 | ADR | 1 ... 10 |  |  |  |  |
| 3 | 14 | 1 ... 10 | 31 | 31 | 9 | Overflow diagnosis buffer | Restart |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |
| 3 | 14 | 1 ... 10 | 31 | 31 | 26 | Parameter error | Check master |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |

Remarks:

| ${ }^{1}$ ) | In AC500 the following interface identifier applies: $14=1 / O$ bus, $11=$ COM1 (e.g. CS31-Bus), $12=$ COM2. <br> The PNIO diagnosis block does not contain this identifier. |
| :---: | :---: |
| $\left.{ }^{2}\right)$ | With "Device" the following allocation applies: <br> 31 = Module itself, $1 . . .10=$ expansion module $1 \ldots$.. 10, ADR = hardware address (e. g. of the DC551-CS31) |
| ${ }^{3}$ ) | With "Module" the following allocation applies depending on the master: <br> Module error: I/O bus or PNIO: 31 = module itself; COM1/COM2: 1 ... 10 = expansion 1 ... 10 <br> Channel error: I/O bus or PNIO = module type ( 4 = DC); COM1/COM2: 1 ... 10 = expansion 1 ... 10 |
| ${ }^{4}$ ) | In case of module errors, with channel "31 = Module itself" is output. |

## State LEDs

| LED |  | State | Color | LED = OFF | LED = ON |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  | Inputs/outputs C0 ... C15 | Digital input or digital output | Yellow | Input/output is OFF | Input/output is ON <br> (the LEDs are only operating if the module's circuitry is supplied via the I/O bus) |

## Technical data

## Technical data of the module

The system data of AC500-eCo apply.
*y Chapter 5.1.1 "System data AC500-eCo" on page 159
Only additional details are therefore documented below.

| Parameter | Value |  |
| :--- | :--- | :--- |
| Process voltage UP |  |  |
|  | Connections | Terminal 19 for UP (+24 V DC) and terminal 20 <br> for ZP (0 V) |
|  | Rated value | 24 V DC |
|  | Current consumption via UP terminal | $90 \mathrm{~mA}+0.5 \mathrm{~A}$ per output (max.) |
|  | Max. ripple | $5 \%$ |
|  | Inrush current | Protection against reversed voltage |
| Current consumption from 24 V DC power <br> supply at the L+/UP and M/ZP terminals of <br> the CPU/communication interface module | Yes |  |
| Galvanic isolation | Yes, between the input/output group and the <br> rest of the module |  |
|  | Isolated groups | 1 group for 16 channels |
| Surge voltage (max.) | 35 V DC for 0.5 s |  |
| Max. power dissipation within the module | 4.8 W |  |
| Input data length | 2 bytes |  |
| Output data length | 2 bytes |  |
| Weight | Ca. 125 g |  |
| Mounting position | Horizontal or vertical |  |
| Cooling | The natural convection cooling must not be hin- <br> dered by cable ducts or other parts in the con- <br> trol cabinet. |  |

No effects of multiple overloads

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an external fuse.

Technical data of the digital inputs/outputs if used as inputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 16 configurable inputs (24 V DC) |
| Distribution of the channels into groups | $1(16$ channels per group) |
| Connections of the channels C0 to C15 | Terminals 1 to 16 |
| Reference potential for the channels C0 to <br> C15 | Terminal 20 (negative pole of the process <br> voltage, name ZP) |
| Indication of the input signals | 1 yellow LED per channel; the LED is ON <br> when the input signal is high (signal 1). The <br> module is powered through the I/O bus. |


| Parameter | Value |
| :---: | :---: |
| Input type according to EN 61131-2 | Type 1 sink |
| Input signal range | +24 V DC |
| Signal 0 | -3 V ... +5V |
| Undefined signal | +5 V ... +15 V |
| Signal 1 | +15 V ... +30 V |
| Ripple with signal 0 | -3 V ... +5V |
| Ripple with signal 1 | +15 V ... +30 V |
| Input current per channel |  |
| Input voltage +24 V | Typ. 5 mA |
| Input voltage +5V | Typ. 1 mA |
| Input voltage +15 V | $>2.5 \mathrm{~mA}$ |
| Input voltage +30 V | $<8 \mathrm{~mA}$ |
| Max. permissible leakage current (at 2-wire proximity switches) | 1 mA |
| Input delay (0->1 or 1->0) | Typ. 8 ms |
| Max. cable length |  |
| Shielded | 500 m |
| Unshielded | 300 m |

## Technical data of the digital inputs/outputs if used as outputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 16 configurable transistor outputs |
| Distribution of the channels into groups | 1 (16 channels per group) |
| Connections of the channels C0 to C15 | Terminals 1 to 16 |
| Reference potential for the channels C0 to <br> C15 | Terminal 20 (negative pole of the process <br> voltage, signal name ZP) |
| Common power supply voltage | Terminal 19 (positive pole of the process <br> voltage, signal name UP) |
| Indication of the input signals | 1 yellow LED per channel; the LED is ON when <br> the input signal is high (signal 1). The module <br> is powered through the I/O bus. |
| Way of operation | Non-latching type |
| Output voltage at signal 1 | UP -0.3 V at max. current |
| Output delay (max. at rated load) | 0 50 $\mu \mathrm{s}$ |
|  | 0 to 1 |
| 1 to 0 | $200 \mu \mathrm{~s}$ |
| Output current | 0.5 A at UP 24 V DC |
|  | Rated current per channel (max.) |


| Parameter |  | Value |
| :--- | :--- | :--- |
|  | Lamp load (max.) | 5 W |
|  | Max. leakage current with signal 0 | $<0.5 \mathrm{~mA}$ |
| Output type | Non-protected |  |
| Protection type | External fuse on each channel |  |
| Rated protection fuse (for each channel) | 3 A fast |  |
| Demagnetization when inductive loads are <br> switched off | Must be performed externally according to <br> driven load specification |  |
| Switching frequency |  |  |
|  | With inductive loads | Max. 0.5 Hz |
|  | With lamp loads | Max. 11 Hz at max. 5 W |
| Short-circuit-proof / Overload-proof | No |  |
|  | Overload message | No |
|  | Output current limitation | No |
|  | Resistance to feedback against 24 V DC <br> signals | Yes |
| Connection of 2 outputs in parallel | Not possible |  |
| Max. cable length |  |  |
|  | Shielded | 500 m |
|  | Unshielded | 150 m |

## Dimensions



The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 231900 R0000 | DC562, digital input/output module, <br> 16 configurable inputs/outputs, <br> transistor output | Active |
| 1TNE 968 901 R3101 | Terminal block TA563-9, 9 pins, screw <br> front, cable side, 6 pieces per unit | Active |
| 1TNE 968 901 R3102 | Terminal block TA563-11, 11 pins, <br> screw front, cable side, 6 pieces per <br> unit | Active |
| 1TNE 968 901 R3103 | Terminal block TA564-9, 9 pins, screw <br> front, cable front, 6 pieces per unit | Active |
| 1TNE 968901 R3104 | Terminal block TA564-11, 11 pins, <br> screw front, cable front, 6 pieces per <br> unit | Active |
| 1TNE 968901 R3105 | Terminal block TA565-9, 9 pins, spring <br> front, cable front, 6 pieces per unit | Active |
| 1TNE 968 901 R3106 | Terminal block TA565-11, 11 pins, <br> spring front, cable front, 6 pieces per <br> unit | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

## DI561-Digital input module

## Features

- 8 digital inputs 24 V DC / 24 V AC (IO ... 17) in 1 group
- Module-wise galvanically isolated


1 I/O bus
28 yellow LEDs to display the signal states of the inputs 10 to 17
3 Terminal number
4 Allocation of signal name
5 Terminal block for input signals (9-pin)
62 holes for wall-mounting with screws
7 DIN rail

## Intended purpose

The device can be used as a decentralized I/O extension module for S500 communication interface modules (e. g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs.

The inputs are group-wise galvanically isolated from each other.
All other circuitry of the module is galvanically isolated from the inputs.

The I/O module must not be used as a decentralized I/O module with CI590-CS31-HA communication interface modules.

## Functionality

| Parameter | Value |
| :--- | :--- |
| LED displays | For signal states |
| Internal power supply | Via I/O bus |
| External power supply | Not necessary |

## Connections

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

The connection is carried out by using a removable 9-pin terminal block. These terminal blocks differ in their connection system (spring terminals or screw terminals, cable mounting from the front or from the side). The terminal blocks are not included in the module's scope of delivery and must be ordered separately.


Fig. 45: Internal construction of the digital inputs

Table 123: Assignment of the terminals:

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 1 | $\mathrm{CO} \ldots 7$ | Input common for signals IO to <br> I7 |
| 2 | IO | Input signal IO |
| 3 | I1 | Input signal I1 |
| 4 | I2 | Input signal I2 |
| 5 | I3 | Input signal I3 |
| 6 | I4 | Input signal I4 |
| 7 | I5 | Input signal I5 |
| 8 | I6 | Input signal I6 |
| 9 | I7 | Input signal I7 |

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a communication interface module or a CPU). Thus, the current consumption from $24 \vee$ DC power supply at the terminals L+/UP and M/ZP of the CPU/communication interface module increases by 10 mA per DI561.
An external power supply connection is not needed.

## WARNING!

## Removal/Insertion under power

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.
Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.
Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.
The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

## Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

The digital inputs can be used as source inputs or as sink inputs.

## NOTICE!

## Risk of malfunctions in the plant!

A ground fault, e. g. caused by a damaged cable insulation, can bridge switches accidentally.

Use sink inputs when possible or make sure that, in case of error, there will be no risks to persons or plant.

Table 124: Connection of the digital input module DI561


The module provides several diagnosis functions $\stackrel{\wedge}{ }{ }^{\circ}$ Chapter 5.2.4.3.1.2.7 "Diagnosis" on page 406.
The meaning of the LEDs is described in the section State LEDs ${ }^{\mu}$ Chapter 5.2.4.3.1.2.8 "State LEDs" on page 407.

## I/O Configuration

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or communication interface module) during power-up of the system.
Hence, replacing I/O modules is possible without any re-parameterization via software.

> If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.

## Parameterization

The arrangement of the parameter data is performed with Automation Builder software.
The parameter data directly influences the functionality of modules.
For non-standard applications, it is necessary to adapt the parameters to your system configuration.

| Name | Value | Internal <br> Value | Internal <br> Value, <br> Type | Default | Min. | Max. | EDS SIot <br> Index |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Module ID | Internal | $6105^{1}$ ) | WORD | 6105 <br> $0 x 17 D 9$ | 0 | 65535 | $x x 01$ |
| Ignore <br> module | No <br> Yes | 0 <br> 1 | BYTE | No (0x00) |  |  |  |
| Parameter <br> length $\left.{ }^{2}\right)$ | Internal | $1-$ CPU | BYTE | 0 | 0 | 255 | $\left.x \times 02^{3}\right)$ |

${ }^{1}$ ) with CS31 and addresses smaller than 70 , the value is increased by 1
${ }^{2}$ ) the module has no additional user-configurable parameters
${ }^{3}$ ) Value is hexadecimal: HighByte is slot (xx: $0 \ldots 7$ ), LowByte is index (1 ... n) GSD file:

| Ext_User_Prm_Data_Len $=$ | $0 \times 03$ |
| :--- | :--- |
| Ext_User_Prm_Data_Const $(0)=$ | $0 x D A, 0 \times 17,0 \times 00 ;$ |

Diagnosis

| E1 ... E4 | d1 | d2 | d3 | d4 | $\begin{aligned} & \hline \text { Identifier } \\ & 000 \ldots 063 \end{aligned}$ | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err |  |  |
| Byte 6 <br> Bit 6 ... 7 | - | Byte 3 | Byte 4 | Byte 5 | Byte 6 <br> Bit 0 ... 5 |  |  |
| Class | Interface | Device | Module | Channel | Error Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) | ${ }^{4}$ ) |  |  |  |
| Module error |  |  |  |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 19 | Checksum error in the I/O module | Replace I/O module |
|  | 11/12 | ADR | $1 . .10$ |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 43 | Internal error in the module | Replace I/O module |
|  | $11 / 12$ | ADR | $1 . .10$ |  |  |  |  |
| 3 | 14 | $1 \ldots 10$ | 31 | 31 | 9 | Overflow diagnosis buffer | Restart |
|  | 11/12 | ADR | $1 . .10$ |  |  |  |  |
| 3 | 14 | $1 \ldots 10$ | 31 | 31 | 26 | Parameter error | Check master |
|  | 11/12 | ADR | $1 . .10$ |  |  |  |  |

Remarks:

| ${ }^{1}$ ） | In AC500 the following interface identifier applies： <br> $14=$ I／O bus， $11=$ COM1（e．g．CS31 bus），12＝COM2． <br> The PNIO diagnosis block does not contain this identifier． |
| :--- | :--- |
| $\left.{ }^{2}\right)$ | With＂Device＂the following allocation applies： <br> $31=$ module itself， <br> $1 \ldots 10=$ decentralized communication interface module 1 $\ldots$ ．．10， <br> ADR＝hardware address（e．g．of the DC551－CS31） |
| $\left.{ }^{3}\right)$ | With＂Module＂the following allocation applies depending on the master： <br> Module error：I／O bus or PNIO：31＝module itself；COM1／COM2： $1 \ldots 10=$ <br> expansion 1 ．．．10 |
| $\left.{ }^{4}\right)$ | In case of module errors，with channel＂31＝module itself＂is output． |

## State LEDs

| LED |  | State | Color | LED＝OFF | LED＝ON |
| :---: | :---: | :---: | :---: | :---: | :---: |
| A13 D1561 | Inputs 10．．．17 | Digital input | Yellow | Input is OFF | Input is ON |
| 品吅吅吅吅 |  |  |  |  |  |
| 20 ${ }^{\circ}$ |  |  |  |  |  |
| 『『ロロロロ |  |  |  |  |  |

In the undefined signal range，the state LED for the inputs can be ON although the input state detected by the module is OFF．

## Technical data

## Technical data of the module

The system data of AC500－eCo apply．
＊Chapter 5．1．1＂System data AC500－eCo＂on page 159
Only additional details are therefore documented below．

| Parameter | Value |
| :--- | :--- |
| Galvanic isolation | Yes，between the input group and the rest of <br> the module |
|  | Isolated groups |
| Current consumption from 24 V DC power <br> supply at the L＋／UP and M／ZP terminals of <br> the CPU／communication interface module | Ca． 10 mA |
| Max．power dissipation within the module | 1.6 W |
| Weight | Ca． 110 g |


| Parameter | Value |
| :--- | :--- |
| Mounting position | Horizontal or vertical |
| Cooling | The natural convection cooling must not be hin- <br> dered by cable ducts or other parts in the con- <br> trol cabinet. |

## Technical data of the digital inputs



[^9]
## Dimensions



The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1TNE 968 902 R2101 | DI561, digital input module, 8 DI, <br> 24 V DC / 24 V AC | Active |
| 1TNE 968 901 R3101 | Terminal block TA563-9, 9 pins, screw <br> front, cable side, 6 pieces per unit | Active |
| 1TNE 968 901 R3103 | Terminal block TA564-9, 9 pins, screw <br> front, cable front, 6 pieces per unit | Active |
| 1TNE 968 901 R3105 | Terminal block TA565-9, 9 pins, spring <br> front, cable front, 6 pieces per unit | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

## DI562 - Digital input module

## Features

- 16 digital inputs 24 V DC / 24 V AC (IO ... I15) in 2 groups
- Group-wise galvanically isolated


1 I/O bus
216 yellow LEDs to display the signal states of the inputs I0 ... I15
3 Terminal number
4 Allocation of signal name
5 Terminal block for input signals (9-pin)
6 Terminal block for input signals (11-pin)
72 holes for wall-mounting with screws
DIN rail

## Intended purpose

The device can be used as a decentralized I/O extension module for S 500 communication interface modules (e. g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs.

The inputs are group-wise galvanically isolated from each other.
The other electronic circuitry of the module is galvanically isolated from the inputs.

The I/O module must not be used as a decentralized I/O module with CI590-CS31-HA communication interface modules.

Functionality

| Parameter | Value |
| :--- | :--- |
| LED displays | For signal states |
| Internal power supply | Via I/O bus |
| External power supply | Not necessary |

## Connections

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

The connection is carried out by using removable 9-pin and 11-pin terminal blocks. These terminal blocks differ in their connection system (spring terminals or screw-type terminals, cable mounting from the front or from the side). The terminal blocks are not included in the module's scope of delivery and must be ordered separately.


Fig. 46: Internal construction of the digital inputs

Table 125: Assignments of the terminal

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 1 | CO $\ldots$ C7 | Input common for signals I0 to I7 |
| 2 | IO | Input signal IO |
| 3 | I1 | Input signal I1 |
| 4 | I2 | Input signal I2 |
| 5 | I3 | Input signal I3 |
| 6 | I4 | Input signal I4 |
| 7 | I5 | Input signal I5 |
| 8 | I6 | Input signal I6 |
| 9 | I8 C15 | Input signal I7 |
| 10 | I9 | Input common for signals I8 ... I15 |
| 11 | I11 | Input signal I8 |
| 12 | I12 | Input signal I9 |
| 13 | I13 | Input signal I10 |
| 14 | I14 | Input signal I11 |
| 15 | I15 | Input signal I12 |
| 16 | --- | Input signal I13 |
| 17 | --- | Input signal I14 |
| 18 | In | Input signal I15 |
| 19 | 20 | Reserved |

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a communication interface module or a CPU). Thus, the current consumption from 24 V DC power supply at the terminals L+/UP and M/ZP of the CPU/communication interface module increases by 10 mA per DI562.
An external power supply connection is not needed.

## WARNING!

## Removal/Insertion under power

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.
Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.
Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.
The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

## Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

The module provides several diagnosis functions $\Rightarrow$ Chapter 5.2.4.3.1.3.7 "Diagnosis" on page 415.
The digital inputs can be used as source inputs or as sink inputs.

## NOTICE!

## Risk of malfunctions in the plant!

A ground fault, e. g. caused by a damaged cable insulation, can bridge switches accidentally.
Use sink inputs when possible or make sure that, in case of error, there will be no risks to persons or plant.

Table 126: Connection of the digital input module DI562


The meaning of the LEDs is described in section State LEDs ${ }^{\wedge}>$ Chapter 5.2.4.3.1.3.8 "State LEDs" on page 415.

## I/O configuration

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or communication interface module) during power-up of the system.

Hence, replacing I/O modules is possible without any re-parameterization via software.

If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.

## Parameterization

The arrangement of the parameter data is performed with Automation Builder software.
The parameter data directly influences the functionality of modules.
For non-standard applications, it is necessary to adapt the parameters to your system configuration.

| Name | Value | Internal <br> Value <br> Internal <br> Value, <br> Type | Default | Min. | Max. | EDS Slot <br> Index |  |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Module ID | Internal | $6110^{1}$ ) | WORD | 6110 <br> $0 x 17 D E$ | 0 | 65535 | $x x 01$ |
| Ignore <br> module | No <br> Yes | 0 | BYTE | No (0x00) |  |  |  |
| Parameter <br> length ${ }^{2}$ ) | Internal | $1-$ CPU | BYTE | 0 | 0 | 255 | $\left.x \times 02^{3}\right)$ |

Remarks:

| $\left.{ }^{1}\right)$ | With CS31 and addresses less than 70, the value is increased by 1 |
| :--- | :--- |
| $\left.{ }^{2}\right)$ | The module has no additional user-configurable parameters |
| $\left.{ }^{3}\right)$ | Value is hexadecimal: HighByte is slot (xx: $0 \ldots 7)$, LowByte is index $(1 \ldots$ <br> $n)$ |

GSD file:

| Ext_User_Prm_Data_Len $=$ | $0 x 03$ |
| :--- | :--- |
| Ext_User_Prm_Data_Const $(0)=$ | $0 x D F, 0 x 17,0 \times 00 ;$ |

Diagnosis

| E1 ... E4 | d1 | d2 | d3 | d4 | Identifier $000 \ldots 063$ | AC500- <br> Display | <- Display in |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err | $\begin{array}{\|l} \hline \text { PS501 } \\ \text { PLC } \\ \text { Browser } \end{array}$ |  |
| Byte 6 <br> Bit 6 ... 7 | - | Byte 3 | Byte 4 | Byte 5 | Byte 6 <br> Bit 0 ... 5 | PNIO diagnosis block |  |
| Class | Interface | Device | Module | Channel | Error Identifier | Error message | Remedy |
|  | $\left.{ }^{1}\right)$ | ${ }^{2}$ ) | ${ }^{3}$ ) | ${ }^{4}$ ) |  |  |  |
| Module error DI562 |  |  |  |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 19 | Checksum error in the I/O module | Replace I/O module |
|  | 11/12 | ADR | $1 . .10$ |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 43 | Internal error in the module | Replace I/O module |
|  | $11 / 12$ | ADR | $1 . .10$ |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 9 | Overflow diagnosis buffer | Restart |
|  | $11 / 12$ | ADR | $1 . .10$ |  |  |  |  |
| 3 | 14 | $1 \ldots 10$ | 31 | 31 | 26 | Parameter error | Check master |
|  | 11/12 | ADR | $1 . .10$ |  |  |  |  |

Remarks:

| $\left.{ }^{1}\right)$ | In AC500 the following interface identifier applies: <br> $14=$ I/O bus, 11 = COM1 (e.g. CS31 bus), 12 = COM2. <br> The PNIO diagnosis block does not contain this identifier. |
| :--- | :--- |
| $\left.{ }^{2}\right)$ | With "Device" the following allocation applies: <br> $31=$ module itself, <br> $1 \ldots 10=$ decentralized communication interface module $1 \ldots 10$, <br> ADR = hardware address (e. g. of the DC551-CS31) |
| $\left.{ }^{3}\right)$ | With "Module" the following allocation applies depending on the master: <br> Module error: I/O bus or PNIO: 31 = module itself; COM1/COM2: $1 \ldots 10=$ <br> expansion $1 \ldots 10$ |
| $\left.{ }^{4}\right)$ | In case of module errors, with channel "31 = module itself" is output. |

## State LEDs

| LED |  | State | Color | LED = OFF | LED = ON |
| :---: | :---: | :---: | :---: | :---: | :---: |
| A1] D1562 | $\int_{15}^{\operatorname{Inp} 5 t s} 10 \ldots$ | Digital input | Yellow | Input is OFF | Input is ON |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| 20 \% 1100 |  |  |  |  |  |
|  |  |  |  |  |  |

In the undefined signal range, the state LED for the inputs can be ON although the input state detected by the module is OFF.

## Technical data

## Technical data of the module

The system data of AC500-eCo apply.

* Chapter 5.1.1 "System data AC500-eCo" on page 159

Only additional details are therefore documented below.

| Parameter |  |
| :--- | :--- |
| Galvanic isolation |  |
| Value |  |
| Current consumption from 24 V DC power <br> supply at the L+/UP and M/ZP terminals of the <br> CPU/communication interface module | Yes, between the input groups and the rest of <br> the module |
| Ca. 10 mA |  |
| Max. power dissipation within the module | 3.2 W |
| Weight | Ca. 115 g |
| Mounting position | Horizontal or vertical |
| Cooling | The natural convection cooling must not be <br> hindered by cable ducts or other parts in the <br> control cabinet. |

## Technical data of the digital inputs

| Parameter | Value |  |  |
| :---: | :---: | :---: | :---: |
| Number of channels per module | 16 inputs ( 24 V DC / 24 V AC ) |  |  |
| Distribution of the channels into groups | 2 (8 channels per group) |  |  |
| Connections of the channels 10 to 17 | Terminals $2 . . .9$ |  |  |
| Connections of the channels I8 to I15 | Terminals $11 . . .18$ |  |  |
| Reference potential for the channels 10 to 17 | Terminal 1 (positive or negative pole of the process supply voltage, signal name 10 ... 17) |  |  |
| Reference potential for the channels 18 to I15 | Terminal 10 (positive or negative pole of the process supply voltage, signal name I8 ... I15) |  |  |
| Indication of the input signals | 1 yellow LED per channel; the LED is ON when the input signal is high (signal 1). The module is powered through the I/O bus. |  |  |
| Monitoring point of input indicator | LED is part of the input circuitry |  |  |
| Input type according to EN 61131-2 | Type 1 source | Type 1 sink | Type 1 AC ${ }^{1}$ ) |
| Input signal range | -24 V DC | +24 V DC | 24 V AC $50 / 60 \mathrm{~Hz}$ |
| Signal 0 | -5V ... +3V | -3V ... +5V | 0 V AC ... 5 V AC |


| Parameter | Value |  |  |
| :---: | :---: | :---: | :---: |
| Undefined signal | -15 V ... -5V | +5 V ... +15 V | $\begin{aligned} & 5 \mathrm{~V} \text { AC } \ldots 14 \mathrm{~V} \\ & \mathrm{AC} \end{aligned}$ |
| Signal 1 | -30 V ... -15 V | +15 V ... +30 V | $\begin{aligned} & 14 \mathrm{~V} \text { AC ... } 27 \mathrm{~V} \\ & \mathrm{AC} \end{aligned}$ |
| Input current per channel |  |  |  |
| Input voltage 24 V | Typ. 5 mA |  | Typ. 5 mA r.m.s. |
| Input voltage 5 V | Typ. 1 mA |  | Typ. 1 mA r.m.s. |
| Input voltage 14 V |  |  | Typ. 2.7 mA r.m.s. |
| Input voltage 15 V | $>2.5 \mathrm{~mA}$ |  |  |
| Input voltage 27 V |  |  | Typ. 5.5 mA r.m.s. |
| Input voltage 30 V | $<8 \mathrm{~mA}$ |  |  |
| Max. permissible leakage current (at 2wire proximity switches) | 1 mA |  | Typ. 1 mA r.m.s. |
| Input delay (0->1 or 1->0) | Typ. 8 ms |  |  |
| Input data length | 2 bytes |  |  |
| Max. cable length |  |  |  |
| Shielded | 500 m |  |  |
| Unshielded | 300 m |  |  |

${ }^{1}$ ) When inputs are used with 24 VAC , external surge limiting filters are required.

## Dimensions



The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1TNE 968 902 R2102 | DI562, digital input module, 16 DI, <br> 24 V DC / 24 V AC | Active |
| 1TNE 968 901 R3101 | Terminal block TA563-9, 9 pins, screw <br> front, cable side, 6 pieces per unit | Active |
| 1TNE 968901 R3102 | Terminal block TA563-11, 11 pins, <br> screw front, cable side, 6 pieces per <br> unit | Active |
| 1TNE 968901 R3103 | Terminal block TA564-9, 9 pins, screw <br> front, cable front, 6 pieces per unit | Active |
| 1TNE 968901 R3104 | Terminal block TA564-11, 11 pins, <br> screw front, cable front, 6 pieces per <br> unit | Active |
| 1TNE 968901 R3105 | Terminal block TA565-9, 9 pins, spring <br> front, cable front, 6 pieces per unit | Active |
| 1TNE 968 901 R3106 | Terminal block TA565-11, 11 pins, <br> spring front, cable front, 6 pieces per <br> unit | Active |

*) Modules in lifecycle Classic are available from stock but not recommended
for planning and commissioning of new installations.

## DI571-Digital input module

## Features

- 8 digital inputs $100 \ldots 240 \mathrm{~V}$ AC (I0 ... 17 ) in 8 groups
- Module-wise galvanically isolated


I/O bus
28 yellow LEDs to display the signal states of the inputs IO ... I7
3 Terminal number
4 Allocation of signal name
5 Terminal block for input signals (9-pin)
6 Terminal block for input signals (11-pin)
72 holes for wall-mounting with screws
DIN rail

## Intended purpose

The device can be used as a decentralized I/O extension module for S500 communication interface modules (e. g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs.

The inputs are group-wise galvanically isolated from each other.
All other circuitry of the module is galvanically isolated from the inputs.

The I/O module must not be used as a decentralized I/O module with CI590-CS31-HA communication interface modules.

## Functionality

| Parameter | Value |
| :--- | :--- |
| LED displays | For signal states |
| Internal power supply | Via I/O bus |
| External power supply | Not necessary |

## Connections

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

The connection is carried out by using removable 9 -pin and 11-pin terminal blocks. These terminal blocks differ in their connection system (spring terminals or screw terminals, cable mounting from the front or from the side). The terminal blocks are not included in the module's scope of delivery and must be ordered separately.


Fig. 47: Internal construction of the digital inputs

Table 127: Assignment of the terminals:

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 1 | IO | Input signal IO |
| 2 | N0 | Neutral conductor for the input signal IO |
| 3 | I1 | Input signal I1 |
| 4 | N1 | Neutral conductor for the input signal I1 |
| 5 | I2 | Input signal I2 |
| 6 | N2 | Neutral conductor for the input signal I2 |
| 7 | I3 | Input signal I3 |
| 8 | N3 | Neutral conductor for the input signal I3 |
| 9 | I4 | Reserved |
| 10 | N4 | Input signal I4 |
| 11 | I5 | Neutral conductor for the input signal I4 |
| 12 | N5 | Input signal I5 |
| 13 |  | Neutral conductor for the input signal I5 |


| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 14 | I6 | Input signal I6 |
| 15 | N6 | Neutral conductor for the input signal I6 |
| 16 | I7 | Input signal I7 |
| 17 | N7 | Neutral conductor for the input signal I7 |
| 18 | --- | Reserved |
| 19 | --- | Reserved |
| 20 | --- | Reserved |

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a communication interface module or a CPU). Thus, the current consumption from 24 V DC power supply at the terminals L+/UP and M/ZP of the CPU/communication interface module increases by 10 mA per DI571.

An external power supply connection is not needed.

## WARNING!

## Risk of death by electric shock!

Hazardous voltages can be present at the terminals of the module.
Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.

## WARNING!

## Removal/Insertion under power

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.
Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

## Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.


Fig. 48: Connection of the digital input module DI571

## NOTICE!

## Risk of damaging the PLC modules!

The PLC modules will be irreparably damaged if a voltage $>240 \mathrm{~V}$ is connected.

Make sure that all inputs are fed from the same phase. The module must not be connected to a 400 V voltage.

The module provides several diagnosis functions $\stackrel{\mu}{ }{ }^{\circ}$ Chapter 5.2.4.3.1.4.8 "Diagnosis" on page 425.
The meaning of the LEDs is described in the section State LEDs * Chapter 5.2.4.3.1.4.9 "State LEDs" on page 425.

## Internal data exchange

| Parameter | Value |
| :--- | :--- |
| Digital inputs (bytes) | 1 |
| Digital outputs (bytes) | 0 |

## I/O configuration

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or communication interface module) during power-up of the system.

Hence, replacing I/O modules is possible without any re-parameterization via software.

$$
\begin{aligned}
& \text { If the external power supply voltage via UP/ZP terminals fails, the I/O module } \\
& \text { loses its configuration data. The whole station has to be switched off and on } \\
& \text { again to re-configure the module. }
\end{aligned}
$$

## Parameterization

The arrangement of the parameter data is performed with Automation Builder software.
The parameter data directly influences the functionality of the modules.
For non-standard applications, it is necessary to adapt the parameters to your system configuration.

| Name | Value | Internal <br> Value | Internal <br> Value, <br> Type | Default | Min. | Max. | EDS SIot <br> Index |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Module ID | Internal | $6115^{1}$ ) | WORD | 6115 <br> $0 \times 17 E 3$ | 0 | 65535 | $x \times 01$ |
| Ignore <br> module | No <br> Yes | 0 | BYTE | No (0x00) |  |  |  |
| Parameter <br> length ${ }^{2}$ ) | Internal | 1 - CPU | BYTE | 0 | 0 | 255 | $\left.x \times 02^{3}\right)$ |

[^10]${ }^{2}$ ) the module has no additional user-configurable parameters
${ }^{3}$ ) Value is hexadecimal: HighByte is slot (xx: $0 \ldots 7$ ), LowByte is index ( $1 \ldots n$ ) GSD file:

| Ext_User_Prm_Data_Len $=$ | $0 x 03$ |
| :--- | :--- |
| Ext_User_Prm_Data_Const $(0)=$ | $0 x D F, 0 x 17,0 x 00 ;$ |

Diagnosis

| E1 ... E4 | d1 | d2 | d3 | d4 | $\begin{aligned} & \hline \text { Identifier } \\ & 000 \ldots 063 \end{aligned}$ |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err |  |  |
| Byte 6 <br> Bit 6 ... 7 | - | Byte 3 | Byte 4 | Byte 5 | Byte 6 <br> Bit 0 ... 5 |  |  |
| Class | Interface | Device | Module | Channel | Error Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | $\left.{ }^{3}\right)$ | $\left.{ }^{4}\right)$ |  |  |  |
| Module error |  |  |  |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 19 | Checksum error in the I/O module | Replace I/O module |
|  | 11 / 12 | ADR | 1... 10 |  |  |  |  |
| 3 | 14 | 1 ... 10 | 31 | 31 | 43 | Internal error in the module | Replace I/O module |
|  | 11/12 | ADR | 1... 10 |  |  |  |  |
| 3 | 14 | $1 \ldots 10$ | 31 | 31 | 9 | Overflow diagnosis buffer | Restart |
|  | 11/12 | ADR | 1... 10 |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 26 | Parameter error | Check master |
|  | 11/12 | ADR | $1 \ldots 10$ |  |  |  |  |

Remarks:

| ${ }^{1}$ ) | In AC500 the following interface identifier applies: <br> $14=$ I/O bus, 11 = COM1 (e.g. CS31 bus), 12 = COM2. <br> The PNIO diagnosis block does not contain this identifier. |
| :--- | :--- |
| $\left.{ }^{2}\right)$ | With "Device" the following allocation applies: <br> $31=$ module itself, $1 \ldots 10=$ decentralized communication interface module <br> $1 \ldots 10$, ADR = hardware address (e. g. of the DC551-CS31) |
| $\left.{ }^{3}\right)$ | With "Module" the following allocation applies depending on the master: <br> Module error: I/O bus or PNIO: 31 = module itself; COM1/COM2: $1 \ldots 10=$ <br> expansion $1 \ldots 10$ |
| $\left.{ }^{4}\right)$ | In case of module errors, with channel "31 = Module itself" is output. |

## State LEDs

| LED |  | State | Color | LED = OFF | LED = ON |
| :---: | :---: | :---: | :---: | :---: | :---: |
| A1\# Dis71 | Inputs $10 . . .17$ | Digital input | Yellow | Input is OFF | Input is ON |
| ○吅 |  |  |  |  | (the input voltag |
| 吅 ${ }^{51}$ |  |  |  |  | only displayed if the |
| $\underbrace{3}_{801}$ |  |  |  |  | supply voltage of the module is ON) |

## Technical data

## Technical data of the module

The system data of AC500-eCo apply.
« Chapter 5.1.1 "System data AC500-eCo" on page 159
Only additional details are therefore documented below.

| Parameter | Value |
| :--- | :--- |
| Galvanic isolation | Yes, between the channels and the rest of the <br> module |
|  | Isolated groups |
| Current consumption from 24 V DC power <br> supply at the L+/UP and M/ZP terminals of <br> the CPU/communication interface module | Ca. 10 mA |
| Max. power dissipation within the module | On request |
| Weight | Ca. 135 g |
| Mounting position | Horizontal or vertical |
| Cooling | The natural convection cooling must not be <br> hindered by cable ducts or other parts in the <br> control cabinet. |

## Technical data of the digital inputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 8 AC inputs (100-240 V AC) |
| Distribution of the channels into groups | 8 (1 channel per group) |
| Input voltage range | $0 \vee \mathrm{AC} \ldots 264 \mathrm{~V}$ AC ( $47 \mathrm{~Hz} \ldots 63 \mathrm{~Hz}$ ) |
| Input current per channel (typically at $+25^{\circ} \mathrm{C}$ ) | $\begin{aligned} & <5 \mathrm{~mA} \text { (at } 40 \mathrm{~V} \mathrm{AC} \text { ) } \\ & >6 \mathrm{~mA}(\text { at } 159 \mathrm{~V} \mathrm{AC}, 50 \mathrm{~Hz}) \\ & >7 \mathrm{~mA}(\text { at } 159 \mathrm{~V} \mathrm{AC}, 60 \mathrm{~Hz}) \end{aligned}$ |
| Connections of the channels 10 to 17 | Terminals 1, 3, 5, 7, 10, 12, 14, 16 |
| Reference potential for the channels 10 to 17 | Terminals $2,4,6,8,11,13,15,17$ |
| Indication of the input signals | 1 yellow LED per channel; the LED is ON when the input signal is high (signal 1) |
| Input type according to EN 61131-2 | Type 1 |
| Input signal range |  |
| Signal 0 (max.) | 20 V AC |
| Undefined signal | $20 \mathrm{~V} \mathrm{AC}<\mathrm{U}<79 \mathrm{~V} \mathrm{AC}$ |
| Signal 1 (min.) | 79 V AC |
| Input delay |  |
| Signal 0 -> 1 | Typ. 15 ms |
| Signal 1 -> 0 | Typ. 30 ms |
| Input data length | 1 byte |
| Max. permissible leakage current (at 2-wire proximity switches) | 1 mA |


| Parameter | Value |
| :--- | :--- |
| Max. cable length |  |
|  | Shielded |
|  | Unshielded |

## Dimensions



The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1TNE 968 902 R2103 | DI571, digital input module, 8 DI, <br> 100 V AC ... 240 V AC | Active |
| 1TNE 968 901 R3101 | Terminal block TA563-9, 9 pins, screw <br> front, cable side, 6 pieces per unit | Active |
| 1TNE 968 901 R3102 | Terminal block TA563-11, 11 pins, <br> screw front, cable side, 6 pieces per <br> unit | Active |
| 1TNE 968 901 R3103 | Terminal block TA564-9, 9 pins, screw <br> front, cable front, 6 pieces per unit | Active |
| 1TNE 968 901 R3104 | Terminal block TA564-11, 11 pins, <br> screw front, cable front, 6 pieces per <br> unit | Active |


| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1TNE 968 901 R3105 | Terminal block TA565-9, 9 pins, spring <br> front, cable front, 6 pieces per unit | Active |
| 1TNE 968 901 R3106 | Terminal block TA565-11, 11 pins, <br> spring front, cable front, 6 pieces per <br> unit | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

## DI572 - Digital input module

## Features

- 16 digital inputs $100 \ldots 240$ V AC (IO ... I15) in 2 groups
- Module-wise galvanically isolated


1 I/O bus
216 yellow LEDs to display the signal states of the inputs IO ... I15

3 Terminal number
4 Allocation of signal name
5 Terminal block for input signals (9-pin)
6 Terminal block for input signals (11-pin)
72 holes for wall-mounting with screws
8 DIN rail

## Intended purpose

The device can be used as a decentralized I/O extension module for S500 communication interface modules (e. g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs.

The inputs are group-wise galvanically isolated from each other.
All other circuitry of the module is galvanically isolated from the inputs.

The I/O module must not be used as a decentralized I/O module with CI590-CS31-HA communication interface modules.

## Functionality

| Parameter | Value |
| :--- | :--- |
| LED displays | For signal states |
| Internal power supply | Via I/O bus |
| External power supply | Not necessary |

## Connections

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

The connection is carried out by using removable 9-pin and 11-pin terminal blocks. These terminal blocks differ in their connection system (spring terminals or screw terminals, cable mounting from the front or from the side). The terminal blocks are not included in the module's scope of delivery and must be ordered separately.


Fig. 49: Block diagram for the internal construction of the digital inputs.

Table 128: Assignment of the terminals

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 1 | IO | Input signal I0 |
| 2 | I1 | Input signal I1 |
| 3 | I2 | Input signal I2 |
| 4 | I3 | Input signal I3 |
| 5 | I4 | Input signal I4 |
| 6 | I5 | Input signal I5 |
| 7 | I7 | Input signal I6 |
| 8 | I8 $\ldots 7$ | Input signal I7 |
| 9 | I9 | Neutral conductor for the input signals I0 ... I7 |
| 10 | I10 | Input signal I8 |
| 11 | I11 | Input signal I9 |
| 12 | I13 | Input signal I10 |
| 13 | Input signal I11 |  |
| 14 |  | Input signal I12 |
| 15 |  | Input signal I13 |


| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 16 | I14 | Input signal I14 |
| 17 | I15 | Input signal I15 |
| 18 | N8 ... 15 | Neutral conductor for the input signals I8 ... I15 |
| 19 | --- | Reserved |
| 20 | --- | Reserved |

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a communication interface module or a CPU). Thus, the current consumption from 24 V DC power supply at the terminals L+/UP and M/ZP of the CPU/communication interface module increases by 10 mA per DI572.

An external power supply connection is not needed.

## WARNING!

## Risk of death by electric shock!

Hazardous voltages can be present at the terminals of the module.
Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.

## WARNING!

## Removal/Insertion under power

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.
Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.
Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.
The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

Risk of damaging the PLC modules!
Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.



## NOTICE!

Risk of damaging the PLC modules!
The PLC modules will be irreparably damaged if a voltage $>240 \mathrm{~V}$ is connected.
Make sure that all inputs are fed from the same phase. The module must not be connected to a 400 V voltage.

The module provides several diagnosis functions $\nRightarrow$ Chapter 5.2.4.3.1.5.7 "Diagnosis" on page 434.

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or communication interface module) during power-up of the system.

Hence, replacing I/O modules is possible without any re-parameterization via software.

If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.

## Parameterization

The arrangement of the parameter data is performed with Automation Builder software.
The parameter data directly influences the functionality of modules.
For non-standard applications, it is necessary to adapt the parameters to your system configuration.

| Param- <br> eter <br> name | Value | Internal <br> value | Data type <br> of <br> internal <br> value | Default <br> value | Min. | Max. | EDS Slot <br> Index |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Module ID | Internal | $6160^{1}$ ) | WORD | 6160 <br> $0 \times 1810$ | 0 | 65535 | $x \times 01^{2}$ ) |
|  | No | 0 | BYTE | No <br> $0 x 00$ | - | - | - |
|  | Yes | 1 | BYTE | 3 | 0 | 255 | $\left.x x 02^{2}\right)$ |
| Parameter <br> length | Internal | 3 | BYTE | 20 ms <br> $0 x 00$ | 0 | 1 | - |
| Input <br> delay | 20 ms | 0 | 100 ms | 1 |  |  |  |

${ }^{1}$ ) With CS31 and addresses less than 70, the value is increased by 1.
${ }^{2}$ ) Value is hexadecimal: HighByte is slot (xx: $0 \ldots 7$ ), LowByte is index (1 ... n).
GSD file:

| Ext_Module_Prm_Data_Len $=$ | 7 |
| :--- | :--- |
| Ext_User_Prm_Data_Const $(0)=$ | $0 \times 18,0 \times 11,0 \times 00,0 \times 03,0 \times 00,0 \times 00,0 \times 00 ;$ |

Diagnosis

| E1 ... E4 | d1 | d2 | d3 | d4 | $\begin{aligned} & \text { Identifier } \\ & 000 \ldots 063 \end{aligned}$ | AC500- <br> Display | <- Display in |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err | $\begin{aligned} & \text { PS501 } \\ & \text { PLC } \\ & \text { Browser } \end{aligned}$ |  |
| Byte 6 <br> Bit 6 ... 7 | - | Byte 3 | Byte 4 | Byte 5 | Byte 6 <br> Bit 0 ... 5 | PNIO diagnosis block |  |
| Class | Interface | Device | Module | Channel | Error Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) | ${ }^{4}$ ) |  |  |  |
| 3 | 14 | $1 . . .10$ | 31 | 31 | 19 | Checksum error in the I/O module | Replace I/O module |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 43 | Internal error in the module | Replace I/O module |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 9 | Overflow diagnosis buffer | Restart |
|  | $11 / 12$ | ADR | $1 . .10$ |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 26 | Parameter error | Check master |
|  | 11/12 | ADR | $1 . .10$ |  |  |  |  |

Remarks:

| Param- <br> eter | Remark |
| :--- | :--- |
| $\left.{ }^{1}\right)$ | In AC500 the following interface identifier applies: <br> $14=I / O$ bus, 11 = COM1 (e.g. CS31-Bus), 12 = COM2. <br> The PNIO diagnosis block does not contain this identifier. |
| $\left.{ }^{2}\right)$ | With "Device" the following allocation applies: <br> $31=$ module itself, $1 \ldots 10=$ decentralized communication interface module $1 \ldots$ <br> 10, ADR = hardware address (e.g. of the DC551-CS31) |
| $\left.{ }^{3}\right)$ | With "Module" the following allocation applies depending on the master: <br> module error: I/O bus or PNIO: 31 = module itself; COM1/COM2: $1 \ldots 10=$ expan- <br> sion $1 \ldots 10$ |
| $\left.{ }^{4}\right)$ | In case of module errors, with channel "31 = module itself" is output. |

## State LEDs

| LED |  | State | Color | LED = OFF | LED = ON |
| :---: | :---: | :---: | :---: | :---: | :---: |
| ATH D1572 | Inputs 10 | Digital input | Yellow | Input is OFF | Input is ON |
|  | 115 |  |  |  |  |
|  |  |  |  |  | only displayed if the |
|  |  |  |  |  | onpply volage of |
|  |  |  |  |  | supply voltage of the module is ON ) |

## Technical data

## Technical data of the module

The system data of AC500-eCo apply.
« Chapter 5.1.1 "System data AC500-eCo" on page 159
Only additional details are therefore documented below.

| Parameter | Value |
| :--- | :--- |
| Galvanic isolation | Yes, between the input groups and the rest of <br> the module |
| Isolated groups | 2 (8 channels per group) |
| Current consumption from 24 V DC power <br> supply at the L+/UP and M/ZP terminals of the <br> CPU/communication interface module | Ca. 10 mA |
| Max. power dissipation within the module | 6 W |
| Weight | Ca. 222 g |
| Mounting position | Horizontal or vertical |
| Cooling | The natural convection cooling must not be <br> hindered by cable ducts or other parts in the <br> control cabinet. |

## Technical data of the digital inputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 16 AC inputs (100-240 V AC) |
| Distribution of the channels into groups | 2 (8 channels per group) |
| Input voltage range | 0 V AC ... 264 V AC ( 47 Hz ... 63 Hz ) |
| Input current per channel (typically at $+25^{\circ} \mathrm{C}$ ) | $\begin{aligned} & <3 \mathrm{~mA}(\text { at } 40 \mathrm{~V} \mathrm{AC}) \\ & >6 \mathrm{~mA}(\text { at } 164 \mathrm{~V} \mathrm{AC}) \\ & >8 \mathrm{~mA}(\text { at } 240 \mathrm{~V} \mathrm{AC}) \end{aligned}$ |
| Connections of the channels 10..17 | Terminals 1... 8 |
| Connections of the channels 18...I15 | Terminals $10 . .17$ |
| Reference potential for the channels 10...17 | Terminal 9 |
| Reference potential for the channels 18...115 | Terminal 18 |
| Indication of the input signals | 1 yellow LED per channel. The LED is on when the input signal is high (signal 1). |
| Input type according to EN 61131-2 | Type 1 |
| Input signal range |  |
| Signal 0 (max.) | 40 V AC |
| Undefined signal | $40 \mathrm{~V} \mathrm{AC}<\mathrm{U}<79 \mathrm{~V}$ AC |
| Signal 1 (min.) | 79 V AC |
| Input delay |  |
| Signal 0 -> 1 | Typ. 24 ms |
| Signal 1 -> 0 | Typ. 24 ms |


| Parameter | Value |
| :--- | :--- |
| Input data length | 2 bytes |
| Max. permissible leakage current (at 2-wire prox- <br> imity switches) | 1 mA |
| Max. cable length |  |
|  | Shielded |
|  | Unshielded |

## Dimensions



The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 230 500 R0000 | DI572, digital input module, 16 DI, <br> 100 V AC ... 240 V AC | Active |
| 1TNE 968 901 R3101 | Terminal block TA563-9, 9 pins, screw <br> front, cable side, 6 pieces per unit | Active |
| 1TNE 968 901 R3102 | Terminal block TA563-11, 11 pins, <br> screw front, cable side, 6 pieces per <br> unit | Active |
| 1TNE 968 901 R3103 | Terminal block TA564-9, 9 pins, screw <br> front, cable front, 6 pieces per unit | Active |


| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1TNE 968 901 R3104 | Terminal block TA564-11, 11 pins, <br> screw front, cable front, 6 pieces per <br> unit | Active |
| 1TNE 968 901 R3105 | Terminal block TA565-9, 9 pins, spring <br> front, cable front, 6 pieces per unit | Active |
| 1TNE 968 901 R3106 | Terminal block TA565-11, 11 pins, <br> spring front, cable front, 6 pieces per <br> unit | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

## DO561 - Digital output module

## Features

- 8 digital outputs 24 V DC ( $\mathrm{O} 0 \ldots \mathrm{O}$ ) in 1 group
- Module-wise galvanically isolated


1 I/O bus
28 yellow LEDs to display the signal states of the outputs $\mathrm{O} 0 \ldots \mathrm{O} 7$
3 Terminal number
4 Allocation of signal name
5 Terminal block for output signals (11-pin)
62 holes for wall-mounting with screws
7 DIN rail

Intended purpose
The device can be used as a decentralized I/O extension module for S 500 communication interface modules (e. g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs.

The outputs are group-wise galvanically isolated from each other.
All other circuitry of the module is galvanically isolated from the outputs.


The I/O module must not be used as a decentralized I/O module with CI590-CS31-HA communication interface modules.

Functionality

| Parameter | Value |
| :--- | :--- |
| LED displays | For signal states |
| Internal power supply | Via I/O bus |
| External power supply | Via the terminals ZP and UP (process supply voltage <br> $24 \mathrm{~V} \mathrm{DC})$ |

## Connections

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

The connection is carried out by using removable $9-$ pin and 11-pin terminal blocks. These terminal blocks differ in their connection system (spring terminals or screw terminals, cable mounting from the front or from the side). The terminal blocks are not included in the module's scope of delivery and must be ordered separately.


Fig. 50: Internal construction of the digital outputs

Table 129: Assignment of the terminals:

| Terminals | Signal | Description |
| :--- | :--- | :--- |
| 10 | --- | Reserved |
| 11 | O0 | Output signal O0 |
| 12 | O1 | Output signal O1 |
| 13 | O2 | Output signal O2 |
| 14 | O3 | Output signal O3 |
| 15 | O4 | Output signal O4 |


| Terminals | Signal | Description |
| :--- | :--- | :--- |
| 16 | O5 | Output signal O5 |
| 17 | O6 | Output signal O6 |
| 18 | O7 | Output signal O7 |
| 19 | UP | Process supply voltage <br> UP +24 V DC |
| 20 | ZP | Process supply voltage <br> ZP 0 V |

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a communication interface module or a CPU). Thus, the current consumption from 24 V DC power supply at the terminals L+/UP and M/ZP of the CPU/communication interface module increases by 10 mA per DO561.

The external power supply connection is carried out via the UP (+24 V DC) and ZP ( 0 V DC) terminals.

## WARNING!

## Removal/Insertion under power

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.
Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.
Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.
The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

Risk of damaging the PLC modules!
Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.


Fig. 51: Connection of the digital output module DO561

## NOTICE!

Risk of malfunctions in the plant!
The outputs may switch on for a period of 10 to $50 \mu$ s if the process supply voltage UP/ZP is switched on.
This must be considered in the planning of the application.

## - NOTICE!

## Risk of damaging the I/O module!

The outputs are not protected against short circuits and overload.

- Never short-circuit or overload the outputs.
- Never connect the outputs to other voltages.
- Use an external 3 A fast-protection fuse for the outputs.

The module provides several diagnosis functions ${ }^{\sharp}>$ Chapter 5.2.4.3.1.6.7 "Diagnosis" on page 442.

The meaning of the LEDs is described in the section State LEDs ${ }^{\mu} \Rightarrow$ Chapter 5.2.4.3.1.6.8 "State LEDs" on page 443.

## I/O configuration

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or communication interface module) during power-up of the system.
Hence, replacing I/O modules is possible without any re-parameterization via software.

> If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.

## Parameterization

The arrangement of the parameter data is performed with Automation Builder software.
The parameter data directly influences the functionality of modules.
For non-standard applications, it is necessary to adapt the parameters to your system configuration.

| Name | Value | Internal <br> Value | Internal <br> Value, <br> Type | Default | Min. | Max. | EDS SIot <br> Index |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Module ID | Internal | $6120^{1}$ ) | WORD | 6120 <br> $0 x 17 E 8$ | 0 | 65535 | xx 01 |
| Ignore <br> module | No <br> Yes | 0 | BYTE | No <br> $(0 x 00)$ |  |  |  |
| Parameter <br> length | Internal | 1 | BYTE | 0 | 0 | 255 | $\left.x \times 02^{2}\right)$ |

${ }^{1}$ ) with CS31 and addresses smaller than 70, the value is increased by 1
${ }^{2}$ ) Value is hexadecimal: HighByte is slot (xx: $0 \ldots 7$ ), LowByte is index (1 ... n) GSD file:

| Ext_User_Prm_Data_Len $=$ | $0 x 03$ |
| :--- | :--- |
| Ext_User_Prm_Data_Const $(0)=$ | $0 x E 9,0 x 17,0 x 00 ;$ |

## Diagnosis

| E1 ... E4 | d1 | d2 | d3 | d4 | $\begin{aligned} & \hline \text { Identifier } \\ & 000 \ldots 063 \end{aligned}$ | AC500- <br> Display | <- Display in |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err | $\begin{array}{\|l} \hline \text { PS501 } \\ \text { PLC } \\ \text { Browser } \end{array}$ |  |
| Byte 6 <br> Bit 6 ... 7 | - | Byte 3 | Byte 4 | Byte 5 | Byte 6 <br> Bit 0 ... 5 | PNIO diagnosis block |  |
| Class | Interface | Device | Module | Channel | Error Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | $\left.{ }^{3}\right)$ | 4) |  |  |  |
| Module error D0561 |  |  |  |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 19 | Checksum error in the I/O module | Replace I/O module |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |


| E1 ... E4 | d1 | d2 | d3 | d4 | Identifier $000 \text {... } 063$ | $\left\lvert\, \begin{aligned} & \text { AC500- } \\ & \text { Display }\end{aligned}\right.$ | <- Display in |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err | $\begin{aligned} & \text { PS501 } \\ & \text { PLC } \\ & \text { Browser } \end{aligned}$ |  |
| Byte 6 <br> Bit 6 ... 7 | - | Byte 3 | Byte 4 | Byte 5 | $\begin{aligned} & \text { Byte } 6 \\ & \text { Bit } 0 \text {... } 5 \end{aligned}$ | PNIO diagnosis block |  |
| Class | Interface | Device | Module | Channel | Error Identifier | Error message | Remedy |
|  | $\left.{ }^{1}\right)$ | $\left.{ }^{2}\right)$ | ${ }^{3}$ ) | ${ }^{4}$ ) |  |  |  |
| Module error DO561 |  |  |  |  |  |  |  |
| 3 | 14 | 1 ... 10 | 31 | 31 | 43 | Internal error in the module | Replace I/O module |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |
| 3 | 14 | 1 ... 10 | 31 | 31 | 9 | Overflow diagnosis buffer | Restart |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |
| 3 | 14 | 1 ... 10 | 31 | 31 | 26 | Parameter error | Check master |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |

Remarks:

| ${ }^{1}$ ) | In AC500 the following interface identifier applies: 14 = I/O bus, 11 = COM1 (e.g. CS31-Bus), 12 = COM2. <br> The PNIO diagnosis block does not contain this identifier. |
| :---: | :---: |
| ${ }^{2}$ ) | With "Device" the following allocation applies: <br> $31=$ module itself, $1 \ldots 10=$ decentralized communication interface module $1 \ldots$ <br> 10, ADR = hardware address (e. g. of the DC551-CS31) |
| $\left.{ }^{3}\right)$ | With "Module" the following allocation applies depending on the master: <br> Module error: I/O bus or PNIO: 31 = module itself; COM1/COM2: 1 ... $10=$ expansion 1 ... 10 <br> Channel error: I/O bus or PNIO = module type (2 = DO); COM1/COM2: 1 ... 10 = expansion 1 ... 10 |
| ${ }^{4}$ ) | In case of module errors, with channel "31 = Module itself" is output. |

## State LEDs

| LED |  | State | Color | LED = OFF | LED $=0 \mathrm{~N}$ |
| :---: | :---: | :---: | :---: | :---: | :---: |
| A18 Do56 | Outputs O0 ... | Digital output | Yellow | Output is | Output is ON |
|  |  |  |  |  | (the output volt |
|  |  |  |  |  | only displayed if the |
|  |  |  |  |  | supply voltage of the |

## Technical data

## Technical data of the module

The system data of AC500-eCo apply.
*y Chapter 5.1.1 "System data AC500-eCo" on page 159
Only additional details are therefore documented below.

| Parameter |  | Value |
| :--- | :--- | :--- |
| Process supply voltage UP |  |  |
|  | Connections | Terminal 19 for UP (+24 V DC) and terminal <br> 20 for ZP (0 V DC) |
|  | Rated value | 24 V DC |
|  | Current consumption via UP terminal | $5 \mathrm{~mA}+$ max. 0.5 A per output |
|  | Max. ripple | 5 \% |
|  | Protection against reversed voltage | Rated protection fuse for UP |
| Current consumption from 24 V DC power <br> supply at the L+/UP and M/ZP terminals of the <br> CPU/communication interface module | Recommended; the outputs must be pro- <br> tected by an 3 A fast-acting fuse |  |
| Galvanic isolation | Yes, between the output group and the rest of <br> the module |  |
| Isolated groups | 1 (8 channels per group) |  |
| Surge-voltage (max.) | 35 V DC for 0.5 s |  |
| Power dissipation within the module (max.) | 1.6 W |  |
| Weight | Ca. 115 g |  |
| Mounting position | Horizontal or vertical |  |
| Cooling | The natural convection cooling must not be <br> hindered by cable ducts or other parts in the <br> control cabinet. |  |

No effects of No effects of multiple overloads on isolated multi-channel modules occur, as every channel is multiple overprotected individually by an external fuse.

## loads

## Technical data of the digital outputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 8 transistor outputs (24 V DC, 0.5 A max.) |
| Distribution of the channels into groups | 1 (8 channels per group) |
| Connection of the channels O0 to O7 | Terminals 11 to 18 |
| Common power supply voltage | Terminal 19 (positive pole of the process <br> voltage, signal name UP) |
| Reference potential for the channels O0 to O7 | Terminal 20 (negative pole of the process <br> voltage, signal name ZP) |


| Parameter | Value |
| :---: | :---: |
| Indication of the output signals | 1 yellow LED per channel; the LED is on when the output signal is high (signal 1) and the module is powered via the I/O bus |
| Way of operation | Non-latching type |
| Min. output voltage at signal 1 | 20 V DC at max. current consumption |
| Output delay (max. at rated load) |  |
| 0 to 1 | $50 \mu \mathrm{~s}$ |
| 1 to 0 | $200 \mu \mathrm{~s}$ |
| Output data length | 1 byte |
| Output current |  |
| Rated current per channel (max.) | 0.5 A at UP 24 V DC |
| Rated current per group (max.) | 4 A |
| Lamp load (max.) | 5 W |
| Max. leakage current with signal 0 | 0.5 mA |
| Output type | Non-protected |
| Protection type | External fuse on each channel |
| Rated protection fuse (for each channel) | 3 A fast |
| Demagnetization when inductive loads are switched off | Must be performed externally according to driven load specification |
| Switching Frequencies |  |
| With inductive loads | Max. 0.5 Hz |
| With lamp loads | Max. 11 Hz at max. 5 W |
| Short-circuit-proof / Overload-proof | No |
| Overload message | No |
| Output current limitation | No |
| Resistance to feedback against 24 V DC | No |
| Connection of 2 outputs in parallel | Not possible |
| Max. cable length |  |
| Shielded | 500 m |
| Unshielded | 150 m |

## Dimensions



The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1TNE 968 902 R2201 | DO561, digital output module, 8 DO, <br> transistor output | Active |
| 1TNE 968 901 R3102 | Terminal block TA563-11, 11 pins, <br> screw front, cable side, 6 pieces per <br> unit | Active |
| 1TNE 968 901 R3104 | Terminal block TA564-11, 11 pins, <br> screw front, cable front, 6 pieces per <br> unit | Active |
| 1TNE 968 901 R3106 | Terminal block TA565-11, 11 pins, <br> spring front, cable front, 6 pieces per <br> unit | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

## DO562-Digital output module

## Features

- 16 digital outputs $24 \mathrm{~V} \mathrm{DC}(\mathrm{OO} \ldots \mathrm{O} 15)$ in 1 group
- Module-wise galvanically isolated


1 I/O bus
216 yellow LEDs to display the signal states of the outputs O0 ... O15
3 Terminal number
4 Allocation of signal name
5 Terminal block for output signals (9-pin)
6 Terminal block for output signals (11-pin)
72 holes for wall-mounting with screws
8 DIN rail

## Intended purpose

The device can be used as a decentralized I/O extension module for S500 communication interface modules (e. g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs.

The outputs are group-wise galvanically isolated from each other.

All other circuitry of the module is galvanically isolated from the outputs.


The I/O module must not be used as a decentralized I/O module with CI590-CS31-HA communication interface modules.

## Functionality

| Parameter | Value |
| :--- | :--- |
| LED displays | For signal states |
| Internal power supply | Via I/O bus |
| External power supply | Via the terminals ZP and UP (process supply voltage <br> 24 V DC) |

## Connections

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

The connection is carried out by using removable 9-pin and 11-pin terminal blocks. These terminal blocks differ in their connection system (spring terminals or screw terminals, cable mounting from the front or from the side). The terminal blocks are not included in the module's scope of delivery and must be ordered separately.


Fig. 52: Internal construction of the digital outputs

Table 130: Assignment of the terminals:

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 1 | --- | Reserved |
| 2 | O0 | Output signal O0 |
| 3 | O1 | Output signal O1 |
| 4 | O2 | Output signal O2 |
| 5 | O3 | Output signal O3 |
| 6 | O4 | Output signal O4 |
| 7 | O5 | Output signal O5 |
| 8 | O6 | Output signal O6 |
| 9 | O7 | Output signal O7 |
| 10 | --- | Reserved |
| 11 | O8 | Output signal O8 |
| 12 | O9 | Output signal O9 |
| 13 | O10 | Output signal O10 |


| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 14 | O11 | Output signal O11 |
| 15 | O12 | Output signal O12 |
| 16 | O13 | Output signal O13 |
| 17 | O14 | Output signal O14 |
| 18 | O15 | Output signal O15 |
| 19 | UP | Process voltage UP (24 V DC) |
| 20 | ZP | Process voltage ZP (0 V DC) |

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a communication interface module or a CPU). Thus, the current consumption from 24 V DC power supply at the terminals L+/UP and M/ZP of the CPU/communication interface module increases by 10 mA per DO562.

The external power supply connection is carried out via the UP (+24 V DC) and ZP ( 0 V DC) terminals.

## WARNING!

## Removal/Insertion under power

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.
Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.
The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

## Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.


Fig. 53: Connection of the digital output module DO562

## NOTICE!

Risk of malfunctions in the plant!
The outputs may switch on for a period of 10 to $50 \mu$ s if the process supply voltage UP/ZP is switched on.
This must be considered in the planning of the application.

## NOTICE!

## Risk of damaging the I/O module!

The outputs are not protected against short circuits and overload.

- Never short-circuit or overload the outputs.
- Never connect the outputs to other voltages.
- Use an external 3 A fast-protection fuse for the outputs.

The module provides several diagnosis functions ${ }^{\mu}$ Chapter 5.2.4.3.1.7.7 "Diagnosis" on page 453.

The meaning of the LEDs is described in the section Status LEDs $\Longleftrightarrow$ Chapter 5.2.4.3.1.7.8
"State LEDs" on page 453.

## I/O configuration

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or communication interface module) during power-up of the system.
Hence, replacing I/O modules is possible without any re-parameterization via software.


If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.

## Parameterization

The arrangement of the parameter data is performed with Automation Builder software.
The parameter data directly influences the functionality of modules.
For non-standard applications, it is necessary to adapt the parameters to your system configuration.

| Name | Value | Internal <br> Value | Internal <br> Value, <br> Type | Default | Min. | Max. | EDS Slot <br> Index |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Module ID | Internal | $6145^{1}$ ) | WORD | 6145 <br> $0 \times 1801$ | 0 | 65535 | xx01 |
| Ignore <br> module | No <br> Yes | 0 | BYTE | No <br> $(0 x 00)$ |  |  |  |
| Parameter <br> length | Internal | 1 | BYTE | 0 | 0 | 255 | $\left.x \times 02^{2}\right)$ |

${ }^{1}$ ) with CS31 and addresses less than 70, the value is increased by 1
${ }^{2}$ ) Value is hexadecimal: HighByte is slot (xx: $0 \ldots 7$ ), LowByte is index (1 ... n) GSD file:

| Ext_User_Prm_Data_Len $=$ | $0 \times 06$ |
| :--- | :--- |
| Ext_User_Prm_Data_Const $(0)=$ | $0 \times 18,0 \times 02,0 \times 00,0 \times 02,0 \times 00,0 \times 00 ;$ |

Diagnosis

| E1...E4 | d1 | d2 | d3 | d4 | Identifier $000 \ldots 063$ | AC500- <br> Display | <- Display in |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err | PS501 <br> PLC <br> Browser |  |
| Byte 6 <br> Bit 6 ... 7 | - | Byte 3 | Byte 4 | Byte 5 | Byte 6 <br> Bit 0 ... 5 | PNIO diagnosis block |  |
| Class | Inter- face | Device | Module | Channel | ErrorIdentifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) | $\left.{ }^{4}\right)$ |  |  |  |
| Module error |  |  |  |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 19 | Checksum error in the I/O module | Replace I/O module |
|  | 11/12 | ADR | $1 . .10$ |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 43 | Internal error in the module | Replace I/O module |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 9 | Overflow diagnosis buffer | Restart |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 26 | Parameter error | Check master |
|  | $11 / 12$ | ADR | $1 . .10$ |  |  |  |  |

Remarks:

| ${ }^{1}$ ) | In AC500 the following interface identifier applies: $14=$ I/O bus, $11=$ COM1 (e.g. CS31 bus), $12=$ COM2. <br> The PNIO diagnosis block does not contain this identifier. |
| :---: | :---: |
| ${ }^{2}$ ) | With "Device" the following allocation applies: <br> $31=$ module itself, $1 \ldots 10$ = decentralized communication interface module $1 \ldots 10$, ADR = hardware address (e. g. of the DC551-CS31) |
| ${ }^{3}$ ) | With "Module" the following allocation applies dependent of the master: <br> Module error: I/O bus or PNIO: 31 = Module itself; COM1/COM2: $1 . .10$ = expansion 1 ... 10 <br> Channel error: I/O bus or PNIO = module type (2 = DO); COM1/COM2: 1 ... $10=$ expansion 1 ... 10 |
| ${ }^{4}$ ) | In case of module errors, with channel "31 = Module itself" is output. |

## State LEDs

| LED |  | State | Color | LED = OFF | LED = ON |
| :---: | :---: | :---: | :---: | :---: | :---: |
| A13 DO562 | $\begin{aligned} & \text { Outputs O0 ... } \\ & \text { O15 } \end{aligned}$ | Digital output | Yellow | Output is OFF | Output is ON <br> (the output voltage is only displayed if the supply voltage of the module is ON ) |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| 030070 |  |  |  |  |  |
| $1800.7240 \operatorname{cosa}$ |  |  |  |  |  |

## Technical data

## Technical data of the module

The system data of AC500-eCo apply.
*y Chapter 5.1.1 "System data AC500-eCo" on page 159
Only additional details are therefore documented below.

| Parameter | Value |
| :---: | :---: |
| Process supply voltage UP |  |
| Connections | Terminal 19 for UP (+24 V DC) and terminal 20 for ZP (0 V DC) |
| Rated value | 24 V DC |
| Current consumption via UP terminal | $20 \mathrm{~mA}+\mathrm{max}$. 0.5 A per output |
| Max. ripple | 5 \% |
| Inrush current | $0.000002 \mathrm{~A}^{2} \mathrm{~s}$ |
| Protection against reversed voltage | Yes |
| Rated protection fuse for UP | Recommended; the outputs must be protected by an 3 A fast-acting fuse |
| Current consumption from 24 V DC power supply at the L+/UP and M/ZP terminals of the CPU/communication interface module | Ca. 10 mA |
| Galvanic isolation | Yes, between the output group and the rest of the module |
| Isolated groups | 1 (16 channels per group) |
| Surge-voltage (max.) | 35 V DC for 0.5 s |
| Max. power dissipation within the module | 1.4 W |
| Weight | Ca. 125 g |
| Mounting position | Horizontal or vertical |
| Cooling | The natural convection cooling must not be hindered by cable ducts or other parts in the control cabinet. |

No effects of No effects of multiple overloads on isolated multi-channel modules occur, as every channel is multiple overloads protected individually by an external fuse.

## Technical data of the digital outputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 16 transistor outputs (24 V DC, 0.5 A max.) |
| Distribution of the channels into groups | 1 (16 channels per group) |
| Connection of the channels O0 ... O7 | Terminals $1 \ldots 9$ |
| Connection of the channels O8 ... O15 | Terminals $11 \ldots 18$ |
| Common power supply voltage | Terminal 19 (positive pole of the process voltage, <br> signal name UP) |
| Reference potential for the channels O0 ... <br> O15 | Terminal 20 (negative pole of the process <br> voltage, signal name ZP) |


| Parameter | Value |
| :---: | :---: |
| Indication of the output signals | 1 yellow LED per channel; the LED is on when the output signal is high (signal 1 ) and the module is powered via the I/O bus |
| Way of operation | Non-latching type |
| Min. output voltage at signal 1 | UP -0.3 V at max. current consumption |
| Output delay (max. at rated load) |  |
| 0 to 1 | $50 \mu \mathrm{~s}$ |
| 1 to 0 | $200 \mu \mathrm{~s}$ |
| Output data length | 2 bytes |
| Output current |  |
| Rated current per channel (max.) | 0.5 A at UP 24 V DC |
| Rated current per group (max.) | 8 A |
| Lamp load (max.) | 5 W |
| Max. leakage current with signal 0 | 0.5 mA |
| Output type | Non-protected |
| Protection type | External fuse on each channel |
| Rated protection fuse (for each channel) | 3 A fast |
| Demagnetization when inductive loads are switched off | Must be performed externally according to driven load specification |
| Switching Frequencies |  |
| With inductive loads | Max. 0.5 Hz |
| With lamp loads | Max. 11 Hz at max. 5 W |
| Short-circuit-proof / Overload-proof | No |
| Overload message | No |
| Output current limitation | No |
| Resistance to feedback against 24 V DC | No |
| Connection of 2 outputs in parallel | Not possible |
| Max. cable length |  |
| Shielded | 500 m |
| Unshielded | 150 m |

## Dimensions



The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 230900 R0000 | DO562, digital output module, 16 DO, <br> transistor output | Active |
| 1TNE 968 901 R3101 | Terminal block TA563-9, 9 pins, screw <br> front, cable side, 6 pieces per unit | Active |
| 1TNE 968 901 R3102 | Terminal block TA563-11, 11 pins, <br> screw front, cable side, 6 pieces per <br> unit | Active |
| 1TNE 968 901 R3103 | Terminal block TA564-9, 9 pins, screw <br> front, cable front, 6 pieces per unit | Active |
| 1TNE 968901 R3104 | Terminal block TA564-11, 11 pins, <br> screw front, cable front, 6 pieces per <br> unit | Active |
| 1TNE 968 901 R3105 | Terminal block TA565-9, 9 pins, spring <br> front, cable front, 6 pieces per unit | Active |
| 1TNE 968 901 R3106 | Terminal block TA565-11, 11 pins, <br> spring front, cable front, 6 pieces per <br> unit | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

## DO571 - Digital output module

## Features

- 8 digital normally open relay outputs 24 V DC / 24 V AC or 100 V AC ... 240 V AC, 2 A max. (NOO ... NO7) in 2 groups
- Group-wise galvanically isolated
(6)


1 I/O bus
28 yellow LEDs to display the signal states of the outputs 00 ... O7
3 Terminal number
4 Allocation of signal name
5 Terminal block for output signals (11-pin)
62 holes for wall-mounting with screws
7 DIN rail

## Intended purpose

The device can be used as a decentralized I/O extension module for S500 communication interface modules (e. g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs.

The outputs are group-wise galvanically isolated from each other.
All other circuitry of the module is galvanically isolated from the outputs.

The I/O module must not be used as a decentralized I/O module with CI590-CS31-HA communication interface modules.

## Functionality

| Parameter | Value |
| :--- | :--- |
| LED displays | For signal states |
| Internal power supply | Via I/O bus |
| External power supply | Via the terminal L+ (process voltage 24 V DC). The negative <br> pole is provided by the I/O bus. |

## Connections

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

The connection is carried out by using removable 9 -pin and 11-pin terminal blocks. These terminal blocks differ in their connection system (spring terminals or screw terminals, cable mounting from the front or from the side). The terminal blocks are not included in the module's scope of delivery and must be ordered separately.


Fig. 54: Internal construction of the digital outputs

Table 131: Assignment of the terminals:

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 10 | NO0 | Normally-open contact of the output NO0 |
| 11 | NO1 | Normally-open contact of the output NO1 |
| 12 | NO2 | Normally-open contact of the output NO2 |
| 13 | NO3 | Normally-open contact of the output NO3 |
| 14 | R0..3 | Output common for signals NO0 to NO3 |
| 15 | NO4 | Normally-open contact of the output NO4 |
| 16 | NO6 | Normally-open contact of the output NO5 |
| 17 | NO7 | Normally-open contact of the output NO6 |
| 18 | L4..7 | Normally-open contact of the output NO7 |
| 19 | Output common for signals NO4 to NO7 |  |
| 20 |  | Process voltage L+ +24 V DC |

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a communication interface module or a CPU). Thus, the current consumption from 24 V DC power supply at the terminals L+/UP and M/ZP of the CPU/communication interface module increases by 5 mA per DO571.
The external power supply connection is carried out via the L+ (+24 V DC) terminal. The negative pole of the external power supply is realized via the I/O bus. Therefore, the CPU/ communication interface module and the DO571 must have a common power supply.

## WARNING!

## Risk of death by electric shock!

Hazardous voltages can be present at the terminals of the module.
Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.

For screw-type terminals only:

## WARNING!

## For screw terminals only: Danger of death by electric shock!

The IP 20 protection degree is only provided if all terminal screws are tightened.
Tighten all screws of unused load terminals of relay outputs if voltages $>24 \mathrm{~V}$ are connected to the relay group.

## WARNING!

## Removal/Insertion under power

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.
Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.
Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.
The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

Risk of damaging the PLC modules!
Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.


## NOTICE!

## Risk of damaging the PLC modules!

The PLC modules can be damaged by overload.
Make sure that the total current of each output common terminal (R0 ... R3 and R4 ... R7) does not exceed 8 A.
Never connect total currents > 8 A per group.
If the group fuse protection is not sufficient, then individual fuse protection of the outputs should be used.

Connection of the module:


Fig. 55: Connection of 24 V DC actuators


Fig. 56: Connection of 24 V AC or 100 V AC ... 240 V AC actuators

## NOTICE!

## Risk of damaging the I/O module!

The outputs are not protected against short circuit and overload.

- Never short-circuit or overload the outputs.
- Never connect inductive loads without an external suppression against voltage peaks due to inductive kickback.
- Never connect voltages > 240 V. All outputs must be supplied from the same phase.
- Use an external 5 A fast protection fuse for the outputs.


Fig. 57: Power supply - the negative connection is realized via the I/O bus
CPU or communication interface module
2 I/O bus
3 DO571

The L+ connection of the DO571 and the 24 V supply of the CPU/communication interface module must be connected to the same $24 V$ power supply.

The module provides several diagnosis functions ${ }^{\mu} \Rightarrow$ Chapter 5.2.4.3.1.8.7 "Diagnosis" on page 464.
The meaning of the LEDs is described in the section Status LEDs $\Leftrightarrow$ Chapter 5.2.4.3.1.8.8 "State LEDs" on page 465 .

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or communication interface module) during power-up of the system.

Hence, replacing I/O modules is possible without any re-parameterization via software.

$$
\begin{aligned}
& \text { If the external power supply voltage via UP/ZP terminals fails, the I/O module } \\
& \text { loses its configuration data. The whole station has to be switched off and on } \\
& \text { again to re-configure the module. }
\end{aligned}
$$

## Parameterization

The arrangement of the parameter data is performed with Automation Builder software.
The parameter data directly influences the functionality of modules.
For non-standard applications, it is necessary to adapt the parameters to your system configuration.

| Name | Value | Internal <br> Value | Internal <br> Value, <br> Type | Default | Min. | Max. | EDS Slot <br> Index |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Module ID | Internal | $6125^{1}$ ) | WORD | 6125 <br> $0 x 17 E D$ | 0 | 65535 | xx01 |
| Ignore <br> module | No <br> Yes | 0 | 1 | BYTE | No <br> $(0 x 00)$ |  |  |
| Parameter <br> length | Internal | 1 | BYTE | 0 | 0 | 255 | xx02 $\left.{ }^{2}\right)$ |
| Check <br> supply | Off <br> On | 0 | BYTE | On <br> $0 x 01$ |  |  |  |

${ }^{1}$ ) with CS31 and addresses smaller than 70, the value is increased by 1
${ }^{2}$ ) Value is hexadecimal: HighByte is slot ( $x x: 0 \ldots 7$ ), LowByte is index ( $1 \ldots n$ ) GSD file:

| Ext_User_Prm_Data_Len $=$ | $0 \times 04$ |
| :--- | :--- |
| Ext_User_Prm_Data_Const $(0)=$ | $0 x E F, 0 \times 17,0 \times 00,1$ |
|  | $0 \times 01 ;$ |

## Diagnosis

| E1 ... E4 | d1 | d2 | d3 | d4 | Identifier $000 \text {... }$ $063$ |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err |  |  |
| $\begin{array}{\|l\|} \hline \text { Byte } 6 \\ \text { Bit } 6 \ldots \\ 7 \end{array}$ | - | Byte 3 | Byte 4 | Byte 5 | $\begin{array}{\|l\|} \hline \text { Byte } 6 \\ \text { Bit } 0 \ldots \\ 5 \end{array}$ |  |  |
| Class | Interface | Device | Module | Channel | Error Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) | ${ }^{4}$ ) |  |  |  |
| Module error |  |  |  |  |  |  |  |
| 3 | 14 | 1 ... 10 | 31 | 31 | 19 | Checksum error in the I/O module | Replace I/O module |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |
| 3 | 14 | 1 ... 10 | 31 | 31 | 43 | Internal error in the module | Replace I/O module |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |
| 3 | 14 | 1 ... 10 | 31 | 31 | 9 | Overflow diagnosis buffer | Restart |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |
| 4 | 14 | 1 ... 10 | 31 | 31 | 26 | Parameter error | Check master |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |
| 3 | 14 | 1 ... 10 | 31 | 31 | 11 | Process voltage too low | Check process voltage |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |

Remarks:

| ${ }^{1}$ ) | In AC500 the following interface identifier applies: <br> $14=$ I/O bus, 11 = COM1 (e.g. CS31 bus), 12 = COM2. <br> The PNIO diagnosis block does not contain this identifier. |
| :--- | :--- |
| ${ }^{2}$ ) | With "Device" the following allocation applies: <br> $31=$ module itself, $1 \ldots 10=$ decentralized communication interface module <br> $1 \ldots$... 10, ADR = Hardware address (e. g. of the DC551-CS31) |
| ${ }^{3}$ ) | With "Module" the following allocation applies depending on the master: <br> Module error: I/O bus or PNIO: 31 = module itself; COM1/COM2: $1 . . .10=$ <br> expansion 1 ... 10 <br> Channel error: I/O bus or PNIO = module type (2 = DO); COM1/COM2: $1 . .$. <br> $10=$ expansion 1 ... 10 |
| ${ }^{4}$ ) | In case of module errors, with channel "31 = Module itself" is output. |

## State LEDs

| LED |  | State | Color | LED = OFF | LED = ON |
| :---: | :---: | :---: | :---: | :---: | :---: |
| AHP Do571 | Outputs O0 ... O7 | Digital output | Yellow | Output is OFF | Output is ON |
| $10 口 10000000$ |  |  |  |  | (the output voltage is |
| - $\square_{0}$ |  |  |  |  | only displayed if the |
| - $\square 0$ (009 (100] |  |  |  |  | supply voltage of the |
| 800 2 20VVC 21 |  |  |  |  | module is ON) |

## Technical data

## Technical data of the module

The system data of AC500-eCo apply.
«2 Chapter 5.1.1 "System data AC500-eCo" on page 159
Only additional details are therefore documented below.

| Parameter | Value |  |
| :--- | :--- | :--- |
| Process supply voltage L+ |  |  |
|  | Connections | Terminal 20 for L+ (+24 V DC). The negative <br> pole is provided by the I/O bus. |
|  | Rated value | 24 V DC |
|  | Current consumption via L+ | 50 mA |
|  | Inrush current (at power-up) | $0.0035 \mathrm{~A}^{2} \mathrm{~s}$ |
|  | Protection against reversed voltage | $5 \%$ |
| Current consumption from 24 V DC power <br> supply at the L+/UP and M/ZP terminals of the <br> CPU/communication interface module | Yes |  |
| Galvanic isolation | Recommended; the outputs must be pro- <br> tected by a 3 A fast-acting fuse |  |
| Isolated groups | Yes, between the output group and the rest <br> of the module |  |
| Surge-voltage (max.) | 2 (4 channels per group) |  |
| Max. power dissipation within the module | 35 V DC for 0.5 s |  |
| Weight | 2.0 W |  |
| Mounting position | Ca. 150 g |  |
| Cooling | Horizontal or vertical |  |

No effects of multiple overloads

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an external fuse.

## Technical data of the digital outputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 8 normally-open relay outputs |
| Distribution of the channels into groups | 2 (4 channels per group) |
| Connection of the channels O0 ... O3 | Terminals $10 . . .13$ |
| Connection of the channels O4 ... O7 | Terminals $15 . . .18$ |
| Reference potential for the channels O0 ... O3 | Terminal 14 (signal name R0 ... R3) |
| Reference potential for the channels O4 ... O7 | Terminal 19 (signal name R4 ... R7) |
| Relay coil power supply | Terminal 20 (positive pole of the process supply voltage, signal name L+). The negative pole is provided by the I/O bus. |
| Indication of the output signals | 1 yellow LED per channel; the LED is on when the output signal is high (signal 1 ) and the module is powered via the I/O bus |
| Way of operation | Non-latching type |
| Relay output voltage |  |
| Rated value | 24 V DC / 24 V AC or 120/240 V AC |
| Output delay |  |
| Switching 0 to 1 (max.) | Typ. 10 ms |
| Switching 1 to 0 (max.) | Typ. 10 ms |
| Output data length | 1 byte |
| Output current |  |
| Rated current per channel (max.) | 2.0 A (24 V DC / 24 V AC / 48 V AC / 120 V AC / 240 V AC, only resistive loads) 2.0 A (24 V AC / 48 V AC / 120 V AC, only pilot duty) <br> 1.5 A (240 V AC, only pilot duty) |
| Rated current per group (max.) | 8 A |
| Lamp load (max.) | 200 W (230 V AC), 30 W (24 V DC) |
| Spark suppression with inductive AC loads | Must be performed externally according to driven load specification |
| Switching Frequencies |  |
| With resistive loads | Max. 1 Hz |
| With inductive loads | On Request |
| With lamp loads | Max. 1 Hz |
| Output type | Non-protected |
| Protection type | External fuse ${ }^{1}$ ) |
| Rated protection fuse | 5 A fast |
| Short-circuit-proof / Overload-proof | No, should be provided by an external fuse or circuit breaker |
| Overload message | No |
| Output current limitation | No |
| Connection of 2 outputs in parallel | Not possible |
| Lifetime of relay contacts (cycles) | 100.000 at rated load |


| Parameter | Value |
| :--- | :--- |
| Max. cable length |  |
|  | Shielded |
|  | Unshielded |

${ }^{1}$ ) Per group in case of group fuse protection. For each channel in case of channel-by-channel fuse protection. The maximum current per group must not be exceeded.

## Dimensions



The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1TNE 968 902 R2202 | DO571, digital output module, 8 DO, <br> relay output | Active |
| 1TNE 968 901 R3102 | Terminal block TA563-11, 11 pins, <br> screw front, cable side, 6 pieces per <br> unit | Active |
| 1TNE 968901 R3104 | Terminal block TA564-11, 11 pins, <br> screw front, cable front, 6 pieces per <br> unit | Active |
| 1TNE 968 901 R3106 | Terminal block TA565-11, 11 pins, <br> spring front, cable front, 6 pieces per <br> unit | Active |

> *) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

## DO572 - Digital output module

## Features

- 8 digital triac outputs ( O 0 ... O ) in 8 groups
- 120/240 V AC
- Module-wise galvanically isolated


1 I/O bus
28 yellow LEDs to display the signal states of the outputs 00 ... 07
3 Terminal number
4 Allocation of signal name
5 Terminal block for output signals (9-pin)
6 Terminal block for output signals (11-pin)
72 holes for wall-mounting with screws
8 DIN rail

## Intended purpose

The device can be used as a decentralized I/O extension module for S 500 communication interface modules (e. g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs.

The outputs are group-wise galvanically isolated from each other.
All other circuitry of the module is galvanically isolated from the outputs.

The I/O module must not be used as a decentralized I/O module with CI590-CS31-HA communication interface modules.

## Functionality

| Parameter | Value |
| :--- | :--- |
| LED displays | For signal states |
| Internal power supply | Via I/O bus |
| External power supply | Not necessary |

## Connections

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

The connection is carried out by using removable 9-pin and 11-pin terminal blocks. These terminal blocks differ in their connection system (spring terminals or screw terminals, cable mounting from the front or from the side). The terminal blocks are not included in the module's scope of delivery and must be ordered separately.


Fig. 58: Internal construction of the digital outputs

Table 132: Assignment of the terminals:

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 1 | O0 | Output signal O0 |
| 2 | N0 | Neutral conductor for the <br> output signal O0 |
| 3 | O1 | Output signal O1 |
| 4 | N1 | Neutral conductor for the <br> output signal O1 |
| 5 | --- | Reserved |
| 6 | O2 | Output signal O2 |
| 7 | O3 | Neutral conductor for the <br> output signal O2 |
| 8 | N3 | Output signal O3 |
| 9 | --- | Neutral conductor for the <br> output signal O3 |
| 10 | O4 | Reserved |
| 11 |  | Output signal O4 |


| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 12 | N4 | Neutral conductor for the <br> output signal O4 |
| 13 | O5 | Output signal O5 |
| 14 | N5 | Neutral conductor for the <br> output signal O5 |
| 15 | O6 | Reserved |
| 16 | N6 | Output signal O6 |
| 17 | O7 | Neutral conductor for the <br> output signal O6 |
| 18 | N7 | Output signal O7 |
| 19 | --- | Neutral conductor for the <br> output signal O7 |
| 20 | Reserved |  |

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a communication interface module or a CPU). Thus, the current consumption from $24 \vee$ DC power supply at the terminals L+/UP and M/ZP of the CPU/communication interface module increases by 10 mA per DO572.

An external power supply connection is not needed.

## WARNING!

## Risk of death by electric shock!

Hazardous voltages can be present at the terminals of the module.
Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.

## WARNING!

## Removal/Insertion under power

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.
Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.
Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.
The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

## Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.


Fig. 59: Connection of the module

## NOTICE!

## Risk of damaging the PLC modules!

The PLC modules will be irreparably damaged if a voltage $>240 \mathrm{~V}$ is connected.

Make sure that all inputs are fed from the same phase. The module must not be connected to a 400 V voltage.

The module provides several diagnosis functions $\stackrel{y}{ }{ }^{\circ}$ Chapter 5.2.4.3.1.9.7 "Diagnosis" on page 474.
The meaning of the LEDs is described in the section State LEDs ${ }_{y}{ }^{4}$ Chapter 5.2.4.3.1.9.8 "State LEDs" on page 475.

## I/O configuration

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or communication interface module) during power-up of the system.
Hence, replacing I/O modules is possible without any re-parameterization via software.

If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.

## Parameterization

The arrangement of the parameter data is performed with Automation Builder software.
The parameter data directly influences the functionality of modules.
For non-standard applications, it is necessary to adapt the parameters to your system configuration.

| Name | Value | Internal <br> Value | Internal <br> Value, <br> Type | Default | Min. | Max. | EDS SIot <br> Index |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Module ID | Internal | $6130^{1}$ ) | WORD | 6130 <br> $0 x 17 F 2$ | 0 | 65535 | $x x 01$ |
| Ignore <br> module | No <br> Yes | 0 | BYTE | No <br> $(0 x 00)$ |  |  |  |
| Parameter <br> length 2) | Internal | $1-$ CPU | BYTE | 0 | 0 | 255 | $\left.x x 02^{3}\right)$ |


| $\left.{ }^{1}\right)$ | With CS31 and addresses smaller than 70, the value is increased by 1 |
| :--- | :--- |
| $\left.{ }^{2}\right)$ | The module has no additional user-configurable parameters |
| $\left.{ }^{3}\right)$ | Value is hexadecimal: HighByte is slot (xx: $0 \ldots 7)$, LowByte is index $(1 \ldots$ <br> $n)$ |

GSD file:

| Ext_User_Prm_Data_Len $=$ | $0 x 03$ |
| :--- | :--- |
| Ext_User_Prm_Data_Const $(0)=$ | $0 x F 3,0 x 17,0 \times 00 ;$ |

## Diagnosis

| E1 ... E4 | d1 | d2 | d3 | d4 | Identifier $000 \ldots 063$ | AC500- <br> Display | <- Display in |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err | $\begin{array}{\|l} \hline \text { PS501 } \\ \text { PLC } \\ \text { Browser } \end{array}$ |  |
| Byte 6 <br> Bit 6 ... 7 | - | Byte 3 | Byte 4 | Byte 5 | Byte 6 <br> Bit 0 ... 5 | PNIO diagnosis block |  |
| Class | Interface | Device | Module | Channel | Error Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | $\left.{ }^{3}\right)$ | ${ }^{4}$ ) |  |  |  |
| Module error |  |  |  |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 19 | Checksum error in the I/O module | Replace I/O module |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 43 | Internal error in the module | Replace I/O module |
|  | 11/12 | ADR | 1... 10 |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 9 | Overflow diagnosis buffer | Restart |
|  | $11 / 12$ | ADR | 1... 10 |  |  |  |  |
| 4 | 14 | $1 \ldots 10$ | 31 | 31 | 26 | Parameter error | Check master |
|  | 11/12 | ADR | $1 . .10$ |  |  |  |  |

Remarks:

| ${ }^{1}$ ) | In AC500 the following interface identifier applies: <br> $14=$ I/O bus, 11 = COM1 (e.g. CS31-Bus), 12 = COM2. <br> The PNIO diagnosis block does not contain this identifier. |
| :--- | :--- |
| ${ }^{2}$ ) | With "Device" the following allocation applies: <br> $31=$ module itself, 1 ... 10 = decentralized communication interface module 1 ... 10, <br> ADR = hardware address (e. g. of the DC551-CS31) |
| ${ }^{3}$ ) | With "Module" the following allocation applies depending on the master: <br> Module error: I/O bus or PNIO: 31 = module itself; COM1/COM2: $1 \ldots 10=$ expan- <br> sion 1 ... 10 <br> Channel error: I/O bus or PNIO = module type (2 = DO); COM1/COM2: $1 \ldots 10=$ <br> expansion 1 ... 10 |
| ${ }^{4}$ ) | In case of module errors, with channel "31 = Module itself" is output. |

State LEDs

| LED |  | State | Color | LED = OFF | LED = ON |
| :---: | :---: | :---: | :---: | :---: | :---: |
| A13 DO572 | Outputs O0 ...$07$ | Digital output | Yellow | Output is OFF | Output is ON |
| - $\square^{1} 0000040$ |  |  |  |  |  |
| - 0000050 |  |  |  |  |  |
| - 002000 |  |  |  |  |  |
| - -a 03000 |  |  |  |  |  |
| 800.tase 100280Vac 0.3 |  |  |  |  |  |

## Technical data

## Technical data of the module

The system data of AC500-eCo apply.
\& Chapter 5.1.1 "System data AC500-eCo" on page 159
Only additional details are therefore documented below.

| Parameter | Value |
| :--- | :--- |
| Galvanic isolation | Yes, between the channels and the rest of the <br> module |
| Isolated groups | 8 (1 channel per group) |
| Current consumption from 24 V DC power <br> supply at the L+/UP and M/ZP terminals of the <br> CPU/communication interface module | Ca. 10 mA |
| Max. power dissipation within the module | On Request |
| Weight | ca. 120 g |
| Mounting position | Horizontal or vertical |
| Cooling | The natural convection cooling must not be <br> hindered by cable ducts or other parts in the <br> control cabinet. |

No effects of multiple overloads

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an external fuse.

## Technical data of the digital outputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 8 triac outputs |
| Distribution of the channels into groups | 8 groups (1 channel per group) |
| Connection of the channels O0 to O7 | Terminals 1, 3, 5, 7, 10, 12, 14, 16 |
| Reference potential for the channels O0 to O7 | Terminals 2, 4, 6, 8, 11, 13, 15, 17 |
| Output voltage for signal 1 | On Request |
| Max. leakage current with signal 0 | 1.1 mA root mean square at 132 V AC and |
|  | 1.8 mA root mean square at 264 V AC |
| Output voltage |  |
|  | Rated value |


| Parameter | Value |
| :---: | :---: |
| Indication of the output signals | 1 yellow LED per channel; the LED is on when the output signal is high (signal 1) and the module is powered via the I/O bus |
| Way of operation | Non-latching type |
| Output delay | On Request |
| Output data length | 1 byte |
| Output current |  |
| Rated current per channel (max.) | 0.3 A |
| Rated current per group (max.) | 0.3 A |
| Surge current (max.) | On request |
| Lamp load (max.) | On request |
| Spark suppression with inductive AC loads | Must be performed externally according to driven load specification |
| Switching Frequencies |  |
| With resistive loads | Max. 10 Hz |
| With inductive loads | Not applicable |
| With lamp loads | Max. 10 Hz |
| Output type | Non-protected |
| Protection type | External fuse on each channel |
| Rated protection fuse | 2 A fast |
| Short-circuit-proof / Overload-proof | No, should be provided by an external fuse or circuit breaker |
| Overload message | No |
| Output current limitation | No |
| Resistance to feedback against 230 V AC | No |
| Connection of 2 outputs in parallel | Not applicable |
| Max. cable length |  |
| Shielded | 500 m |
| Unshielded | 150 m |

## Dimensions



The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1TNE 968 902 R2203 | DO572, digital output module, 8 DO, <br> triac output | Active |
| 1TNE 968 901 R3101 | Terminal block TA563-9, 9 pins, screw <br> front, cable side, 6 pieces per unit | Active |
| 1TNE 968 901 R3102 | Terminal block TA563-11, 11 pins, <br> screw front, cable side, 6 pieces per <br> unit | Active |
| 1TNE 968 901 R3103 | Terminal block TA564-9, 9 pins, screw <br> front, cable front, 6 pieces per unit | Active |
| 1TNE 968 901 R3104 | Terminal block TA564-11, 11 pins, <br> screw front, cable front, 6 pieces per <br> unit | Active |
| 1TNE 968 901 R3105 | Terminal block TA565-9, 9 pins, spring <br> front, cable front, 6 pieces per unit | Active |
| 1TNE 968 901 R3106 | Terminal block TA565-11, 11 pins, <br> spring front, cable front, 6 pieces per <br> unit | Active |

> *) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

## DO573 - Digital output module

## Features

- 16 digital normally open relay outputs 24 V DC or 100 V AC ... 240 V AC (NOO ... NO15) in 2 groups, 2 A max.
- Group-wise galvanically isolated


1 I/O bus
216 yellow LEDs to display the signal states of the outputs $\mathrm{O} 0 \ldots \mathrm{O} 15$
3 Terminal number
4 Allocation of signal name
5 Terminal block for output signals (9-pin)
6 Terminal block for output signals (11-pin)
72 holes for wall-mounting with screws
8 DIN rail

## Intended purpose

The device can be used as a decentralized I/O extension module for S 500 communication interface modules (e. g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs.

The outputs are group-wise galvanically isolated from each other.
All other circuitry of the module is galvanically isolated from the outputs.

The I/O module must not be used as a decentralized I/O module with CI590-CS31-HA communication interface modules.

## Functionality

| Parameter | Value |
| :--- | :--- |
| LED displays | For signal states |
| Internal power supply | Via I/O bus |
| External power supply | Via the terminals L+ (process voltage 24 V DC) and M (0 V <br> DC); the M terminal is connected to the M terminal of the CPU <br> via the I/O bus |

## Connections

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

The connection is carried out by using removable 9-pin and 11-pin terminal blocks. These terminal blocks differ in their connection system (spring terminals or screw terminals, cable mounting from the front or from the side). The terminal blocks are not included in the module's scope of delivery and must be ordered separately.

| $\frac{1}{\square} \longrightarrow 1$ | NOO |
| :---: | :---: |
| $\xrightarrow{1} \longrightarrow 2$ | NO1 |
| $\stackrel{1}{\square}$ | NO2 |
| $\xrightarrow{\square} 4$ | NO3 |
| $\longrightarrow 5$ | NO4 |
| $1-\bigcirc 6$ | NO5 |
|  |  |
| $\square \longrightarrow 7$ | NO6 |
| $\underset{\sim}{1}$ | NO7 |
| $\bigcirc 9$ | R0.. 7 |
| $\underset{\square}{1} \longrightarrow 10$ | NO8 |
| $\bigcirc$ | NO9 |
| $\bigcirc 12$ | NO10 |
| $\bigcirc 13$ | NO11 |
| $\bigcirc 14$ | NO12 |
| $\bigcirc 15$ | NO13 |
|  | NO14 |
| $\square \longrightarrow 16$ | NO14 |
| $\underset{\square}{1} \longrightarrow 17$ | NO15 |
| $\bigcirc 18$ | R8.. 15 |
| - 19 | L+ |
| - 20 | M |

Fig. 60: Internal construction of the digital outputs

Table 133: Assignment of the terminals:

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 1 | NO0 | Normally-open contact of the output NO0 |
| 2 | NO1 | Normally-open contact of the output NO1 |
| 3 | NO2 | Normally-open contact of the output NO2 |
| 4 | NO3 | Normally-open contact of the output NO3 |
| 5 | NO4 | Normally-open contact of the output NO4 |
| 6 | NO5 | Normally-open contact of the output NO5 |
| 7 | NO6 | Normally-open contact of the output NO6 |
| 8 | RO..7 | Normally-open contact of the output NO7 |
| 9 | NO9 | Output common for signals NO0 to NO7 |
| 10 | NO10 | Normally-open contact of the output NO8 |
| 11 | NO11 | Normally-open contact of the output NO9 |
| 12 |  | Normally-open contact of the output NO10 |
| 13 | Normally-open contact of the output NO11 |  |


| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 14 | NO12 | Normally-open contact of the output NO12 |
| 15 | NO13 | Normally-open contact of the output NO13 |
| 16 | NO14 | Normally-open contact of the output NO14 |
| 17 | NO15 | Normally-open contact of the output NO15 |
| 18 | R8 ... 15 | Output common for signals NO8 to NO15 |
| 19 | L+ | Process voltage L+ (24 V DC) |
| 20 | M | Process voltage M (0 V DC) |

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a communication interface module or a CPU). Thus, the current consumption from 24 V DC power supply at the terminals L+/UP and M/ZP of the CPU/communication interface module increases by 5 mA per DO573.

The external power supply connection is carried out via the $\mathrm{L}+(+24 \mathrm{~V} \mathrm{DC}$ ) and the $\mathrm{M}(0 \mathrm{~V}$ DC) terminals. The $M$ terminal is electrically interconnected to the M/ZP terminal of the CPU/ communication interface module.

## WARNING!

Risk of death by electric shock!
Hazardous voltages can be present at the terminals of the module.
Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.

For screw-type terminals only:

## WARNING!

For screw terminals only: Danger of death by electric shock!
The IP 20 protection degree is only provided if all terminal screws are tightened.
Tighten all screws of unused load terminals of relay outputs if voltages $>24 \mathrm{~V}$ are connected to the relay group.

## WARNING!

## Removal/Insertion under power

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.
Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

## Risk of damaging the I/O module!

The outputs are not protected against short circuit and overload.

- Never short-circuit or overload the outputs.
- Never connect inductive loads without an external suppression against voltage peaks due to inductive kickback.
- Never connect voltages > 240 V . All outputs must be supplied from the same phase.
- Use an external 5 A fast protection fuse for the outputs.


## NOTICE!

## Risk of damaging the PLC modules!

The PLC modules can be damaged by overload.
Make sure that the total current of each output common terminal (R0.. 7 and R8..15) does not exceed 10 A .

Never connect total currents > 10 A per group.
If the group fuse protection is not sufficient, then individual fuse protection of the outputs should be used.

The following figure shows the connection of the module:


Fig. 61: Connection of $24 V D C$ actuators


Fig. 62: Connection of $100-240$ V AC actuators
The module provides several diagnosis functions.
The meaning of the LEDs is described in the section State LEDs $\Leftrightarrow$ Chapter 5.2.4.3.1.9.8 "State LEDs" on page 475.


Fig. 63: Power supply - the negative connection is realized via the I/O bus

The $L+$ connection of the $D 0573$ and the 24 V supply of the CPU/communication interface module must be connected to the same 24 V power supply .

## I/O configuration

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or communication interface module) during power-up of the system.

Hence, replacing I/O modules is possible without any re-parameterization via software.


If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.

## Parameterization

The arrangement of the parameter data is performed with Automation Builder software.
The parameter data directly influences the functionality of modules.
For non-standard applications, it is necessary to adapt the parameters to your system configuration.

| Name | Value | Internal <br> Value | Internal <br> Value, <br> Type | Default | Min. | Max. | EDS SIot <br> Index |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Module ID | Internal | $6150^{1}$ ) | WORD | 6150 <br> $0 x 1806$ | 0 | 65535 | $x x 01$ |
| Ignore <br> module | No <br> Yes | 0 | 1 | BYTE | No <br> $(0 x 00)$ |  |  |
| Parameter <br> length | Internal | 1 | BYTE | 0 | 0 | 255 | $\left.x x 02^{2}\right)$ |
| Check <br> supply | Off <br> On | 0 | BYTE | On <br> $0 x 01$ |  |  |  |

${ }^{1}$ ) with CS31 and addresses less than 70, the value is increased by 1
${ }^{2}$ ) Value is hexadecimal: HighByte is slot (xx: $0 \ldots 7$ ), LowByte is index (1 ... n) GSD file:

| Ext_User_Prm_Data_Len $=$ | $0 \times 070 \times 18,0 \times 07,0 \times 00,0 \times 03,0 \times 01,0 \times 00$, |
| :--- | :--- |
| Ext_User_Prm_Data_Const $(0)=$ | $0 \times 00 ;$ |

## Diagnosis

| E1 ... E4 | d1 | d2 | d3 | d4 | $\begin{aligned} & \text { Identi- } \\ & \text { fier } \\ & 000 \ldots \\ & 063 \end{aligned}$ |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err | <- Display in |  |
| $\begin{array}{\|l\|} \hline \text { Byte } 6 \\ \text { Bit } 6 \ldots \\ 7 \end{array}$ | - | Byte 3 | Byte 4 | Byte 5 | $\begin{aligned} & \text { Byte } 6 \\ & \text { Bit } 0 \ldots \\ & 5 \end{aligned}$ |  |  |
| Class | Interface | Device | Module | Channel | Error-Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | $\left.{ }^{3}\right)$ | ${ }^{4}$ ) |  |  |  |
| Module error |  |  |  |  |  |  |  |
| 3 | 14 | $1 \ldots 10$ | 31 | 31 | 19 | Checksum error in the I/O module | Replace I/O module |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |
| 3 | 14 | $1 \ldots 10$ | 31 | 31 | 43 | Internal error in the module | Replace I/O module |
|  | 11/12 | ADR | $1 . .10$ |  |  |  |  |
| 3 | 14 | $1 \ldots 10$ | 31 | 31 | 9 | Overflow diagnosis buffer | Restart |
|  | 11/12 | ADR | 1... 10 |  |  |  |  |
| 4 | 14 | $1 \ldots 10$ | 31 | 31 | 26 | Parameter error | Check master |
|  | 11 / 12 | ADR | 1... 10 |  |  |  |  |


| E1 ... E4 | d1 | d2 | d3 | d4 | Identi- <br> fier <br> 000 <br> 063 | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err |  |  |
| $\begin{aligned} & \hline \text { Byte } 6 \\ & \text { Bit } 6 \ldots \\ & 7 \end{aligned}$ | - | Byte 3 | Byte 4 | Byte 5 | Byte 6 <br> Bit 0 ... <br> 5 |  |  |
| Class | Interface | Device | Module | Channel | Error-Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) | ${ }^{4}$ ) |  |  |  |
| Module error |  |  |  |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 11 | Process voltage too low | Check process voltage |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |

## Remarks:

| $\left.{ }^{1}\right)$ | In AC500 the following interface identifier applies: <br> $14=$ I/O bus, 11 = COM1 (e.g. CS31-Bus), 12 = COM2. <br> The PNIO diagnosis block does not contain this identifier. |
| :--- | :--- |
| ${ }^{2}$ ) | With "Device" the following allocation applies: <br> $31=$ Module itself, 1 ... 10 = decentralized communication interface module $1 \ldots 10$, <br> ADR = Hardware address (e. g. of the DC551-CS31) |
| $\left.{ }^{3}\right)$ | With "Module" the following allocation applies depending on the master: <br> Module error: I/O bus or PNIO: 31 = module itself; COM1/COM2: $1 \ldots 10=$ expan- <br> sion $1 \ldots 10$ <br> Channel error: I/O bus or PNIO = module type (2 = DO); COM1/COM2: $1 \ldots 10=$ <br> expansion $1 \ldots 10$ |
| $\left.{ }^{4}\right)$ | In case of module errors, with channel "31 = Module itself" is output. |

## State LEDs

| LED |  | State | Color | LED = OFF | LED = ON |
| :---: | :---: | :---: | :---: | :---: | :---: |
| A13 Do573 | Outputs | Digital output | Yellow | Output is | Output is ON |
|  | NO0 ... NO15 |  |  |  | (the output voltage is |
|  |  |  |  |  | only displayed if the |
|  |  |  |  |  | supply voltage of the |

## Technical data

## Technical data of the module

The system data of AC500-eCo apply.

② Chapter 5.1.1 "System data AC500-eCo" on page 159
Only additional details are therefore documented below.

| Parameter |  | Value |
| :--- | :--- | :--- |
| Process supply voltage L+ |  |  |
|  | Connections | Terminals 19 for L+ (+24 V DC) and 20 <br> for M (0 V DC) |
|  | Rated value | 24 V DC |
|  | Current consumption via L+ | 50 mA |
|  | Max. ripple | $5 \%$ |
|  | Protection against reversed voltage | Yes |
|  | Rated protection fuse for L+ | Recommended; the outputs must be <br> protected by an 5 A fast-acting fuse |
| Current consumption from 24 V DC power supply at <br> the L+/UP and M/ZP terminals of the CPU/communi- <br> cation interface module | Ca. 5 mA |  |
| Galvanic isolation | Yes, between the output groups and the <br> rest of the module |  |
| Isolated groups | 2 (8 channels per group) |  |
| Surge-voltage (max.) | $35 \mathrm{~V} \mathrm{DC} \mathrm{for} \mathrm{0.5} \mathrm{~s}$ |  |
| Max. power dissipation within the module | 2.0 W |  |
| Weight | Ca. 160 g |  |
| Mounting position | Horizontal or vertical |  |
| Cooling | The natural convection cooling must not <br> be hindered by cable ducts or other <br> parts in the control cabinet. |  |

No effects of multiple overloads

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an external fuse.

## Technical data of the digital outputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 16 normally-open relay outputs |
| Distribution of the channels into groups | 2 (8 channels per group) |
| Connection of the channels NO0 ... NO7 | Terminals $1 \ldots 8$ |
| Connection of the channels NO8 ... NO15 | Terminals $10 \ldots 17$ |
| Reference potential for the channels NO0 ... <br> NO7 | Terminal 9 (signal name R0 ... 7) |
| Reference potential for the channels NO8 ... <br> NO15 | Terminal 18 (signal name R8 ... 15) |
| Relay coil power supply | Terminals 19 and 20 (signal names L+ and <br> M) |
| Indication of the output signals | 1 yellow LED per channel; the LED is on <br> when the output signal is high (signal 1) and <br> the module is powered via the I/O bus |


| Parameter | Value |
| :---: | :---: |
| Way of operation | Non-latching type |
| Relay output voltage |  |
| Rated value | 24 V DC or 120/240 V AC |
| Output delay |  |
| Switching 0 to 1 (max.) | Typ. 10 ms |
| Switching 1 to 0 (max.) | Typ. 10 ms |
| Output data length | 2 bytes |
| Output current |  |
| Rated current per channel (max.) | 2.0 A (24 V DC / 24 V AC / 48 V AC / 120 V AC / 240 V AC, only resistive loads) 2.0 A (24 V AC / 48 V AC / 120 V AC, only pilot duty) <br> 1.5 A (240 V AC, only pilot duty) |
| Rated current per group (max.) | 10 A |
| Lamp load (max.) | 200 W (230 V AC), 30 W (24 V DC) |
| Spark suppression with inductive AC loads | Must be performed externally according to driven load specification |
| Switching Frequencies |  |
| With resistive loads | Max. 1 Hz |
| With inductive loads | On Request |
| With lamp loads | Max. 1 Hz |
| Output type | Non-protected |
| Protection type | External fuse ${ }^{1}$ ) |
| Rated protection fuse | 5 A fast |
| Short-circuit-proof / Overload-proof | No, should be provided by an external fuse or circuit breaker |
| Overload message | No |
| Output current limitation | No |
| Connection of 2 outputs in parallel | Not possible |
| Lifetime of relay contacts (cycles) | 100.000 at rated load |
| Max. cable length |  |
| Shielded | 500 m |
| Unshielded | 150 m |

[^11]
## Dimensions



The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 231300 R0000 | DO573, digital output module, 16 DO, <br> relay output | Active |
| 1TNE 968 901 R3101 | Terminal block TA563-9, 9 pins, screw <br> front, cable side, 6 pieces per unit | Active |
| 1TNE 968 901 R3102 | Terminal block TA563-11, 11 pins, <br> screw front, cable side, 6 pieces per <br> unit | Active |
| 1TNE 968 901 R3103 | Terminal block TA564-9, 9 pins, screw <br> front, cable front, 6 pieces per unit | Active |
| 1TNE 968901 R3104 | Terminal block TA564-11, 11 pins, <br> screw front, cable front, 6 pieces per <br> unit | Active |
| 1TNE 968 901 R3105 | Terminal block TA565-9, 9 pins, spring <br> front, cable front, 6 pieces per unit | Active |
| 1TNE 968 901 R3106 | Terminal block TA565-11, 11 pins, <br> spring front, cable front, 6 pieces per <br> unit | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

## DX561-Digital input/output module

## Features

- 8 digital inputs 24 V DC ( $10 \ldots \mathrm{I}$ ) in 1 group
- 8 digital transistor outputs 24 V DC ( $\mathrm{O} 0 \ldots \mathrm{O}$ ) in 1 group
- Group-wise galvanically isolated


1 I/O bus
28 yellow LEDs to display the signal states of the inputs 10 ... 17
38 yellow LEDs to display the signal states of the outputs 00 ... O7
4 Terminal number
5 Allocation of signal name
6 Terminal block for input signals (9-pin)
7 Terminal block for output signals (11-pin)
82 holes for wall-mounting with screws
9 DIN rail

## Intended purpose

The device can be used as a decentralized I/O extension module for S500 communication interface modules (e. g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs.

The inputs and outputs are group-wise galvanically isolated from each other.
All other circuitry of the module is galvanically isolated from the inputs.

The I/O module must not be used as a decentralized I/O module with CI590-CS31-HA communication interface modules.

## Functionality

| Parameter | Value |
| :--- | :--- |
| LED displays | For signal states |
| Internal power supply | Via I/O bus |
| External power supply | Via the terminals ZP and UP (process voltage 24 V DC) |

## Connections

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

The connection is carried out by using removable 9-pin and 11-pin terminal blocks. These terminal blocks differ in their connection system (spring terminals or screw terminals, cable mounting from the front or from the side). The terminal blocks are not included in the module's scope of delivery and must be ordered separately.


Fig. 64: Internal construction of the digital inputs and outputs

Table 134: Assignment of the terminals:

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 1 | CO ... 7 | Input common for signals IO $\ldots$ <br> I7 |
| 2 | IO | Input signal IO |
| 3 | I1 | Input signal I1 |
| 4 | I2 | Input signal I2 |
| 5 | I3 | Input signal I3 |
| 6 | I4 | Input signal I4 |
| 7 | I5 | Input signal I5 |
| 8 | I6 | Input signal I6 |
| 9 | I7 | Input signal I7 |
| 10 | --- | Reserved |
| 11 | O0 | Output signal O0 |
| 12 | O1 | Output signal O1 |
| 13 | O2 | Output signal O2 |


| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 14 | O3 | Output signal O3 |
| 15 | O4 | Output signal O4 |
| 16 | O5 | Output signal O5 |
| 17 | O6 | Output signal O6 |
| 18 | O7 | Output signal O7 |
| 19 | UP | Process voltage UP +24 V DC |
| 20 | ZP | Process voltage ZP 0 V DC |

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a communication interface module or a CPU). Thus, the current consumption from 24 V DC power supply at the terminals L+/UP and M/ZP of the CPU/communication interface module increases by 10 mA per DX561.

The external power supply connection is carried out via the UP (+24 V DC) and ZP (0 V DC) terminals.

## WARNING!

## Removal/Insertion under power

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.
Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.
The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

## Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

The digital inputs can be used as source inputs or as sink inputs.

## NOTICE!

## Risk of malfunctions in the plant!

A ground fault, e. g. caused by a damaged cable insulation, can bridge switches accidentally.
Use sink inputs when possible or make sure that, in case of error, there will be no risks to persons or plant.


Fig. 65: Connection of inputs to the digital input/output module - sink inputs


Fig. 66: Connection of inputs to the digital input/output module - source inputs


Fig. 67: Connection of the outputs to the module

## NOTICE!

## Risk of malfunctions in the plant!

The outputs may switch on for a period of 10 to $50 \mu$ s if the process supply voltage UP/ZP is switched on.
This must be considered in the planning of the application.

## - NOTICE!

## Risk of damaging the I/O module!

The outputs are not protected against short circuits and overload.

- Never short-circuit or overload the outputs.
- Never connect the outputs to other voltages.
- Use an external 3 A fast-protection fuse for the outputs.

The module provides several diagnosis functions ${ }^{\mu}$ Chapter 5.2.4.3.1.11.7 "Diagnosis" on page 498.

The meaning of the LEDs is described in the Displays section ${ }^{\&} \geqslant$ Chapter 5.2.4.3.1.11.8 "State LEDs" on page 499 chapter.

## I/O configuration

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or communication interface module) during power-up of the system.
Hence, replacing I/O modules is possible without any re-parameterization via software.

> If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.

## Parameterization

The arrangement of the parameter data is performed with Automation Builder software.
The parameter data directly influences the functionality of modules.
For non-standard applications, it is necessary to adapt the parameters to your system configuration.

| Name | Value | Internal <br> Value | Internal <br> Value, <br> Type | Default | Min. | Max. | EDS Slot <br> Index |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Module ID | Internal | $6135^{1}$ ) | WORD | 6135 <br> $0 x 17 F 7$ | 0 | 65535 | $x x 01$ |
| Ignore <br> module | No <br> Yes | 0 | BYTE | No <br> $(0 x 00)$ |  |  |  |
| Parameter <br> length | Internal | 1 | BYTE | 0 | 0 | 255 | $\left.x x 02^{2}\right)$ |

${ }^{1}$ ) with CS31 and addresses smaller than 70, the value is increased by 1
${ }^{2}$ ) Value is hexadecimal: HighByte is slot (xx: $0 \ldots 7$ ), LowByte is index ( $1 . . . n$ ) GSD file:

| Ext_User_Prm_Data_Len $=$ | $0 x 03$ |
| :--- | :--- |
| Ext_User_Prm_Data_Const $(0)=$ | $0 x F 8,0 \times 17,0 \times 00,1$ |
| $(0)=$ | $0 x 01 ;$ |

Diagnosis

| E1 ... E4 | d1 | d2 | d3 | d4 | Identifier $000 \ldots 063$ | AC500Display | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err | PS501 PLC Browser |  |  |
| $\begin{array}{\|l\|} \hline \text { Byte } 6 \\ \text { Bit } 6 \ldots 7 \end{array}$ | - | Byte 3 | Byte 4 | Byte 5 | Byte 6 <br> Bit 0 ... 5 | PNIO diagnosis block |  |  |
| Class | Interface | Device | Module | Channel | Error Identifier | Error message |  | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) | $\left.{ }^{4}\right)$ |  |  |  |  |
| Module error |  |  |  |  |  |  |  |  |
| 3 | 14 | 1 ... 10 | 31 | 31 | 19 | Checksum error in the I/O module |  | Replace I/O module |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |  |  |  |
| 3 | 14 | 1 ... 10 | 31 | 31 | 43 | Internal error in the module |  | Replace I/O module |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |  |  |  |
| 3 | 14 | 1 ... 10 | 31 | 31 | 9 | Overflow diagnosis buffer |  | Restart |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |  |  |  |
| 4 | 14 | 1 ... 10 | 31 | 31 | 26 | Parameter error |  | Check master |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |  |  |  |

Remarks:

| ${ }^{1}$ ) | In AC500 the following interface identifier applies: 14 = I/O bus, 11 = COM1 (e.g. CS31 bus), $12=$ COM2. <br> The PNIO diagnosis block does not contain this identifier. |
| :---: | :---: |
| ${ }^{2}$ ) | With "Device" the following allocation applies: <br> 31 = module itself, <br> 1 ... 10 = decentralized communication interface module 1 ... 10, <br> ADR = hardware address (e. g. of the DC551-CS31) |
| ${ }^{3}$ ) | With "Module" the following allocation applies depending on the master: <br> Module error: I/O bus or PNIO: 31 = module itself; COM1/COM2: 1 ... 10 = expansion 1 ... 10 <br> Channel error: I/O bus or PNIO = module type (2 = DO); COM1/COM2: 1 ... $10=$ expansion 1 ... 10 |
| ${ }^{4}$ ) | In case of module errors, with channel "31 = module itself" is output. |

State LEDs

| LED |  | State | Color | LED = OFF | LED = ON |
| :---: | :---: | :---: | :---: | :---: | :---: |
| ATH DX561 | Inputs $10 . . .17$ | Digital input | Yellow | Input is OFF | Input is ON |
|  | $\begin{array}{\|l} \text { Outputs O0 ... } \\ \text { O7 } \end{array}$ | Digital output | Yellow | Output is OFF | Output is ON |

## Technical data

## Technical data of the module

The system data of AC500-eCo apply.
\& Chapter 5.1.1 "System data AC500-eCo" on page 159
Only additional details are therefore documented below.

| Parameter | Value |  |
| :--- | :--- | :--- |
| Process supply voltage UP |  |  |
|  | Connections | Terminal 19 for UP (+24 V DC) and ter- <br> minal 20 for ZP (0 V DC) |
|  | Rated value | 24 V DC |
|  | Current consumption via UP terminal | $5 \mathrm{~mA}+$ max. 0.5 A per output |
|  | Max. ripple | $5 \%$ |
|  | Inrush current | Protection against reversed voltage |
| Rated protection fuse for UP | Yes <br> Recommended; the outputs must be pro- <br> tected by an 3 A fast-acting fuse |  |
| Current consumption from 24 V DC power supply <br> at the L+/UP and M/ZP terminals of the CPU/com- <br> munication interface module | Ca. 10 mA |  |
| Galvanic isolation | Yes, between the input group and the <br> output group and the rest of the module |  |
| Isolated groups | 2 groups (1 group for 8 input channels, 1 <br> group for 8 output channels) |  |
| Surge-voltage (max.) | 35 V DC for 0.5 s |  |
| Max. power dissipation within the module | 2.3 W |  |
| Weight | ca. 120 g |  |
| Mounting position | Horizontal or vertical |  |
| Cooling | The natural convection cooling must not <br> be hindered by cable ducts or other parts <br> in the control cabinet. |  |

No effects of multiple overloads

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an external fuse.

## Technical data of the digital inputs

| Parameter | Value |  |
| :---: | :---: | :---: |
| Number of channels per module | 8 |  |
| Distribution of the channels into groups | 1 group for 8 channels |  |
| Connections of the channels $10 \ldots 17$ | Terminals $2 \ldots 9$ |  |
| Reference potential for the channels I0 ... 17 | Terminal 1 |  |
| Indication of the input signals | 1 yellow LED per channel; the LED is ON when the input signal is high (signal 1) |  |
| Monitoring point of input indicator | LED is part of the input circuitry |  |
| Input type according to EN 61131-2 | Type 1 source | Type 1 sink |
| Input signal range | -24 V DC | +24 V DC |
| Signal 0 | -5V ... +3 V | -3V ... +5V |
| Undefined signal | -15V ... + 5 V | +5V ... +15 V |
| Signal 1 | -30 V ... -15V | +15V ... +30 V |
| Ripple with signal 0 | -5 V ... +3 V | -3 V ... +5 V |
| Ripple with signal 1 | -30 V ... -15 V | +15 V ... +30 V |
| Input current per channel |  |  |
| Input voltage +24 V | Typ. 5 mA |  |
| Input voltage +5 V | Typ. 1 mA |  |
| Input voltage +15 V | $>2.5 \mathrm{~mA}$ |  |
| Input voltage +30 V | $<8 \mathrm{~mA}$ |  |
| Max. permissible leakage current (at 2-wire proximity switches) | 1 mA |  |
| Input delay (0->1 or 1->0) | Typ. 8 ms |  |
| Input data length | 1 byte |  |
| Max. cable length |  |  |
| Shielded | 500 m |  |
| Unshielded | 300 m |  |

## Technical data of the digital outputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 8 transistor outputs (24 V DC, 0.5 A max.) |
| Distribution of the channels into groups | 1 group of 8 channels |
| Connection of the channels $\mathrm{O} 0 \ldots \mathrm{O} 7$ | Terminals $11 \ldots 18$ |
| Reference potential for the channels $\mathrm{O} 0 \ldots \mathrm{O} 7$ | Terminal 20 (negative pole of the process <br> voltage, name ZP) |
| Common power supply voltage | Terminal 19 (positive pole of the process <br> voltage, name UP) |
| Indication of the output signals | 1 yellow LED per channel; the LED is on <br> when the output signal is high (signal 1) <br> and the module is powered via the I/O bus |
| Monitoring point of output indicator | Controlled together with transistor |


| Parameter | Value |
| :---: | :---: |
| Way of operation | Non-latching type |
| Max. output voltage at signal 1 | 20 V DC at max. current consumption |
| Output delay |  |
| 0 to 1 | $50 \mu \mathrm{~s}$ |
| 1 to 0 | $200 \mu \mathrm{~s}$ |
| Output data length | 1 byte |
| Output current |  |
| Rated current per channel (max.) | 0.5 A at UP 24 V DC |
| Rated current per group (max.) | 4 A |
| Rated current (all channels together, max.) | 4 A |
| Lamp load (max.) | 5 W |
| Max. leakage current with signal 0 | 0.5 mA |
| Output type | Non-protected |
| Protection type | External fuse on each channel |
| Rated protection fuse (for each channel) | 3 A fast |
| Demagnetization when inductive loads are switched off | Must be performed externally according to driven load specification |
| Switching Frequencies |  |
| With inductive loads | Max. 0.5 Hz |
| With lamp loads | Max. 11 Hz at max. 5 W |
| Short-circuit-proof / Overload-proof | No |
| Overload message | No |
| Output current limitation | No |
| Resistance to feedback against 24 V DC | No |
| Connection of 2 outputs in parallel | Not possible |
| Max. cable length |  |
| Shielded | 500 m |
| Unshielded | 150 m |

## Dimensions



The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1TNE 968 902 R2301 | DX561, digital input/output module, <br> 8 DI 24 V DC, 8 DO 24 V DC, <br> transistor output | Active |
| 1TNE 968 901 R3101 | Terminal block TA563-9, 9 pins, screw <br> front, cable side, 6 pieces per unit | Active |
| 1TNE 968 901 R3102 | Terminal block TA563-11, 11 pins, <br> screw front, cable side, 6 pieces per <br> unit | Active |
| 1TNE 968 901 R3103 | Terminal block TA564-9, 9 pins, screw <br> front, cable front, 6 pieces per unit | Active |
| 1TNE 968 901 R3104 | Terminal block TA564-11, 11 pins, <br> screw front, cable front, 6 pieces per <br> unit | Active |
| 1TNE 968 901 R3105 | Terminal block TA565-9, 9 pins, spring <br> front, cable front, 6 pieces per unit | Active |
| 1TNE 968 901 R3106 | Terminal block TA565-11, 11 pins, <br> spring front, cable front, 6 pieces per <br> unit | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

## DX571-Digital input/output module

## Features

- 8 digital inputs 24 V DC / 24 V AC (IO ... I7) in 1 group
- 8 digital normally open relay outputs 24 V DC / 24 V AC or 100 V AC ... 240 V AC, 2 A max. (NO0 ... NO7) in 2 groups
- Group-wise galvanically isolated


1 I/O bus
28 yellow LEDs to display the signal states of the inputs IO ... I7
38 yellow LEDs to display the signal states of the outputs NO0 ... NO7
4 Terminal number
5 Allocation of signal name
6 Terminal block for input signals (9-pin)

# 7 Terminal block for output signals (11-pin) <br> 82 holes for wall-mounting with screws <br> 9 DIN rail 

## Intended purpose

The device can be used as a decentralized I/O extension module for S 500 communication interface modules (e. g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs.

The inputs and outputs are group-wise galvanically isolated from each other.
All other circuitry of the module is galvanically isolated from the inputs.

The I/O module must not be used as a decentralized I/O module with CI590-CS31-HA communication interface modules.

## Functionality

| Parameter | Value |
| :--- | :--- |
| LED displays | For signal states |
| Internal power supply | Via I/O bus |
| External power supply | Via the terminal L+ (process voltage 24 V DC). The negative <br> pole is provided by the I/O bus. |

## Connections

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

The connection is carried out by using removable 9-pin and 11-pin terminal blocks. These terminal blocks differ in their connection system (spring terminals or screw terminals, cable mounting from the front or from the side). The terminal blocks are not included in the module's scope of delivery and must be ordered separately.


Fig. 68: Internal construction of the digital inputs and outputs

Table 135: Assignment of the terminals:

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 1 | CO ... 7 | Input common for signals IO ... <br> I7 |
| 2 | IO | Input signal IO |
| 3 | I1 | Input signal I1 |
| 4 | I2 | Input signal I2 |
| 5 | I3 | Input signal I3 |
| 6 | I4 | Input signal I4 |
| 7 | I5 | Input signal I5 |
| 8 | I6 | Input signal I6 |
| 9 | I7 | Input signal I7 |
| 10 | NOO | Normally-open contact of the <br> output 0 |
| 11 | NO1 | Normally-open contact of the <br> output 1 |


| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 12 | NO2 | Normally-open contact of the <br> output 2 |
| 13 | NO3 | Normally-open contact of the <br> output 3 |
| 14 | R0 ... 3 | Output common for signals <br> O0 ... O3 |
| 15 | NO4 | Normally-open contact of the <br> output 4 |
| 16 | NO5 | Normally-open contact of the <br> output 5 |
| 17 | NO7 | Normally-open contact of the <br> output 6 |
| 18 | R4 ... 7 | Normally-open contact of the <br> output 7 |
| 19 | L+ | Output common for signals <br> O4 $\ldots$. O7 |
| 20 | Process voltage +24 V DC |  |

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a communication interface module or a CPU). Thus, the current consumption from 24 V DC power supply at the terminals L+/UP and M/ZP of the CPU/communication interface module increases by 5 mA per DX571.
The external power supply connection is carried out via the $L+(+24 \mathrm{~V} C)$ terminal. The negative pole of the external power supply is realized via the I/O bus. Therefore, the CPU/ communication interface module and the DX571 must have a common power supply.

## WARNING!

## Removal/Insertion under power

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.
Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.
Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.
The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

Risk of damaging the PLC modules!
Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.


## NOTICE!

## Risk of damaging the PLC modules!

The PLC modules can be damaged by overload.
Make sure that the total current of each output common terminal (R0 ... 3 and R4 ... 7) does not exceed 8 A.

Never connect total currents > 8 A per group.
If the group fuse protection is not sufficient, then individual fuse protection of the outputs should be used.

The module provides several diagnosis functions (see Diagnosis $\Leftrightarrow$ Chapter 5.2.4.3.1.12.7 "Diagnosis" on page 511).
The digital inputs can be used as source inputs or as sink inputs.

## NOTICE!

Risk of malfunctions in the plant!
A ground fault, e. g. caused by a damaged cable insulation, can bridge switches accidentally.
Use sink inputs when possible or make sure that, in case of error, there will be no risks to persons or plant.


Fig. 69: Connection of inputs to the digital input/output module DX571-sink inputs


Fig. 70: Connection of inputs to the digital input/output module DX571-source inputs The following figures show the connection of the outputs to the module:


Fig. 71: Connection of 24 V DC actuators


Fig. 72: Connection of 24 V AC or 100 ... 240 V AC actuators

The L+ connection of the DX571 and the 24 V supply of the CPU/communication interface module must be connected to the same $24 V$ power supply.


Fig. 73: Power supply - the minus connection is realized via the I/O bus
1 CPU or communication interface module
2 I/O bus
3 DX571

## WARNING!

## Risk of death by electric shock!

Hazardous voltages can be present at the terminals of the module.
Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.

For screw-type terminals only:

## WARNING!

## For screw terminals only: Danger of death by electric shock!

The IP 20 protection degree is only provided if all terminal screws are tightened.
Tighten all screws of unused load terminals of relay outputs if voltages $>24 \mathrm{~V}$ are connected to the relay group.

NOTICE!

## Risk of damaging the I/O module!

The outputs are not protected against short circuit and overload.

- Never short-circuit or overload the outputs.
- Never connect inductive loads without an external suppression against voltage peaks due to inductive kickback.
- Never connect voltages > 240 V. All outputs must be supplied from the same phase.
- Use an external 5 A fast protection fuse for the outputs.

The meaning of the LEDs is described in the Displays section $\leadsto$ Chapter 5.2.4.3.1.12.8 "State LEDs" on page 512.

## I/O configuration

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or communication interface module) during power-up of the system.

Hence, replacing I/O modules is possible without any re-parameterization via software.


If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.

## Parameterization

The arrangement of the parameter data is performed with Automation Builder software.
The parameter data directly influences the functionality of modules.
For non-standard applications, it is necessary to adapt the parameters to your system configuration.

| Name | Value | Internal Value | Internal Value, Type | Default | Min. | Max. | EDS Slot <br> Index |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Module ID | Internal | $6140{ }^{1}$ ) | WORD | $\begin{array}{\|l\|} \hline 6140 \\ 0 \times 17 F C \end{array}$ | 0 | 65535 | xx01 |
| Ignore module | No Yes | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | BYTE | $\begin{array}{\|l\|} \hline \text { No } \\ (0 \times 00) \end{array}$ |  |  |  |
| Parameter length | Internal | 1 | BYTE | 0 | 0 | 255 | xx02 ${ }^{2}$ ) |
| Check supply | $\begin{aligned} & \text { Off } \\ & \text { On } \end{aligned}$ | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | BYTE | $\begin{aligned} & \hline \text { On } \\ & 0 \times 01 \end{aligned}$ |  |  |  |
| ${ }^{1}$ ) with CS31 and addresses smaller than 70, the value is increased by 1 |  |  |  |  |  |  |  |
| ${ }^{2}$ ) Value is hexadecimal: HighByte is slot (xx: $0 \ldots 7$ ), LowByte is index (1...n) |  |  |  |  |  |  |  |

GSD file:

| Ext_User_Prm_Data_Len $=$ | $0 \times 04$ |
| :--- | :--- |
| Ext_User_Prm_Data_Const $(0)=$ | $0 x F D, 0 \times 17,0 \times 00,1$ |
| $(0)=$ | $0 \times 01 ;$ |

Diagnosis

| E1 ... E4 | d1 | d2 | d3 | d4 | Identifier $000 \ldots 063$ | AC500- <br> Display | <- Display in |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err | $\begin{aligned} & \text { PS501 } \\ & \text { PLC } \\ & \text { Browser } \end{aligned}$ |  |
| $\begin{gathered} \text { Byte } 6 \\ \text { Bit } 6 \ldots 7 \end{gathered}$ | - | Byte 3 | Byte 4 | Byte 5 | $\begin{gathered} \text { Byte } 6 \\ \text { Bit } 0 \text {... } 5 \end{gathered}$ | PNIO diagnosis block |  |
| Class | Inter face | Device | Module | Channel | Error Identifier | Error message | Remedy |
|  | $\left.{ }^{1}\right)$ | ${ }^{2}$ ) | ${ }^{3}$ ) | $\left.{ }^{4}\right)$ |  |  |  |
| Module error |  |  |  |  |  |  |  |
| 3 | 14 | 1 ... 10 | 31 | 31 | 19 | Checksum error in the I/O module | Replace I/O module |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |
| 3 | 14 | 1 ... 10 | 31 | 31 | 43 | Internal error in the module | Replace I/O module |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |
| 3 | 14 | 1 ... 10 | 31 | 31 | 9 | Overflow diagnosis buffer | Restart |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |
| 4 | 14 | 1 ... 10 | 31 | 31 | 26 | Parameter error | Check master |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |
| 3 | 14 | 1 ... 10 | 31 | 31 | 11 | Process voltage too low | Check process voltage |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |

Remarks:

| $\left.{ }^{1}\right)$ | In AC500 the following interface identifier applies: <br> $14=$ I/O bus, 11 = COM1 (e.g. CS31 bus), 12 = COM2. <br> The PNIO diagnosis block does not contain this identifier. |
| :--- | :--- |
| ${ }^{2}$ ) | With "Device" the following allocation applies: <br> $31=$ Module itself, <br> $1 \ldots 10=$ decentralized communication interface module $1 \ldots 10$, <br> ADR = hardware address (e. g. of the DC551-CS31) |
| $\left.3^{3}\right)$ | With "Module" the following allocation applies depending on the master: <br> Module error: I/O bus or PNIO: 31 = Module itself; COM1/COM2: $1 \ldots 10=$ expan- <br> sion 1 $\ldots 10$ <br> Channel error: I/O bus or PNIO = Module type (2 = DO); COM1/COM2: $1 \ldots 10=$ <br> expansion 1 ... 10 |
| $\left.{ }^{4}\right)$ | In case of module errors, with channel "31 = module itself" is output. |

## State LEDs

| LED |  | State | Color | LED = OFF | LED = ON |
| :---: | :---: | :---: | :---: | :---: | :---: |
| AT13 DX571 | Inputs $10 . . .17$ | Digital input | Yellow | Input is OFF | Input is ON |
|  | $\begin{aligned} & \text { Outputs } \\ & \text { NO0 ... NO7 } \end{aligned}$ | Digital output | Yellow | Output is OFF | Output is ON |

In the undefined signal range, the state LED for the inputs can be ON although the input state detected by the module is OFF.

## Technical data

## Technical data of the module

The system data of AC500-eCo apply.
\& Chapter 5.1.1 "System data AC500-eCo" on page 159
Only additional details are therefore documented below.

| Parameter |  | Value |
| :--- | :--- | :--- |
| Process supply voltage L+ |  |  |
|  | Connections | Terminal 20 for L+ (+24 V DC). The neg- <br> ative pole is provided by the I/O bus. |
|  | Rated value | 24 V DC |
|  | Current consumption via L+ | 50 mA |
|  | Inrush current (at power-up) | $0.0035 \mathrm{~A}^{2} \mathrm{~s}$ |


| Parameter |  | Value |
| :--- | :--- | :--- |
|  | Max. ripple | $5 \%$ |
|  | Protection against reversed voltage | Yes |
| Rated protection fuse for L+ | Recommended; the outputs must be pro- <br> tected by a 3 A fast-acting fuse |  |
| Current consumption from 24 V DC power supply at <br> the L+/UP and M/ZP terminals of the CPU/commu- <br> nication interface module | Ca. 5 mA |  |
| Galvanic isolation | Yes, between the input group and the <br> output group and the rest of the module |  |
| Isolated groups | 3 groups (1 group for 8 input channels, 2 <br> groups for 8 output channels) |  |
| Surge-voltage (max.) | 35 V DC for 0.5 s |  |
| Max. power dissipation within the module | 2.3 W |  |
| Weight | Ca. 150 g |  |
| Mounting position | Horizontal or vertical |  |
| Cooling | The natural convection cooling must not <br> be hindered by cable ducts or other parts <br> in the control cabinet. |  |

No effects of multiple overloads

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an external fuse.

## Technical data of the digital inputs

| Parameter | Value |  |  |
| :---: | :---: | :---: | :---: |
| Number of channels per module | 8 |  |  |
| Distribution of the channels into groups | 1 group for 8 channels |  |  |
| Connections of the channels 10 ... I7 | Terminals $2 \ldots 9$ |  |  |
| Reference potential for the channels IO ... I7 | Terminal 1 |  |  |
| Indication of the input signals | 1 yellow LED per channel; the LED is ON when the input signal is high (signal 1) |  |  |
| Monitoring point of input indicator | LED is part of the input circuitry |  |  |
| Input type according to EN 61131-2 | Type 1 source | Type 1 sink | Type 1 AC ${ }^{1}$ ) |
| Input signal range | -24 V DC | +24 V DC | 24 V AC 50/60 Hz |
| Signal 0 | -5 V ... +3 V | -3 V ... +5V | 0 V AC ... 5 V AC |
| Undefined signal | -15V ... + 5 V | +5 V ... +15 V | 5 V AC ... 14 V AC |
| Signal 1 | -30 V ... -15 V | +15 V ... +30 V | $\begin{aligned} & 14 \vee \mathrm{AC} \ldots 27 \mathrm{~V} \\ & \mathrm{AC} \end{aligned}$ |
| Input current per channel |  |  |  |
| Input voltage 24 V | Typ. 5 mA |  | Typ. 5 mA r.m.s. |
| Input voltage 5 V | Typ. 1 mA |  | Typ. 1 mA r.m.s. |
| Input voltage 14 V |  |  | Typ. 2.7 mA r.m.s. |


| Parameter | Value |  |
| :---: | :---: | :---: |
| Input voltage 15 V | > 2.5 mA |  |
| Input voltage 27 V |  | Typ. 5.5 mA r.m.s. |
| Input voltage 30 V | $<8 \mathrm{~mA}$ |  |
| Max. permissible leakage current (at 2-wire proximity switches) | 1 mA | Typ. 1 mA r.m.s. |
| Input delay (0->1 or 1->0) | Typ. 8 ms |  |
| Input data length | 1 byte |  |
| Max. cable length |  |  |
| Shielded | 500 m |  |
| Unshielded | 300 m |  |

${ }^{1}$ ) When inputs are used with 24 VAC , external surge limiting filters are required.

## Technical data of the digital outputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 8 normally-open relay outputs |
| Distribution of the channels into groups | 2 (4 channels per group) |
| Connection of the channels $\mathrm{O} 0 \ldots \mathrm{O} 3$ | Terminals $10 \ldots 13$ |
| Connection of the channels $\mathrm{O} 4 \ldots \mathrm{O} 7$ | Terminals $15 . . .18$ |
| Reference potential for the channels O0 ... O3 | Terminal 14 (signal name R0 ... 3) |
| Reference potential for the channels O4 ... O7 | Terminal 19 (signal name R4 ... 7) |
| Relay coil power supply | Terminal 20 (positive pole of the process supply voltage, signal name $L+$ ). The negative pole is provided by the I/O bus. |
| Indication of the output signals | 1 yellow LED per channel; the LED is on when the output signal is high (signal 1) and the module is powered through the I/O bus |
| Monitoring point of output indicator | Controlled together with relay |
| Way of operation | Non-latching type |
| Relay output voltage |  |
| Rated value | 24 V DC / 24 V AC or 120/240 V AC |
| Output delay |  |
| Switching 0 to 1 (max.) | Typ. 10 ms |
| Switching 1 to 0 (max.) | Typ. 10 ms |
| Output data length | 1 byte |
| Output current |  |
| Rated current per channel (max.) | 2.0 A ( 24 V DC / 24 V AC / 48 V AC / $120 \mathrm{~V} \mathrm{AC} \mathrm{/}$ 240 V AC , only resistive loads) <br> 2.0 A (24 V AC / 48 V AC / 120 V AC , only pilot duty) <br> 1.5 A (240 V AC, only pilot duty) |
| Rated current per group (max.) | 8 A |


| Parameter | Value |
| :--- | :--- |
| Lamp load (max.) | $200 \mathrm{~W}(230 \mathrm{~V} \mathrm{AC}), 30 \mathrm{~W}$ (24 V DC) |
| Spark suppression with inductive AC <br> loads | Must be performed externally according to driven <br> load specification |
| Switching Frequencies |  |
| With resistive loads | Max. 1 Hz |
| With inductive loads | On Request |
| With lamp loads | Max. 1 Hz |
| Output type | Non-protected |
| Protection type | External fuse ${ }^{1}$ ) |
| Rated protection fuse | 5 A fast |
| Short-circuit-proof / Overload-proof | No, should be provided by an external fuse or circuit <br> breaker |
|  | Overload message |
|  | No |
| Connection of 2 outputs in parallel | Not possible |
| Lifetime of relay contacts (cycles) | 100.000 at rated load |
| Max. cable length |  |
|  | Shielded |
|  | Unshielded |

[^12]
## Dimensions



The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1TNE 968 902 R2302 | DX571, digital input/output module, <br> 8 DI 24 V DC / 24 V AC, 8 DO, relay <br> output | Active |
| 1TNE 968 901 R3101 | Terminal block TA563-9, 9 pins, screw <br> front, cable side, 6 pieces per unit | Active |
| 1TNE 968 901 R3102 | Terminal block TA563-11, 11 pins, <br> screw front, cable side, 6 pieces per <br> unit | Active |
| 1TNE 968 901 R3103 | Terminal block TA564-9, 9 pins, screw <br> front, cable front, 6 pieces per unit | Active |
| 1TNE 968 901 R3104 | Terminal block TA564-11, 11 pins, <br> screw front, cable front, 6 pieces per <br> unit | Active |
| 1TNE 968 901 R3105 | Terminal block TA565-9, 9 pins, spring <br> front, cable front, 6 pieces per unit | Active |
| 1TNE 968 901 R3106 | Terminal block TA565-11, 11 pins, <br> spring front, cable front, 6 pieces per <br> unit | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.4.3.2 S500 and S500-XC

DC522 - Digital input/output module

## Features

- 16 configurable digital inputs/outputs
- Module-wise galvanically isolated
- Fast counter
- XC version for use in extreme ambient conditions available


1 I/O bus
2 Allocation between terminal number and signal name
3 Sensor power supply 24 V DC / 0.5 A
416 yellow LEDs to display the signal states at the digital inputs/outputs (C0
51 green LED to display the state of the process supply voltage UP
64 red LEDs to display errors
7 Label

```
8 Terminal unit
9 DIN rail
```



## Intended purpose

The device can be used as a decentralized I/O extension module for S 500 communication interface modules (e. g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs.

Digital configurable input/output unit.

- 2 sensor supply voltages 24 V DC, 0.5 A , with short-circuit and overload protection
- 16 digital configurable inputs/outputs 24 V DC (C0 ... C15) in 1 group (2.0 ... 2.7 and 4.0 ... 4.7), each of which can be used
- as an input,
- as a transistor output with short-circuit and overload protection, 0.5 A rated current or
- as a re-readable output (combined input/output) with the technical data of the digital inputs and outputs.
- Optional with fast counter

The configuration is performed by software. The modules are supplied with a process supply voltage of 24 V DC.
All available inputs/outputs are galvanically isolated from all other circuitry of the module. There is no potential separation between the channels within the same group.
For use in extreme ambient conditions (e.g. wider temperature and humidity range), a special XC version of the device is available.

## Functionality

| Parameter | Value |
| :--- | :--- |
| Fast counter | Integrated, many configurable operating <br> modes (only with AC500) |
| LED displays | For signal states, errors and supply voltage |
| Internal power supply | Through the I/O bus interface (I/O bus) |
| External power supply | Via the terminals ZP and UP (process voltage <br> $24 \mathrm{~V} \mathrm{DC)}$ |
| Required terminal unit | TU515 or TU516 \& Chapter 5.2.5.2 "TU515, <br> TU516, TU541 and TU542 for I/O modules" <br> on page 938 |
| Effect of incorrect input terminal connection | Wrong or no signal detected, no damage up to <br> 35 V |

The device is plugged on a terminal unit ${ }^{\star}$ Chapter 5.2.5.2 "TU515, TU516, TU541 and TU542 for I/O modules" on page 938. Position the module properly and press until it locks in place. The terminal unit is either mounted on a DIN rail or to the wall using 2 screws plus the additional accessory for wall mounting ${ }^{\star} \Rightarrow$ Chapter 5.2.8.2.5 "TA526-Wall mounting accessory" on page 1324.

## Connections

The connection of the I/O channels is carried out using the 40 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

The terminals $1.8 \ldots 4.8$ and $1.9 \ldots 4.9$ are electrically interconnected within the I/O terminal unit and always have the same assignment, irrespective of the inserted module:

Terminals 1.8 ... 4.8: process voltage UP $=+24 \mathrm{~V}$ DC
Terminals $1.9 \ldots$ 4.9: process voltage $\mathrm{ZP}=0 \mathrm{~V}$ DC


1 I/O bus
2 4.0 ... 4.7: Connected with UP (switch) -> Input; Connected with ZP (load) -> Output
3 Control cabinet earth
The assignment of the other terminals:

| Terminals | Signal | Description |
| :--- | :--- | :--- |
| $1.0 \ldots 1.3$ | +24 V | $4 \times$ sensor power supply sources (loadable with 0.5 <br> A in total) |
| $1.4 \ldots 1.7$ | 0 V | 0 V (reference potential) |
| $2.0 \ldots 2.7$ | $\mathrm{C} 0 \ldots \mathrm{C} 7$ | 8 digital inputs/outputs |
| $3.0 \ldots 3.3$ | +24 V | $4 \times$ x sensor power supply sources (loadable with 0.5 <br> A in total) |
| $3.4 \ldots 3.7$ | 0 V | 0 V (reference potential) |
| $4.0 \ldots 4.7$ | $\mathrm{C} 8 \ldots \mathrm{C} 15$ | 8 digital inputs/outputs |

## CAUTION!

The process supply voltage must be included in the grounding concept (e. g. grounding of the negative terminal).

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a communication interface module or a CPU). Thus, the current consumption from 24 V DC power supply at the terminals L+/UP and M/ZP of the CPU/communication interface module increases by 2 mA per DC522.

The external power supply connection is carried out via the UP (+24 V DC) and the ZP (0 V DC) terminals.

## WARNING!

## Removal/Insertion under power

Removal or insertion under power is permissible only if all conditions for hot swapping are fullfilled.
(4) Chapter 4.9.3 "Replace an I/O module with hot swap" on page 153

The devices are not designed for removal or insertion under power when the conditions for hot swap do not apply. Because of unforeseeable consequences, it is not allowed to plug in or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while they are energized in a hazardous location could result in an electric arc, which could create an ignition source resulting in fire or explosion.

Prior to proceeding, make sure that power is been disconnected and that the area has been thoroughly checked to ensure that flammable materials are not present.

The devices must not be opened when in operation. The same applies to the network interfaces.


## NOTICE!

## Risk of influences to the connected sensors!

Some sensors may be influenced by the deactivated module outputs of DC522.
Connect a $470 \Omega / 1 \mathrm{~W}$ resistor in series to inputs C8/C9 if they are used as fast counter inputs to avoid any influences.

The modules provide several diagnosis functions.

## Internal data exchange

|  | Without the fast counter | With the fast counter (only <br> with AC500) |
| :--- | :--- | :--- |
| Digital inputs (bytes) | 2 | 4 |
| Digital outputs (bytes) | 2 | 4 |
| Counter input data (words) | 0 | 4 |
| Counter output data (words) | 0 | 8 |

## I/O Configuration

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or communication interface module) during power-up of the system.

Hence, replacing I/O modules is possible without any re-parameterization via software.


## Parameterization

| Firmware version | Configuration |
| :--- | :--- |
| Firmware version $>\mathrm{V} 2.0 .0$ | The arrangement of the parameter data is per- <br> formed by Control Builder Plus/ Automation <br> Builder software. |

The parameter data directly influences the functionality of modules.
For non-standard applications, it is necessary to adapt the parameters to your system configuration.

Module: Module slot address: $\mathrm{Y}=1 \ldots 10$

| Name | Value | Internal value | Internal value, type | Default | Min. | Max. | EDS Slot/ <br> Index |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Module ID | Internal | $\begin{aligned} & 1220 \\ & 1 \text { 1) } \end{aligned}$ | Word | $\begin{aligned} & \hline 1220 \\ & 0 \times 04 \mathrm{C} 4 \end{aligned}$ | 0 | 65535 | 0x0Y01 |
| Ignore module ${ }^{2}$ ) | $\begin{aligned} & \text { No } \\ & \text { Yes } \end{aligned}$ | $0$ | Byte | $\begin{aligned} & \text { No } \\ & 0 \times 00 \end{aligned}$ |  |  | Not for FBP |
| Parameter length | Internal | 7 | Byte | $\begin{aligned} & \hline 7-\mathrm{CPU} \\ & \text { 6-FBP } \end{aligned}$ | 0 | 255 | 0x0Y02 |
| Check supply | $\begin{aligned} & \text { Off } \\ & \text { On } \end{aligned}$ | $0$ | Byte | $\begin{aligned} & \mathrm{On} \\ & 0 \times 01 \end{aligned}$ | 0 | 1 | 0x0Y03 |
| Input delay | 0.1 ms 1 ms 8 ms 32 ms | $\begin{aligned} & 0 \\ & 1 \\ & 2 \\ & 3 \end{aligned}$ | Byte | $\begin{aligned} & 8 \mathrm{~ms} \\ & 0 \times 02 \end{aligned}$ | 0 | 3 | 0x0Y04 |
| Fast counter ${ }^{4}$ ) | $\begin{aligned} & 0 \\ & : \\ & \left.10^{3}\right) \end{aligned}$ | $\begin{aligned} & 0 \\ & : \\ & 10 \end{aligned}$ | Byte | $\begin{aligned} & \text { Mode } 0 \\ & 0 \times 00 \end{aligned}$ |  |  | Not for FBP |
| Short-circuit detection of output or sensor supply | $\begin{aligned} & \text { Off } \\ & \text { On } \end{aligned}$ | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | Byte | $\begin{aligned} & \mathrm{On} \\ & 0 \times 01 \end{aligned}$ | 0 | 1 | 0x0Y05 |
| Behaviour of outputs at com-munication errors | Off <br> Last value <br> Substitute value | $\begin{array}{\|l} \hline 0 \\ 1+(n * 5) \\ 2+(n * 5) \\ n \leq 2 \end{array}$ | Byte | $\begin{aligned} & \text { Off } \\ & 0 \times 00 \end{aligned}$ | 0 | 2 | 0x0Y06 |
| Substitute value at outputs <br> Bit $15=$ Output 15 <br> Bit $0=$ Output 0 | $\begin{aligned} & 0 \ldots \\ & 65535 \end{aligned}$ | $0 \text {... }$ <br> 0xffff | Word | 0 | 0 | 65535 | 0x0Y07 |

Remarks:

| $\left.{ }^{1}\right)$ | With CS31 and addresses smaller than 70 and FBP, the value is increased <br> by 1 |
| :--- | :--- |
| $\left.{ }^{2}\right)$ | Not with FBP |
| $\left.{ }^{3}\right)$ | For a description of the counter operating modes, please refer to the 'Fast <br> Counter' section <br> $y$ |
| $\left.{ }^{4}\right)$ | Whapter 5.2.4.3.2.9 "Fast counter" on page 606 |, | WBP or CS31 without the parameter Fast Counter |
| :--- |

GSD file:

| Ext_User_Prm_Data_Len $=$ | 9 |
| :--- | :--- |
| Ext_User_Prm_Data_Const $(0)=$ | $0 \times 04,0 \times c 5,0 \times 06,1$ |
|  | $0 \times 01,0 \times 02,0 \times 01,0 \times 00,0 \times 00,0 \times 00 ;$ |

## State LEDs

During the power ON procedure, the module initializes automatically. All LEDs (except the channel LEDs) are ON during this time.

| LED |  | State | Color | LED = OFF | LED = ON | LED flashes |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | Inputs/ outputs C0 ... C15 | Digital input or digital output | Yellow | Input/output $=\text { OFF }$ | Input/output = ON ${ }^{1}$ ) | -- |
|  | UP | Process supply voltage 24 V DC via terminal | Green | Process supply voltage is missing | Process supply voltage OK | -- |
|  | CH-ERR1 | Channel Error, error messages in groups (digital inputs/ outputs combined into the groups 1, 2, 3, 4) | Red | No error or process supply voltage is missing | Severe error within the corresponding group | Error on one channel of the corresponding group (e.g. short circuit at an output) |
| $\underbrace{197 P}_{\text {CH-ERRI }}$ | CH-ERR2 |  | Red |  |  |  |
|  | CH-ERR3 |  | Red |  |  |  |
|  | CH-ERR4 |  | Red |  |  |  |
|  | $\mathrm{CH}-E R \mathrm{R}^{2}$ ) | Module error | Red | -- | Internal error | -- |
|  | ${ }^{1}$ ) Indication LED is ON even if an input signal is applied to the channel and the supply voltage is off. In this case the module is not operating and does not generate an input signal. |  |  |  |  |  |
|  | ${ }^{2}$ ) All of the LEDs CH-ERR1 to CH-ERR4 light up together |  |  |  |  |  |

## Technical data

## Technical data of the module

The system data of AC500 and S500 are applicable to the standard version ${ }^{\sharp}$ Chapter 5.1.2 "System data AC500" on page 166.

The system data of AC500-XC are applicable to the XC version $\stackrel{y}{ }{ }^{〔}$ Chapter 5.1.3 "System data AC500-XC" on page 169.
Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter | Value |
| :---: | :---: |
| Process supply voltage UP |  |
| Connections | Terminals 1.8, 2.8, 3.8 and 4.8 for +24 V (UP) as well as 1.9, 2.9, 3.9 and 4.9 for $0 \mathrm{~V}(\mathrm{ZP})$ |
| Rated value | 24 V DC |
| Max. ripple | 5 \% |
| Protection against reversed voltage | Yes |
| Rated protection fuse on UP | 10 A fast |
| Galvanic isolation | Yes, per module |
| Current consumption |  |
| From 24 V DC power supply at the L+/UP and M/ZP terminals of the CPU/communication interface module | Ca. 2 mA |
| From UP at normal operation / with outputs | 0.15 A + max. 0.5 A per output |
| Inrush current from UP (at power up) | $0.005 \mathrm{~A}^{2} \mathrm{~s}$ |
| Max. power dissipation within the module | 6 W (outputs unloaded) |
| Sensor power supply |  |
| Connections | Terminals $1.0 \ldots 1.3=+24 \mathrm{~V}, 1.4 \ldots 1.7=0 \mathrm{~V}$ <br> Terminals 3.0 ... $3.3=+24 \mathrm{~V}, 3.4 \ldots 3.7=0 \mathrm{~V}$ |
| Voltage | 24 V DC with short-circuit and overload protection |
| Loadability | Terminals 1.0 ... 1.3, in total max. 0.5 A Terminals 3.0 ... 3.3, in total max. 0.5 A |
| Weight (without terminal unit) | Ca. 125 g |
| Mounting position | Horizontal <br> Or vertical with derating (output load reduced to $50 \%$ at $+40{ }^{\circ} \mathrm{C}$ per group) |
| Cooling | The natural convection cooling must not be hindered by cable ducts or other parts in the control cabinet. |

## Multiple overloads

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an internal smart high-side switch.

## 0 <br> NOTICE! <br> All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and temporary overvoltage up to 30 V DC.

## Technical data of the configurable digital inputs/outputs

Each of the configurable I/O channels is defined as input or output by the user program. This is done by interrogating or allocating the corresponding channel.

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 16 inputs/outputs (with transistors) |
| Distribution of the channels into groups | 1 group of 16 channels |
| If the channels are used as inputs |  |
|  | Channels C0 ... C7 |
|  | Channels C8 ... C15 |
| If the channels are used as outputs | Terminals $4.0 \ldots 4.7$ |
|  | Channels C0 ... C7 |
|  | Channels C8 ... C15 |
| Indication of the input/output signals | Terminals $2.0 \ldots 2.7$ |
| Monitoring point of input/output indicator | Terminals $4.0 \ldots 4.7$ |
| Galvanic isolation | 1 yellow LED per channel, the LED is ON <br> when the input/output signal is high (signal 1) |

## Technical data of the digital inputs/outputs if used as inputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | Max. 16 digital inputs |
| Reference potential for all inputs | Terminals $1.9,2.9,3.9$ and 4.9 (negative pole <br> of the process supply voltage, signal name <br> ZP) |
| Galvanic isolation | From the rest of the module |
| Indication of the input signals | 1 yellow LED per channel, the LED is ON <br> when the input signal is high (signal 1) |
| Monitoring point of input/output indicator | LED is part of the input circuitry |
| Input type acc. to EN $61131-2$ | Type 1 |
| Input delay (0->1 or 1->0) | Typ. 8 ms, configurable from $0.1 \ldots 32 \mathrm{~ms}$ |
| Input signal voltage | 24 V DC |
|  | Signal 0 |
| Undefined signal | $-3 \mathrm{~V} \ldots+5 \mathrm{~V}$ *) |
|  | $>+5 \mathrm{~V} \ldots<+15 \mathrm{~V}$ |
| Sipple with signal 0 | $+15 \mathrm{~V} \ldots+30 \mathrm{~V}$ |
| Ripple with signal 1 | Within $-3 \mathrm{~V} \ldots+5 \mathrm{~V}$ *) |
| Input current per channel | Within $+15 \mathrm{~V} \ldots+30 \mathrm{~V}$ |
|  | Input voltage +24 V |
| Input voltage +5 V | Typ. 5 mA |
| Input voltage +15 V | $>1 \mathrm{~mA}$ |
| Input voltage +30 V | $>5 \mathrm{~mA}$ |
| Max. cable length | $<8 \mathrm{~mA}$ |
|  |  |


| Parameter |  | Value |
| :--- | :--- | :--- |
|  | Shielded | 1000 m |
|  | Unshielded | 600 m |

*) Due to the direct connection to the output, the demagnetizing varistor is also effective at the input (see figure) above. This is why the difference between UPx and the input signal may not exceed the clamp voltage of the varistor. The varistor limits the voltage to approx. 36 V . Consequently, the input voltage must range $-12 \mathrm{~V} \ldots+30 \mathrm{~V}$ when $\mathrm{UPx}=24 \mathrm{~V}$ and $-6 \mathrm{~V} \ldots+30 \mathrm{~V}$ when UPx $=30 \mathrm{~V}$.

## Technical data of the digital inputs/outputs if used as outputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | Max. 16 transistor outputs |
| Reference potential for all outputs | Terminals 1.9, 2.9, 3.9 and 4.9 (negative pole of the process supply voltage, signal name ZP) |
| Common power supply voltage | For all outputs: terminals $1.8,2.8,3.8$ and 4.8 (positive pole of the process supply voltage, signal name UP) |
| Output voltage for signal 1 | UP (-0.8 V) |
| Output delay (0->1 or 1->0) | On request |
| Output current |  |
| Rated value, per channel | 500 mA at $\mathrm{UP}=24 \mathrm{~V}$ |
| Maximum value (all channels together) | 8 A |
| Leakage current with signal 0 | $<0.5 \mathrm{~mA}$ |
| Rated protection fuse on UP | 10 A fast |
| Demagnetization when inductive loads are switched off | With varistors integrated in the module (see figure below) |
| Switching frequency |  |
| With resistive load | On request |
| With inductive loads | Max. 0.5 Hz |
| With lamp loads | Max. 11 Hz with max. 5 W |
| Short-circuit-proof / overload-proof | Yes |
| Overload message ( $1>0.7$ A) | Yes, after ca. 100 ms |
| Output current limitation | Yes, automatic reactivation after short circuit/overload |
| Resistance to feedback against 24 V signals | Yes |
| Max. cable length |  |
| Shielded | 1000 m |
| Unshielded | 600 m |

The following drawing shows the circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off.


Fig. 74: Digital input/output (circuit diagram)
1 UPx (+ 24 V )
2 Digital input/output
3 ZPx (0 V)
4 For demagnization when inductive loads are switched off

## Technical data of the fast counter

The fast counter of the module does not work if the module is connected to a

- FBP interface module
- CS31 bus module
- CANopen communication interface module

| Parameter | Value |
| :--- | :--- |
| Used inputs | $\mathrm{C} 8 / \mathrm{C} 9$ |
| Used outputs | C 10 |
| Counting frequency | Max. 50 kHz |

(3) Chapter 6.8.2.12 "Fast counters in AC500 devices" on page 4536

## Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm

The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 240 600 R0001 | DC522, digital input/output module, <br> 16 DC, 24 V DC / 0.5 A, 2-wires | Active |
| 1SAP 440 600 R0001 | DC522-XC, digital input/output <br> module, 16 DC, 24 V DC / 0.5 A, <br> 2-wires, XC version | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

## DC523 - Digital input/output module

## Features

- 24 configurable digital inputs/outputs
- Module-wise galvanically isolated
- Fast counter
- XC version for use in extreme ambient conditions available


1 I/O bus
2 Allocation between terminal number and signal name
3 Sensor power supply 24 V DC / 0.5 A
424 yellow LEDs to display the signal states at the digital inputs/outputs (C0 ... C23)
51 green LED to display the status of the process supply voltage UP
64 red LEDs to display errors
7 Label
8 Terminal unit
9 DIN rail
Sign for XC version

## Intended purpose

The device can be used as a decentralized I/O extension module for S 500 communication interface modules (e. g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs.

Digital configurable input/output unit.

- 1 sensor supply voltage 24 V DC, 0.5 A , with short circuit and overload protection
- 24 digital configurable inputs/outputs 24 V DC (C0 ... C23) in 1 group (2.0 ... 2.7, 3.0 ... 3.7 and $4.0 \ldots 4.7$ ), of which each can be used
- as an input,
- as a transistor output with short circuit and overload protection, 0.5 A rated current or
- as a re-readable output (combined input/output) with the technical data of the digital inputs and outputs.
- Optional with fast counter

The configuration is performed by software. The modules are supplied with a process supply voltage of 24 V DC.

All available inputs/outputs are galvanically isolated from all other circuitry of the module. There is no potential separation between the channels within the same group.

For use in extreme ambient conditions (e.g. wider temperature and humidity range), a special $X C$ version of the device is available.

## Functionality

| Parameter | Value |
| :--- | :--- |
| Fast counter | Integrated, many configurable operating <br> modes (only with AC500) |
| LED displays | For signal states, errors and supply voltage |
| Internal power supply | Through the I/O bus interface (I/O bus) |
| External power supply | Via the terminals ZP and UP (process voltage <br> 24 V DC) |
| Required terminal unit | TU515 or TU516 « Chapter 5.2.5.2 "TU515, <br> TU516, TU541 and TU542 for I/O modules" <br> on page 938 |
| Effect of incorrect input terminal connection | Wrong or no signal detected, no damage up to <br> 35 V |

## NOTICE!

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and temporary overvoltage up to 30 V DC.

The device is plugged on a terminal unit ${ }^{\wedge}$ Chapter 5.2.5.2 "TU515, TU516, TU541 and TU542 for I/O modules" on page 938. Position the module properly and press until it locks in place. The terminal unit is either mounted on a DIN rail or to the wall using 2 screws plus the additional accessory for wall mounting ${ }^{\mu}$, Chapter 5.2.8.2.5 "TA526-Wall mounting accessory" on page 1324.

## Connections

The connection of the I/O channels is carried out using the 40 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

The terminals $1.8 \ldots 4.8$ and $1.9 \ldots 4.9$ are electrically interconnected within the I/O terminal unit and always have the same assignment, irrespective of the inserted module:

Terminals $1.8 \ldots 4.8$ : process voltage UP $=+24 \mathrm{~V}$ DC

Terminals $1.9 \ldots 4.9$ : process voltage $\mathrm{ZP}=0 \mathrm{~V}$ DC


1 I/O bus
2 4.0 ... 4.7: Connected with UP (switch) -> Input;
Connected with ZP (load) -> Output
3 Control cabinet earth
The assignment of the other terminals:

| Terminals | Signal | Description |
| :--- | :--- | :--- |
| $1.0 \ldots 1.3$ | +24 V | $4 \times$ sensor power supply sources (loadable <br> with 0.5 A in total) |
| $1.4 \ldots 1.7$ | 0 V | 0 V (reference potential) |
| $2.0 \ldots 2.7$ | $\mathrm{C} 0 \ldots \mathrm{C} 7$ | 8 digital inputs/outputs |
| $3.0 \ldots 3.7$ | $\mathrm{C} 8 \ldots \mathrm{C} 15$ | 8 digital inputs/outputs |
| $4.0 \ldots 4.7$ | $\mathrm{C} 16 \ldots \mathrm{C} 23$ | 8 digital inputs/outputs |

## CAUTION!

The process supply voltage must be included in the grounding concept (e. g. grounding of the negative terminal).

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a communication interface module or a CPU). Thus, the current consumption from 24 V DC power supply at the terminals L+/UP and M/ZP of the CPU/communication interface module increases by 2 mA per DC523.
The external power supply connection is carried out via the UP (+24 V DC) and the ZP (0 V DC) terminals.

## WARNING!

## Removal/Insertion under power

Removal or insertion under power is permissible only if all conditions for hot swapping are fullfilled.

Chapter 4.9.3 "Replace an I/O module with hot swap" on page 153
The devices are not designed for removal or insertion under power when the conditions for hot swap do not apply. Because of unforeseeable consequences, it is not allowed to plug in or unplug devices with the power being ON.
Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while they are energized in a hazardous location could result in an electric arc, which could create an ignition source resulting in fire or explosion.
Prior to proceeding, make sure that power is been disconnected and that the area has been thoroughly checked to ensure that flammable materials are not present.

The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

## Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.


## NOTICE!

## Risk of influences to the connected sensors!

Some sensors may be influenced by the deactivated module outputs of DC523.
Connect a $470 \Omega / 1 \mathrm{~W}$ resistor in series to inputs C16/C17 if they are used as fast counter inputs to avoid any influences.

The modules provide several diagnosis functions.

Internal data exchange

|  | Without the fast counter | With the fast counter (only <br> with AC500) |
| :--- | :--- | :--- |
| Digital inputs (bytes) | 3 | 5 |
| Digital outputs (bytes) | 3 | 5 |
| Counter input data (words) | 0 | 4 |
| Counter output data (words) | 0 | 8 |

## I/O configuration

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or communication interface module) during power-up of the system.

Hence, replacing I/O modules is possible without any re-parameterization via software.
If the external power supply voltage via UP/ZP terminals fails, the I/O module
loses its configuration data. The whole station has to be switched off and on
again to re-configure the module.

## Parameterization

| Firmware version | Configuration |
| :--- | :--- |
| Firmware version > V2.0.0 | The arrangement of the parameter data is per- <br> formed by Control Builder Plus/ Automation <br> Builder software. |

The parameter data directly influences the functionality of modules.
For non-standard applications, it is necessary to adapt the parameters to your system configuration.

Module: Module slot address: $\mathrm{Y}=1 \ldots 10$

| Name | Value | Internal value | Internal value, type | Default | Min. | Max. | EDS Slot/ <br> Index |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Module ID | Internal | $\begin{aligned} & \hline 1215 \\ & \left.{ }^{1}\right) \end{aligned}$ | Word | $\begin{aligned} & 1215 \\ & 0 \times 04 \mathrm{BF} \end{aligned}$ | 0 | 65535 | 0x0Y01 |
| Ignore module ${ }^{2}$ ) | No Yes | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | Byte | $\begin{aligned} & \mathrm{No} \\ & 0 \times 00 \end{aligned}$ |  |  | Not for FBP |
| Parameter length | Internal | 9 | Byte | $\begin{aligned} & 9-\mathrm{CPU} \\ & 8-\mathrm{FBP} \end{aligned}$ | 0 | 255 | 0x0Y02 |
| Check supply | Off on | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | Byte | $\begin{array}{\|l\|} \hline \text { On } \\ 0 \times 01 \end{array}$ | 0 | 1 | $0 x=Y 03$ |


| Name | Value | Internal value | Internal value, type | Default | Min. | Max. | EDS Slot Index |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Input delay | 0.1 ms 1 ms 8 ms 32 ms | $\begin{array}{\|l\|} \hline 0 \\ 1 \\ 2 \\ 3 \end{array}$ | Byte | $\begin{aligned} & 8 \mathrm{~ms} \\ & 0 \times 02 \end{aligned}$ | 0 | 3 | 0x0Y04 |
| Fast counter ${ }^{4}$ ) | 0 <br> 10 <br> ${ }^{3}$ ) | 0 $10$ | Byte | Mode 0 $0 \times 00$ |  |  | Not for FBP |
| Short circuit detection of output or sensor supply | $\begin{aligned} & \text { Off } \\ & \text { On } \end{aligned}$ | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | Byte | $\begin{array}{\|l\|} \hline \text { On } \\ 0 \times 01 \end{array}$ | 0 | 1 | $0 \times 0 Y 05$ |
| Behaviour of outputs at com-munication errors | Off <br> Last value <br> Substitute value | $\begin{aligned} & \hline 0 \\ & 1+\left(n^{*} 5\right) \\ & 2+\left(n^{*} 5\right) \\ & n \leq 2 \end{aligned}$ | Byte | $\begin{array}{\|l\|} \hline \text { Off } \\ 0 \times 00 \end{array}$ | 0 | 2 | 0x0Y06 |
| Substitute value at outputs <br> B23 = <br> Output 23 <br> Bit $0=$ <br> Output 0 | $\begin{aligned} & 0 \ldots \\ & 16777215 \end{aligned}$ | 0 ... <br> 0x00ff-ffff | DWord | $\begin{array}{\|l\|} \hline 0 \\ 0 \times 0000 \\ -0000 \end{array}$ | 0 | 224-1 | 0x0Y07 |

Remarks:

| $\left.{ }^{1}\right)$ | With CS31 and addresses smaller than 70 and FBP, the value is increased <br> by 1 |
| :--- | :--- |
| ${ }^{2}$ ) | Not with FBP |
| ${ }^{3}$ ) | For a description of the counter operating modes, please refer to the 'Fast <br> Counter' section $\# ~ C h a p t e r ~ 5.2 .4 .3 .2 .9 ~ " F a s t ~ c o u n t e r " ~ o n ~ p a g e ~ 606 ~$ |$|$| ${ }^{4}$ ) | With FBP or CS31 without the parameter Fast Counter |
| :--- | :--- |

GSD file:

| Ext_User_Prm_Data_Len $=$ | 11 |
| :--- | :--- |
| Ext_User_Prm_Data_Const $(0)=$ | $0 \times 04,0 \times c 0,0 \times 08,1$ |
|  | $0 \times 01,0 \times 02,0 \times 01,0 \times 00,0 \times 00,0 \times 00,0 \times 00$, |
|  | $0 \times 00 ;$ |

## State LEDs

During the power ON procedure, the module initializes automatically. All LEDs (except the channel LEDs) are ON during this time.


## Technical data

## Technical data of the module

The system data of AC500 and S500 are applicable to the standard version ${ }^{\Perp}$ Chapter 5.1.2 "System data AC500" on page 166.
The system data of AC500-XC are applicable to the XC version ${ }^{\mu}$ Chapter 5.1.3 "System data AC500-XC" on page 169.
Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter |  | Value |
| :--- | :--- | :--- |
| Process supply voltage UP |  |  |
|  | Connections | Terminals 1.8, 2.8, 3.8 and 4.8 for +24 V (UP) as <br> well as 1.9, 2.9, 3.9 and 4.9 for 0 V (ZP) |
|  | Rated value | 24 V DC |
|  | Max. ripple | $5 \%$ |
|  | Protection against reversed voltage | Yes |
|  | Rated protection fuse on UP | 10 A fast |
|  | Galvanic isolation | Yes, per module |
| Current consumption |  |  |


| Parameter | Value |
| :---: | :---: |
| From 24 V DC power supply at the terminals UP/L+ and ZP/M of the CPU/ communication interface module | Ca. 2 mA |
| From UP at normal operation / with outputs | 0.1 A + max. 0.5 A per output |
| Inrush current from UP (at power up) | $0.008 \mathrm{~A}^{2} \mathrm{~s}$ |
| Max. power dissipation within the module | 6 W (outputs unloaded) |
| Sensor power supply |  |
| Connections | Terminals 1.0 ... $1.3=+24 \mathrm{~V}, 1.4 \ldots 1.7=0 \mathrm{~V}$ |
| Voltage | 24 V DC with short circuit and overload protection |
| Loadability | Terminals $1.0 \ldots 1.3$, in total max. 0.5 A |
| Weight (without terminal unit) | Ca. 125 g |
| Mounting position | Horizontal <br> Or vertical with derating (output load reduced to $50 \%$ at $+40{ }^{\circ} \mathrm{C}$ per group) |
| Cooling | The natural convection cooling must not be hindered by cable ducts or other parts in the control cabinet. |

## NOTICE!

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and temporary overvoltage up to 30 V DC.

## Multiple overloads

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an internal smart high-side switch.

## Technical data of the configurable digital inputs/outputs

Each of the configurable I/O channels is defined as input or output by the user program. This is done by interrogating or allocating the corresponding channel.

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 24 inputs/outputs (with transistors) |
| Distribution of the channels into groups | 1 group of 24 channels |
| If the channels are used as inputs |  |
| Channels $\mathrm{C} 0 . . . \mathrm{C} 7$ | Terminals 2.0 ... 2.7 |
| Channels C8 ... C15 | Terminals 3.0 ... 3.7 |
| Channels C16 ... C23 | Terminals 4.0 ... 4.7 |
| If the channels are used as outputs |  |
| Channels C0 ... C7 | Terminals 2.0 ... 2.7 |


| Parameter |  | Value |
| :--- | :--- | :--- |
|  | Channels C8 ... C15 | Terminals 3.0 ... 3.7 |
|  | Channels C16 ... C23 | Terminals 4.0 ... 4.7 |
| Indication of the input/output signals | 1 yellow LED per channel, the LED is ON when <br> the input/output signal is high (signal 1) |  |
| Monitoring point of input/output indicator | LED is part of the input circuitry |  |
| Galvanic isolation | From the rest of the module |  |

## Technical data of the digital inputs/outputs if used as inputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | Max. 24 digital inputs |
| Reference potential for all inputs | Terminals 1.9, 2.9, 3.9 and 4.9 (negative pole of the process supply voltage, signal name ZP) |
| Galvanic isolation | From the rest of the module |
| Indication of the input signals | 1 yellow LED per channel, the LED is ON when the input signal is high (signal 1) |
| Monitoring point of input/output indicator | LED is part of the input circuitry |
| Input type acc. to EN 61131-2 | Type 1 |
| Input delay (0->1 or 1->0) | Typ. 8 ms , configurable from 0.1 to 32 ms |
| Input signal voltage | 24 V DC |
| Signal 0 | $-3 \vee \ldots+5 \vee *)$ |
| Undefined signal | > +5V ... $<+15 \mathrm{~V}$ |
| Signal 1 | +15 V ... +30 V |
| Ripple with signal 0 | Within -3 V ... +5V *) |
| Ripple with signal 1 | Within +15 V ... +30 V |
| Input current per channel |  |
| Input voltage +24 V | Typ. 5 mA |
| Input voltage +5 V | $>1 \mathrm{~mA}$ |
| Input voltage +15 V | $>5 \mathrm{~mA}$ |
| Input voltage +30 V | $<8 \mathrm{~mA}$ |
| Max. cable length |  |
| Shielded | 1000 m |
| Unshielded | 600 m |

*) Due to the direct connection to the output, the demagnetizing varistor is also effective at the input (see figure) above. This is why the difference between UPx and the input signal must not exceed the clamp voltage of the varistor. The varistor limits the clamp voltage to approx. 36 V . Consequently, the input voltage must range from $-12 \mathrm{~V} \ldots+30 \mathrm{~V}$ when $\mathrm{UPx}=24 \mathrm{~V}$ and from $-6 \mathrm{~V} . . .+30 \mathrm{~V}$ when $\mathrm{UPx}=30 \mathrm{~V}$.

## Technical data of the digital inputs/outputs if used as outputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | Max. 24 transistor outputs |
| Reference potential for all outputs | Terminals 1.9, 2.9, 3.9 and 4.9 (negative pole of the process supply voltage, signal name ZP) |
| Common power supply voltage | For all outputs: terminals 1.8, 2.8, 3.8 and 4.8 (positive pole of the process supply voltage, signal name UP) |
| Output voltage for signal 1 | UP (-0.8 V) |
| Output delay (0->1 or 1->0) | On request |
| Output current |  |
| Rated value, per channel | 500 mA at $\mathrm{UP}=24 \mathrm{~V}$ |
| Maximum value (all channels together) | 8 A |
| Leakage current with signal 0 | $<0.5 \mathrm{~mA}$ |
| Rated protection fuse on UP | 10 A fast |
| Demagnetization when inductive loads are switched off | With varistors integrated in the module (see figure below) |
| Switching frequency |  |
| With resistive load | On request |
| With inductive loads | Max. 0.5 Hz |
| With lamp loads | Max. 11 Hz with max. 5 W |
| Short-circuit-proof / overload-proof | Yes |
| Overload message ( $1>0.7$ A) | Yes, after ca. 100 ms |
| Output current limitation | Yes, automatic reactivation after short circuit/overload |
| Resistance to feedback against 24 V signals | Yes |
| Max. cable length |  |
| Shielded | 1000 m |
| Unshielded | 600 m |

The following drawing shows the circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off.


Fig. 75: Digital input/output (circuit diagram)
1 UPx (+ 24 V )
2 Digital input/output
3 ZPx (0 V)
4 For demagnization when inductive loads are switched off

## Technical data of the fast counter

The fast counter of the module does not work if the module is connected to a

- FBP interface module
- CS31 bus module
- CANopen communication interface module

| Parameter | Value |
| :--- | :--- |
| Used inputs | $\mathrm{C} 16 / \mathrm{C} 17$ |
| Used outputs | C 18 |
| Counting frequency | Max. 50 kHz |

② Chapter 6.8.2.12 "Fast counters in AC500 devices" on page 4536

How to prepare a device as fast counter and how to connect it to the PLC is described in an application example.

## Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm

The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 240 500 R0001 | DC523, digital input/output module, <br> 24 DC, 24 V DC / 0.5 A, 1-wire | Active |
| 1SAP 440 500 R0001 | DC523-XC, digital input/output <br> module, 24 DC, 24 V DC / 0.5 A, <br> 1-wire, XC version | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

## DC532 - Digital input/output module

## Features

- 16 digital inputs 24 V DC, 16 configurable digital inputs/outputs
- Module-wise galvanically isolated
- Fast counter
- XC version for use in extreme ambient conditions available


1 I/O bus
2 Allocation between terminal number and signal name
316 yellow LEDs to display the signal states at the digital inputs (IO
416 yellow LEDs to display the signal states at the digital inputs/outputs (C16 ... C31)
51 green LED to display the state of the process supply voltage UP
64 red LEDs to display errors
7 Label
8 Terminal unit
DIN rail
Sign for XC version

## Intended purpose

The device can be used as a decentralized I/O extension module for S 500 communication interface modules (e. g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs.

Digital configurable input / output unit.

- 16 digital inputs 24 V DC in 2 groups (1.0 ... 1.7 and 2.0 ... 2.7)
- 16 digital configurable inputs/outputs 24 V DC (C16 ... C31) in 1 group ( $3.0 \ldots 3.7$ and 4.0 ... 4.7), of which each can be used
- as an input,
- as a transistor output with short circuit and overload protection, 0.5 A rated current or
- as a re-readable output (combined input/output) with the technical data of the digital inputs and outputs.
- Optional with fast counter

The configuration is performed by software. The modules are supplied with a process supply voltage of 24 V DC.

All available inputs/outputs are galvanically isolated from all other circuitry of the module. There is no potential separation between the channels within the same group.

For use in extreme ambient conditions (e.g. wider temperature and humidity range), a special $X C$ version of the device is available.

## Functionality

| Parameter | Value |
| :--- | :--- |
| Digital inputs | $16(24 \mathrm{~V} \mathrm{DC})$ |
| Digital inputs/outputs | $16(24 \mathrm{~V} \mathrm{DC})$ |
| Fast counter | Integrated, many configurable operating <br> modes (only with AC500) |
| LED displays | For signal states, errors and supply voltage |
| Internal power supply | Through the I/O bus interface (I/O bus) |
| External power supply | Via the terminals ZP and UP (process voltage <br> $24 ~ V ~ D C) ~$ |
| Required terminal unit | TU515 or TU516 © Chapter 5.2.5.2 "TU515, <br> TU516, TU541 and TU542 for I/O modules" <br> on page 938 |
| Effect of incorrect input terminal connection | Wrong or no signal detected, no damage up to <br> 35 V |

The device is plugged on a terminal unit ${ }^{\star}$ Chapter 5.2.5.2 "TU515, TU516, TU541 and TU542 for I/O modules" on page 938. Position the module properly and press until it locks in place. The terminal unit is either mounted on a DIN rail or to the wall using 2 screws plus the additional accessory for wall mounting ${ }^{\circledR} \Rightarrow$ Chapter 5.2.8.2.5 "TA526-Wall mounting accessory" on page 1324.


For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

## Connections

The connection of the I/O channels is carried out using the 40 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

The terminals $1.8 \ldots 4.8$ and $1.9 \ldots 4.9$ are electrically interconnected within the I/O terminal unit and always have the same assignment, irrespective of the inserted module:
Terminals $1.8 \ldots 4.8$ : process voltage UP $=+24 \mathrm{~V}$ DC
Terminals 1.9 ... 4.9: process voltage ZP $=0 \mathrm{~V}$ DC


1 I/O bus
2 4.0 ... 4.7: Connected with UP (switch) -> Input; Connected with ZP (load) -> Output
3 Control cabinet earth
The assignment of the other terminals:

| Terminals | Signal | Description |
| :--- | :--- | :--- |
| $1.0 \ldots 1.7$ | IO $\ldots$ I7 | 8 digital inputs |
| $2.0 \ldots 2.7$ | I8 .. I15 | 8 digital inputs |
| $3.0 \ldots 3.7$ | $\mathrm{C} 16 \ldots \mathrm{C} 23$ | 8 digital inputs/outputs |
| $4.0 \ldots 4.7$ | $\mathrm{C} 24 \ldots \mathrm{C} 31$ | 8 digital inputs/outputs |

## CAUTION!

The process supply voltage must be included in the grounding concept (e. g. grounding of the negative terminal).

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a communication interface module or a CPU). Thus, the current consumption from 24 V DC power supply at the terminals L+/UP and M/ZP of the CPU/communication interface module increases by 2 mA per DC532.
The external power supply connection is carried out via the UP (+24 V DC) and the ZP (0 V DC) terminals.

## WARNING!

## Removal/Insertion under power

Removal or insertion under power is permissible only if all conditions for hot swapping are fullfilled.
(4) Chapter 4.9.3 "Replace an I/O module with hot swap" on page 153

The devices are not designed for removal or insertion under power when the conditions for hot swap do not apply. Because of unforeseeable consequences, it is not allowed to plug in or unplug devices with the power being ON.
Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while they are energized in a hazardous location could result in an electric arc, which could create an ignition source resulting in fire or explosion.
Prior to proceeding, make sure that power is been disconnected and that the area has been thoroughly checked to ensure that flammable materials are not present.
The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

## Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.


The module provides several diagnosis functions.

## Internal data exchange

|  | Without the fast counter | With the fast counter (only <br> with AC500) |
| :--- | :--- | :--- |
| Digital inputs (bytes) | 4 | 6 |
| Digital outputs (bytes) | 2 | 4 |
| Counter input data (words) | 0 | 4 |
| Counter output data (words) | 0 | 8 |

## I/O configuration

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or communication interface module) during power-up of the system.

Hence, replacing I/O modules is possible without any re-parameterization via software.


If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.

## Parameterization

| Firmware version | Configuration |
| :--- | :--- |
| Firmware version > V2.0.0 | The arrangement of the parameter data is per- <br> formed by Control Builder Plus/ Automation <br> Builder software. |

The parameter data directly influences the functionality of modules.
For non-standard applications, it is necessary to adapt the parameters to your system configuration.
Module: Module slot address: $\mathrm{Y}=1$... 10

| Name | Value | Internal value | Internal value, type | Default | Min. | Max. |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Module ID | Internal | $\begin{aligned} & 1200 \\ & 1 \\ & 1 \end{aligned}$ | Word | $\begin{aligned} & \hline 1200 \\ & 0 \times 04 \mathrm{BO} \end{aligned}$ | 0 | 65535 | 0x0Y01 |
| Ignore module ${ }^{2}$ ) | $\begin{aligned} & \text { No } \\ & \text { Yes } \end{aligned}$ | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | Byte | $\begin{aligned} & \mathrm{No} \\ & \mathrm{Ox} 00 \end{aligned}$ |  |  | Not for FBP |
| Parameter length | Internal | 7 | Byte | $\begin{aligned} & \hline 7-\mathrm{CPU} \\ & 6 \text {-FBP } \end{aligned}$ | 0 | 255 | 0x0Y02 |
| Check supply | Off on | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | Byte | $\begin{aligned} & \text { On } \\ & \text { 0x01 } \end{aligned}$ | 0 | 1 | 0x0Y03 |
| Input delay | $\begin{aligned} & 0.1 \mathrm{~ms} \\ & 1 \mathrm{~ms} \\ & 8 \mathrm{~ms} \\ & 32 \mathrm{~ms} \end{aligned}$ | $\begin{aligned} & 0 \\ & 1 \\ & 2 \\ & 2 \\ & 3 \end{aligned}$ | Byte | $\begin{aligned} & 8 \mathrm{~ms} \\ & 0 \times 02 \end{aligned}$ | 0 | 3 | 0x0Y04 |
| Fast counter ${ }^{4}$ ) | $\begin{aligned} & \hline 0 \\ & : \\ & 10 \\ & \left.{ }^{3}\right) \end{aligned}$ |  | Byte | $\begin{aligned} & \text { Mode 0 } \\ & \text { Ox00 } \end{aligned}$ |  |  | Not for FBP |
| Output short circuit detection | $\begin{aligned} & \text { Off } \\ & \text { On } \end{aligned}$ | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | Byte | $\begin{array}{\|l\|} \hline \text { On } \\ \text { 0x01 } \end{array}$ | 0 | 1 | 0x0Y05 |
| Behaviour of outputs at com-munication errors | Off <br> Last value <br> Substitute value | $\begin{aligned} & \hline 0 \\ & 1+\left(n^{*} 5\right) \\ & 2+\left(n^{*} 5\right), \\ & n \leq 2 \end{aligned}$ | Byte | $\begin{aligned} & \text { Off } \\ & 0 \times 00 \end{aligned}$ | 0 | 2 | 0x0Y06 |
| Substitute value at outputs <br> Bit $15=$ Output 15 <br> Bit $0=$ Output 0 | $\begin{aligned} & 0 \ldots \\ & 65535 \end{aligned}$ | $\begin{aligned} & 0 \ldots \\ & 0 x f f f f \end{aligned}$ | Word | $\begin{aligned} & \hline 0 \\ & 0 \times 0000 \end{aligned}$ | 0 | 65535 | 0x0Y07 |

Remarks:

| ${ }^{1}$ ) | With CS31 and addresses smaller than 70 and FBP, the value is increased <br> by 1 |
| :--- | :--- |
| $\left.{ }^{2}\right)$ | Not with FBP |
| $\left.{ }^{3}\right)$ | For a description of the counter operating modes, please refer to the 'Fast <br> Counter' section \& Chapter 5.2.4.3.2.9 "Fast counter" on page 606 |
| ${ }^{4}$ ) | With FBP or CS31 without the parameter Fast Counter |

GSD file:

| Ext_User_Prm_Data_Len $=$ | 9 |
| :--- | :--- |
| Ext_User_Prm_Data_Const $(0)=$ | $0 \times 04,0 \times b 1,0 \times 06,1$ |
|  | $0 \times 01,0 \times 02,0 \times 01,0 \times 00,0 \times 00,0 \times 00 ;$ |

## State LEDs

During the power ON procedure, the module initializes automatically. All LEDs (except the channel LEDs) are ON during this time.


## Technical data

## Technical data of the module

The system data of AC500 and S500 are applicable to the standard version ${ }^{4}$ Chapter 5.1.2 "System data AC500" on page 166.

The system data of AC500-XC are applicable to the XC version $\Leftrightarrow$ Chapter 5.1.3 "System data AC500-XC" on page 169.

Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter | Value |
| :---: | :---: |
| Process supply voltage UP |  |
| Connections | Terminals $1.8,2.8,3.8$ and 4.8 for +24 V (UP) as well as 1.9, 2.9, 3.9 and 4.9 for $0 \mathrm{~V}(\mathrm{ZP})$ |
| Rated value | 24 V DC |
| Max. ripple | 5 \% |
| Protection against reversed voltage | Yes |
| Rated protection fuse on UP | 10 A fast |
| Galvanic isolation | Yes, per module |
| Current consumption |  |
| From 24 V DC power supply at the terminals UP/L+ and ZP/M of the CPU/communication interface module | Ca. 2 mA |
| From UP at normal operation / with outputs | 0.15 A + max. 0.5 A per output |
| Inrush current from UP (at power up) | $0.007 \mathrm{~A}^{2} \mathrm{~s}$ |
| Max. power dissipation within the module | 6 W (outputs unloaded) |
| Weight (without terminal unit) | ca. 125 g |
| Mounting position | Horizontal <br> Or vertical with derating (output load reduced to $50 \%$ at $+40^{\circ} \mathrm{C}$ per group) |
| Cooling | The natural convection cooling must not be hindered by cable ducts or other parts in the control cabinet. |

## - NOTICE!

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and temporary overvoltage up to 30 V DC.

## Multiple overloads

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an internal smart high-side switch.

## Technical data of the digital inputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 16 |
| Distribution of the channels into groups | 1 group of 16 channels |
| Terminals of the channels I0 ... I7 | $1.0 \ldots 1.7$ |
| Terminals of the channels I8 ... I15 | $2.0 \ldots 2.7$ |
| Reference potential for all inputs | Terminals $1.9,2.8,3.8$ and 4.9 (negative pole of <br> the process supply voltage, signal name ZP) |
| Galvanic isolation | From the rest of the module (I/O bus) |


| Parameter | Value |
| :--- | :--- |
| Indication of the input signals | 1 yellow LED per channel, the LED is ON when <br> the input signal is high (signal 1) |
| Monitoring point of input indicator | LED is part of the input circuitry |
| Input type acc. to EN 61131-2 | Type 1 |
| Input delay (0->1 or 1->0) | Typ. 8 ms, configurable from 0.1 to 32 ms |
| Input signal voltage | 24 V DC |
|  | Signal 0 |
| Undefined signal | $-3 \mathrm{~V} \ldots+5 \mathrm{~V}$ |
|  | $>+5 \mathrm{~V} \ldots<+15 \mathrm{~V}$ <br> Signal 1 |
| Ripple with signal 0 | $+15 \mathrm{~V} \ldots+30 \mathrm{~V}$ |
| Ripple with signal 1 | Within $-3 \mathrm{~V} \ldots+5 \mathrm{~V}$ |
| Input current per channel | Within $+15 \mathrm{~V} \ldots+30 \mathrm{~V}$ |
|  | Input voltage +24 V |
| Input voltage +5 V | Typ. 5 mA |
|  | Input voltage +15 V |
| Input voltage +30 V | $>1 \mathrm{~mA}$ |
| Max. cable length | $>5 \mathrm{~mA}$ |
|  | Shielded |
| Unshielded | $<8 \mathrm{~mA}$ |

## Technical data of the configurable digital inputs/outputs

Each of the configurable I/O channels is defined as input or output by the user program. This is done by interrogating or allocating the corresponding channel.

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 16 inputs/outputs (with transistors) |
| Distribution of the channels into groups | 1 group of 16 channels |
| If the channels are used as inputs |  |
|  | Channels I16 ... I23 |
|  | Channels I24 ... I31 |
| If the channels are used as outputs | Terminals $4.0 \ldots 4.7$ |
|  | Channels Q16 ... Q23 |
|  | Channels Q24 ... Q31 |
| Indication of the input/output signals | Terminals $3.0 \ldots 3.7$ |
| Monitoring point of input/output indicator | Terminals 4.0 ... 4.7 <br> the input/output signal is high (signal 1) |
| Galvanic isolation | LED is part of the input circuitry |

Technical data of the digital inputs/outputs if used as inputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | Max. 16 digital inputs |
| Reference potential for all inputs | Terminals 1.9, 2.9, 3.9 and 4.9 (negative pole of <br> the process supply voltage, signal name ZP) |
| Input current, per channel | See 'Technical Data of the Digital Inputs' <br> ¿ Chapter 5.2.4.3.2.3.9.2 "Technical data of <br> the digital inputs" on page 548 |
| Input type acc. to EN 61131-2 | Type 1 |
| Input delay (0->1 or 1->0) | Typ. 8 ms, configurable from 0.1 to 32 ms |
| Input signal voltage | 24 V DC |
| Signal 0 | $-3 \mathrm{~V} \ldots+5 \mathrm{~V}$ *) |
| undefined signal | $>+5 \mathrm{~V} \ldots<+15 \mathrm{~V}$ |
| Signal 1 | $+15 \mathrm{~V} \ldots+30 \mathrm{~V}$ |
| Ripple with signal 0 | Within $-3 \mathrm{~V} \ldots+5 \mathrm{~V}$ *) |
| Ripple with signal 1 | Within $+15 \mathrm{~V} \ldots+30 \mathrm{~V}$ |
| Max. cable length | 1000 m |
|  | Shielded |
| Unshielded | 600 m |

*) Due to the direct connection to the output, the demagnetizing varistor is also effective at the input (see figure) above. This is why the difference between UPx and the input signal must not exceed the clamp voltage of the varistor. The varistor limits the clamp voltage to approx. 36 V . Consequently, the input voltage must range from $-12 \mathrm{~V} \ldots+30 \mathrm{~V}$ when $\mathrm{UPx}=24 \mathrm{~V}$ and from $-6 \mathrm{~V} . .+30 \mathrm{~V}$ when UPx $=30 \mathrm{~V}$.

## Technical data of the digital inputs/outputs if used as outputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | Max. 16 transistor outputs |
| Reference potential for all outputs | Terminals 1.9, 2.9, 3.9 and 4.9 (negative pole <br> of the process supply voltage, signal name ZP) |
| Common power supply voltage | For all outputs: terminals 1.8, 2.8, 3.8 and 4.8 <br> (positive pole of the process supply voltage, <br> signal name UP) |
| Output voltage for signal 1 | UP (-0.8 V) |
| Output delay (0->1 or 1->0) | On request |
| Output current | 5 |
|  | Rated value, per channel |
| Maximum value (all channels together) | 8 A |
| Leakage current with signal 0 | $<0.5$ mA |
| Rated protection fuse on UP | 10 A fast |
| Demagnetization when inductive loads are <br> switched off | With varistors integrated in the module (see <br> figure below) |
| Switching frequency |  |
|  | With resistive load |


| Parameter |  | Value |
| :--- | :--- | :--- |
| With inductive loads | Max. 0.5 Hz |  |
|  | With lamp loads | Max. 11 Hz with max. 5 W |
| Short-circuit-proof / overload-proof | Yes |  |
| Overload message (I > 0.7 A) | Yes, after ca. 100 ms |  |
| Output current limitation | Yes, automatic reactivation after short cir- <br> cuit/overload |  |
| Resistance to feedback against 24 V signals | Yes |  |
| Max. cable length |  |  |
|  | Shielded | 1000 m |
|  | Unshielded | 600 m |

The following drawing shows the circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off.


Fig. 76: Digital input/output (circuit diagram)
1 UPx (+ 24 V )
2 Digital input/output
3 ZPx (0 V)
4 For demagnization when inductive loads are switched off

## Technical data of the fast counter

The fast counter of the module does not work if the module is connected to a

- FBP interface module
- CS31 bus module
- CANopen communication interface module

| Parameter | Value |
| :--- | :--- |
| Used inputs | $\mathrm{C} 24 / \mathrm{C} 25$ |
| Used outputs | C 26 |
| Counting frequency | Max. 50 kHz |

## Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm

The dimensions are in mm and in brackets in inch.

Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 240 100 R0001 | DC532, digital input/output module, <br> 16 DI, 16 DC, 24 V DC / 0.5 A, 1-wire | Active |
| 1SAP 440 100 R0001 | DC532-XC, digital input/output <br> module, 16 DI, 16 DC, <br> $24 ~ V ~ D C ~ / ~ 0.5 ~ A, ~ 1-w i r e, ~ X C ~ v e r s i o n ~$ | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

## DI524 - Digital input module

## Features

- 32 digital inputs 24 V DC in 4 groups (1.0 ... 1.7, 2.0 ... 2.7, 3.0 ... 3.7 and $4.0 \ldots 4.7$ )
- Fast counter
- Module-wise galvanically isolated
- XC version for use in extreme ambient conditions available


1 I/O bus
2 Allocation between terminal number and signal name
332 yellow LEDs to display the signal states at the digital inputs (IO ... I31)
41 green LED to display the state of the process supply voltage UP
54 red LEDs to display errors
6 Label
7 Terminal unit
8 DIN rail


## Intended purpose

The device can be used as a decentralized I/O extension module for S500 communication interface modules (e. g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs.

The configuration is performed by software. The modules are supplied with a process supply voltage of 24 V DC.
All available inputs/outputs are galvanically isolated from all other circuitry of the module. There is no potential separation between the channels within the same group.

For use in extreme ambient conditions (e.g. wider temperature and humidity range), a special $X C$ version of the device is available.

## Functionality

| Parameter | Value |
| :--- | :--- |
| Fast counter | Integrated, many configurable operating modes (only <br> with AC500) |
| LED displays | For signal states, errors and supply voltage |
| Internal power supply | Via the I/O bus interface (I/O bus) |
| External power supply | Via the terminals ZP and UP (process voltage <br> 24 V DC) |
| Required terminal units | TU515 or TU516 \% Chapter 5.2.5.2 "TU515, TU516, <br> TU541 and TU542 for I/O modules" on page 938 |
| Effect of incorrect input terminal con- <br> nection | Wrong or no signal detected, no damage up to 35 V |

The device is plugged on a terminal unit ${ }^{\star} \Rightarrow$ Chapter 5.2.5.2 "TU515, TU516, TU541 and TU542 for I/O modules" on page 938. Position the module properly and press until it locks in place. The terminal unit is either mounted on a DIN rail or to the wall using 2 screws plus the additional accessory for wall mounting ${ }^{\text {}} \gg$ Chapter 5.2.8.2.5 "TA526 - Wall mounting accessory" on page 1324.

## Connections

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

The connection of the I/O channels is carried out using the 40 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

The terminals $1.8 \ldots 4.8$ and $1.9 \ldots 4.9$ are electrically interconnected within the I/O terminal unit and have always the same assignment, irrespective of the inserted module:

Terminals 1.8 ... 4.8: process voltage UP $=+24 \mathrm{~V}$ DC
Terminals $1.9 \ldots 4.9$ process voltage $\mathrm{ZP}=0 \mathrm{~V}$ DC

Table 136: Assignment of the other terminals:

| Terminals | Signal | Description |
| :--- | :--- | :--- |
| $1.0 \ldots 1.7$ | $\mathrm{I} 0 \ldots \mathrm{I} 7$ | 8 digital inputs |
| $2.0 \ldots 2.7$ | $\mathrm{I} 8 \ldots \mathrm{I} 15$ | 8 digital inputs |
| $3.0 \ldots 3.7$ | $\mathrm{I} 16 \ldots \mathrm{I} 23$ | 8 digital inputs |
| $4.0 \ldots 4.7$ | $\mathrm{I} 24 \ldots \mathrm{I} 31$ | 8 digital inputs |

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a communication interface module or a CPU). Thus, the current consumption from 24 V DC power supply at the terminals L+/UP and M/ZP of the CPU/communication interface module increases by 2 mA per DI524.

The external power supply connection is carried out via the UP (+24 V DC) and the ZP (0 V DC) terminals.

## WARNING!

## Removal/Insertion under power

Removal or insertion under power is permissible only if all conditions for hot swapping are fullfilled.
« Chapter 4.9.3 "Replace an I/O module with hot swap" on page 153
The devices are not designed for removal or insertion under power when the conditions for hot swap do not apply. Because of unforeseeable consequences, it is not allowed to plug in or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while they are energized in a hazardous location could result in an electric arc, which could create an ignition source resulting in fire or explosion.

Prior to proceeding, make sure that power is been disconnected and that the area has been thoroughly checked to ensure that flammable materials are not present.
The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

## Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.


1 I/O bus
2 Control cabinet earth

## CAUTION!

The process supply voltage must be included in the grounding concept (e. g. grounding of the negative terminal).

The module provides several diagnosis functions.

## Internal data exchange

|  | Without the fast counter | With the fast counter (only <br> with AC500) |
| :--- | :--- | :--- |
| Digital inputs (bytes) | 4 | 6 |
| Digital outputs (bytes) | 0 | 2 |


|  | Without the fast counter | With the fast counter (only <br> with AC500) |
| :--- | :--- | :--- |
| Counter input data (words) | 0 | 4 |
| Counter output data (words) | 0 | 8 |

## I/O configuration

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or communication interface module) during power-up of the system.

Hence, replacing I/O modules is possible without any re-parameterization via software.


If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.

## Parameterization

| Firmware version | Configuration |
| :--- | :--- |
| Firmware version > V2.0.0 | The arrangement of the parameter data is per- <br> formed by Control Builder Plus/ Automation <br> Builder software. |

The parameter data directly influences the functionality of modules.
For non-standard applications, it is necessary to adapt the parameters to your system configuration.
Module: Module slot address: $\mathrm{Y}=1$... 10

| No. | Name | Value | Internal value | Internal value, type | Default | Min. | Max. | EDS Slot/ Index |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 1 | Module ID | Internal | $1000$ <br> $\left.{ }^{1}\right)$ | Word | $\begin{array}{\|l\|} \hline 1000 \\ 0 x 03 E 8 \end{array}$ | 0 | 65535 | 0x0Y01 |
| 2 | Ignore module ${ }^{2}$ ) | No Yes | $\begin{array}{\|l\|} \hline 0 \\ 1 \end{array}$ | Byte | $\begin{array}{\|l\|} \hline \text { No } \\ 0 \times 00 \end{array}$ |  |  | Not for FBP |
| 3 | Parameter length | Internal | $\begin{aligned} & 3-C P U \\ & 2-F B P \end{aligned}$ | Byte | $\begin{aligned} & 3 \\ & 2 \end{aligned}$ | 0 | 255 | 0x0Y02 |
| 4 | Check supply | $\begin{aligned} & \text { Off } \\ & \text { On } \end{aligned}$ | $\begin{array}{\|l\|} \hline 0 \\ 1 \end{array}$ | Byte | $\begin{array}{\|l\|} \hline \text { On } \\ 0 \times 01 \end{array}$ | 0 | 1 | 0x0Y03 |


| No. | Name | Value | Internal <br> value | Internal <br> value, <br> type | Default | Min. | Max. | EDS <br> Slot/ <br> Index |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| 5 | Input <br> delay | 0.1 ms <br> 1 ms <br> 8 ms <br> 32 ms | 0 <br> 2 <br> 2 | Byte | 8 ms <br> $0 \times 02$ | 0 | 3 | $0 \times$ YY04 |
| 6 | Fast <br> counter <br> 4 | 0 <br> $:$ <br> 10 <br> $3)$ | 0 <br> 10 | Byte | Mode 0 <br> $0 \times 00$ |  |  | Not for <br> FBP |

Remarks:

| $\left.{ }^{1}\right)$ | With CS31 and addresses smaller than 70 and FBP, the value is increased <br> by 1 |
| :--- | :--- |
| $\left.{ }^{2}\right)$ | Not with FBP |
| $\left.{ }^{3}\right)$ | For a description of the counter operating modes, please refer to the 'Fast <br> Counter' section <br> C Chapter 5.2.4.3.2.9 "Fast counter" on page 606 |
| $\left.{ }^{4}\right)$ | With FBP or CS31 without the parameter Fast counter |

GSD file:

| Ext_User_Prm_Data_Len $=$ |  |
| :--- | :--- |
| Ext_User_Prm_Data_Const $(0)=$ | 5 |
|  | $0 \times 03,0 \times e 9,0 \times 02,1$ |
| $0 \times 01,0 \times 02 ;$ |  |

State LEDs
During the power ON procedure, the module initializes automatically. All LEDs (except the channel LEDs) are ON during this time.



## Technical data

## Technical data of the module

The system data of AC500 and S500 are applicable to the standard version ${ }^{\mu}$ Chapter 5.1.2 "System data AC500" on page 166.
The system data of AC500-XC are applicable to the XC version ${ }^{〔}$ Chapter 5.1.3 "System data AC500-XC" on page 169.

Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter | Value |
| :---: | :---: |
| Process supply voltage UP |  |
| Connections | Terminals 1.8, 2.8, 3.8 and 4.8 for +24 V (UP) as well as 1.9, 2.9, 3.9 and 4.9 for $0 \mathrm{~V}(\mathrm{ZP})$ |
| Rated value | 24 V DC |
| Max. ripple | 5 \% |
| Protection against reversed voltage | Yes |
| Rated protection fuse for UP | 10 A fast |
| Galvanic isolation | Yes, per module |
| Current consumption |  |
| From 24 V DC power supply at the terminals UP/L+ and ZP/M of the CPU/ communication interface module | ca. 2 mA |
| From UP at normal operation | 0.15 A |
| Inrush current from UP (at power up) | $0.008 \mathrm{~A}^{2} \mathrm{~s}$ |
| Weight (without terminal unit) | ca. 105 g |
| Mounting position | Horizontal or vertical with derating (output load reduced to $50 \%$ at $+40^{\circ} \mathrm{C}$ per group) |
| Cooling | The natural convection cooling must not be hindered by cable ducts or other parts in the control cabinet. |

## NOTICE!

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and temporary overvoltage up to 30 V DC.

## Technical data of the digital inputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 32 |
| Distribution of the channels into groups | 1 group of 32 channels |
| Terminals of the channels 10 ... 17 | 1.0 ... 1.7 |
| Terminals of the channels 18... 115 | 2.0 ... 2.7 |
| Terminals of the channels I16 ... 123 | 3.0 ... 3.7 |
| Terminals of the channels I24 ... I31 | 4.0 ... 4.7 |
| Reference potential for all inputs | Terminals 1.9, 2.9, 3.9 and 4.9 (negative pole of the process supply voltage, signal name ZP) |
| Galvanic isolation | From the rest of the module (I/O bus) |
| Indication of the input signals | One yellow LED per channel, the LED is ON when the input signal is high (signal 1) |
| Monitoring point of input indicator | LED is part of the input circuitry |
| Input type acc. to EN 61131-2 | Type 1 |
| Input delay (0-> 1 or 1 -> 0 ) | Typ. 8 ms , configurable from $0.1 \ldots 32 \mathrm{~ms}$ |
| Input signal voltage | 24 V DC |
| Signal 0 | -3 V ... +5V |
| Undefined signal | > +5V ... < +15 V |
| Signal 1 | +15 V ... +30 V |
| Ripple with signal 0 | Within -3V ... +5V |
| Ripple with signal 1 | Within +15 V ... +30 V |
| Input current per channel |  |
| Input voltage +24 V | Typ. 5 mA |
| Input voltage +5 V | $>1 \mathrm{~mA}$ |
| Input voltage +15 V | $>5 \mathrm{~mA}$ |
| Input voltage +30 V | $<8 \mathrm{~mA}$ |
| Max. cable length |  |
| Shielded | 1000 m |
| Unshielded | 600 m |

## Technical data of the fast counter

The fast counter of the module does not work if the module is connected to a

- FBP interface module
- CS31 bus module
- CANopen communication interface module

| Parameter | Value |
| :--- | :--- |
| Used inputs | $\mathrm{I} 24 / \mathrm{I} 25$ |
| Used outputs | None |
| Counting frequency | Max. 50 kHz |

(3) Chapter 6.8.2.12 "Fast counters in AC500 devices" on page 4536

## Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm

The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 240 000 R0001 | DI524, digital input module, 32 DI, <br> 24 V DC, 1-wire | Active |
| 1SAP 440 000 R0001 | DI524-XC, digital input module, 32 DI, <br> 24 V DC, 1-wire, XC version | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

## DO524 - Digital output module

## Features

- 32 digital outputs 24 V DC / 0.5 A in 4 groups ( 1.0 ... 4.7) with short circuit and overload protection
- Module-wise galvanically isolated
- XC version for use in extreme ambient conditions available


1 I/O bus
2 Allocation between terminal number and signal name
332 yellow LEDs to display the signal states at the digital outputs (O0 ... O31)
41 green LED to display the state of the process supply voltage UP
54 red LEDs to display errors
6 Label
7 Terminal unit
8 DIN rail
Sign for XC version

## Intended purpose

The device can be used as a decentralized I/O extension module for S 500 communication interface modules (e. g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs.

The outputs are galvanically isolated from all other circuitry of the module. There is no potential separation between the channels.

| Parameter | Value |
| :--- | :--- |
| LED displays | For signal states, errors and supply voltage |
| Internal power supply | Via the I/O bus interface (I/O bus) |
| External power supply | Via the terminals ZP and UP (process voltage <br> $24 ~ V ~ D C) ~$ |
| Required terminal unit | TU515 or TU516 \& Chapter 5.2.5.2 "TU515, TU516, <br> TU541 and TU542 for I/O modules" on page 938 |

The device is plugged on a terminal unit ${ }^{\star}$ Chapter 5.2.5.2 "TU515, TU516, TU541 and TU542 for I/O modules" on page 938 . Position the module properly and press until it locks in place. The terminal unit is either mounted on a DIN rail or to the wall using 2 screws plus the additional accessory for wall mounting ${ }^{\star} \Rightarrow$ Chapter 5.2.8.2.5 "TA526-Wall mounting accessory" on page 1324.

## Multiple overloads

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an internal smart high-side switch.

## Connections

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

The connection of the I/O channels is carried out using the 40 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.
The terminals $1.8 \ldots 4.8$ and $1.9 \ldots 4.9$ are electrically interconnected within the I/O terminal unit and have always the same assignment, independent of the inserted module:
Terminals $1.8 \ldots 4.8$ : process voltage UP $=+24 \mathrm{~V}$ DC
Terminals $1.9 \ldots 4.9$ : process voltage $\mathrm{ZP}=0 \mathrm{~V}$ DC
The assignment of the other terminals:

| Terminals | Signal | Description |
| :--- | :--- | :--- |
| $1.0 \ldots 1.7$ | $\mathrm{O} 0 \ldots \mathrm{O} 7$ | 8 digital outputs |
| $2.0 \ldots 2.7$ | $\mathrm{O} 8 \ldots \mathrm{O} 15$ | 8 digital outputs |
| $3.0 \ldots 3.7$ | $\mathrm{O} 16 \ldots \mathrm{O} 23$ | 8 digital outputs |
| $4.0 \ldots 4.7$ | $\mathrm{O} 24 \ldots \mathrm{O} 31$ | 8 digital outputs |

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a communication interface module or a CPU). Thus, the current consumption from 24 V DC power supply at the terminals L+/UP and M/ZP of the CPU/communication interface module increases by 2 mA per DO524.
The external power supply connection is carried out via the UP (+24 V DC) and the ZP (0 V DC) terminals.

WARNING!

## Removal/Insertion under power

Removal or insertion under power is permissible only if all conditions for hot swapping are fullfilled.
« Chapter 4.9.3 "Replace an I/O module with hot swap" on page 153
The devices are not designed for removal or insertion under power when the conditions for hot swap do not apply. Because of unforeseeable consequences, it is not allowed to plug in or unplug devices with the power being ON.
Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while they are energized in a hazardous location could result in an electric arc, which could create an ignition source resulting in fire or explosion.

Prior to proceeding, make sure that power is been disconnected and that the area has been thoroughly checked to ensure that flammable materials are not present.

The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

## Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.


Fig. 77: Internal construction of the digital outputs
The module provides several diagnosis functions.

## Internal data exchange

| Digital inputs (bytes) | 0 |
| :--- | :--- |
| Digital outputs (bytes) | 4 |

## I/O configuration

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or communication interface module) during power-up of the system.
Hence, replacing I/O modules is possible without any re-parameterization via software.


If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.

## Parameterization

| Firmware version | Configuration |
| :--- | :--- |
| Firmware version $>$ V2.0.0 | The arrangement of the parameter data is per- <br> formed by Control Builder Plus/ Automation <br> Builder software. |

The parameter data directly influences the functionality of modules.
For non-standard applications, it is necessary to adapt the parameters to your system configuration.
Module: Module slot address: $\mathrm{Y}=1$... 10

| Name | Value | Internal value | Internal value, type | Default | Min. | Max. | Max. |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Module ID | Internal | $\begin{aligned} & \hline 1101 \\ & 1 \\ & \hline \end{aligned}$ | WORD | $\begin{aligned} & 1101 \\ & 0 \times 044 \mathrm{D} \end{aligned}$ | 0 | 65535 | 0x0Y01 |
| Ignore module ${ }^{2}$ ) | $\begin{aligned} & \text { No } \\ & \text { Yes } \end{aligned}$ | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | BYTE | $\begin{aligned} & \mathrm{No} \\ & 0 \times 00 \end{aligned}$ |  |  | not for FBP |
| Parameter length | Internal | 7 | BYTE | $\begin{aligned} & 7-\mathrm{CPU} \\ & 7-\mathrm{FBP} \end{aligned}$ | 0 | 255 | 0x0Y02 |
| Check supply | $\begin{aligned} & \text { Off } \\ & \text { on } \end{aligned}$ | $0$ | BYTE | $\begin{array}{\|l\|} \hline \text { On } \\ 0 \times 01 \end{array}$ | 0 | 1 | 0x0Y03 |
| Output short circuit detection | $\begin{aligned} & \text { Off } \\ & \text { On } \end{aligned}$ | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | BYTE | $\begin{aligned} & \text { On } \\ & 0 \times 01 \end{aligned}$ | 0 | 1 | 0x0Y04 |


| Name | Value | Internal value | Internal value, type | Default | Min. | Max. | Max. |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Behaviour of outputs at com-munication errors | Off <br> Last value <br> Substitute value | $\begin{aligned} & 0 \\ & 1+\left(n^{*} 5\right) \\ & 2+(n * 5), \\ & n \leq 2 \end{aligned}$ | BYTE | $\begin{array}{\|l\|} \hline \text { Off } \\ 0 \times 00 \end{array}$ | 0 | 2 | 0x0Y05 |
| Substitute value at outputs Bit 31 = Output 31 Bit $0=$ Output 0 | $\begin{aligned} & 0 \ldots \\ & 42949672 \\ & 95 \end{aligned}$ | 0 ... <br> 0xffffffff | DWORD | $\begin{aligned} & \hline 0 \\ & 0 \times 000000 \\ & 00 \end{aligned}$ | 0 | $\begin{aligned} & 42949672 \\ & 95 \end{aligned}$ | 0x0Y06 |

${ }^{1}$ ) With CS31 and addresses smaller than 70 and FBP, the value is increased by 1
${ }^{2}$ ) Not with FBP
GSD file:

| Ext_User_Prm_Data_Len $=$ | 10 |
| :--- | :--- |
| Ext_User_Prm_Data_Const $(0)=$ | $0 \times 04,0 \times 4 \mathrm{~d}, 0 \times 07,1$ |
|  | $0 \times 01,0 \times 01,0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00 ;$ |

State LEDs
During the power ON procedure, the module initializes automatically. All LEDs (except the channel LEDs) are ON during this time.

| LED |  | State | Color | LED = OFF | LED = ON | LED flashes |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| AH13 DO524 | Outputs $\mathrm{O} \ldots \mathrm{O} 31$ | Digital output | Yellow | Output = OFF | Output $=$ ON | -- |
|  | UP | Process supply voltage 24 V DC via terminal | Green | Process supply voltage is missing | Process supply voltage OK | -- |
|  | CH-ERR1 | Channel error, error messages in groups (digital outputs combined into the groups 1, 2, 3, 4) | Red | No error or process supply voltage is missing | Severe error within the corresponding group | Error on one channel of the corresponding group (e.g. short circuit at an output) |
|  | CH-ERR2 |  | Red |  |  |  |
| CHERR1 ${ }_{\text {CHERR2 }}$ CHERR3 ${ }_{\text {cherra }}$ | CH-ERR3 |  | Red |  |  |  |
|  | CH-ERR4 |  | Red |  |  |  |
|  | CH-ERR *) | Module error | Red | -- | Internal error | -- |
|  | *) All of the LEDs CH-ERR1 to CH-ERR4 light up together |  |  |  |  |  |

## Technical data

## Technical data of the module

The system data of AC500 and S500 are applicable to the standard version Chapter 5.1.2 "System data AC500" on page 166.
The system data of AC500-XC are applicable to the XC version $\Leftrightarrow$ Chapter 5.1.3 "System data AC500-XC" on page 169.

Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter | Value |
| :---: | :---: |
| Process supply voltage UP |  |
| Connections | Terminals $1.8,2.8,3.8$ and 4.8 for +24 V (UP) as well as 1.9, 2.9, 3.9 and 4.9 for 0 V (ZP) |
| Rated value | 24 V DC |
| Max. ripple | 5 \% |
| Protection against reversed voltage | Yes |
| Rated protection fuse on UP | 10 A fast |
| Galvanic isolation | Yes, per module |
| Current consumption |  |
| From 24 V DC power supply at the terminals UP/L+ and ZP/M of the CPU/communication interface module | Ca. 2 mA |
| From UP at normal operation / with outputs | 0.10 A + max. 0.5 A per output |
| Inrush current from UP (at power up) | $0.005 \mathrm{~A}^{2} \mathrm{~s}$ |
| Max. power dissipation within the module | 6 W (outputs unloaded) |
| Weight (without terminal unit) | Ca. 100 g |
| Mounting position | Horizontal <br> Or vertical with derating (output load reduced to $50 \%$ at $+40^{\circ} \mathrm{C}$ per group) |
| Cooling | The natural convection cooling must not be hindered by cable ducts or other parts in the control cabinet. |

## NOTICE!

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and temporary overvoltage up to 30 V DC.

## Multiple overloads

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an internal smart high-side switch.

## Technical data of the digital outputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 32 outputs (with transistors) |
| Distribution of the channels into groups | 1 group of 32 channels |
| Connection of the channels |  |
| O0 ... 07 | Terminals 1.0 ... 1.7 |
| O8 ... 015 | Terminals 2.0 ... 2.7 |
| O16 ... O23 | Terminals 3.0 ... 3.7 |
| O24 ... O31 | Terminals 4.0 ... 4.7 |
| Indication of the output signals | 1 yellow LED per channel, the LED is ON if the output signal is high (signal 1 ) |
| Reference potential for all outputs | Terminals 1.9, 2.9, 3.9 and 4.9 (negative pole of the process supply voltage, signal name ZP ) |
| Common power supply voltage | For all outputs: terminals $1.8,2.8,3.8$ and 4.8 (positive pole of the process supply voltage, signal name UP) |
| Output voltage for signal 1 | UP (-0.8 V) |
| Output delay (0 -> 1 or 1 -> 0) | On request |
| Output current |  |
| Rated value, per channel | 500 mA at $\mathrm{UP}=24 \mathrm{~V}$ |
| Maximum value (channels O0 ... O15) | 4 A |
| Maximum value (channels O16 ... O31) | 4 A |
| Maximum value (all channels together) | 8 A |
| Max. leakage current with signal 0 | $<0.5 \mathrm{~mA}$ |
| Rated protection fuse on UP | 10 A fast |
| Demagnetization when inductive loads are switched off | With varistors integrated in the module (see figure below) |
| Switching frequency |  |
| With resistive load | On request |
| With inductive loads | Max. 0.5 Hz |
| With lamp loads | Max. 11 Hz with max. 5 W |
| Short-circuit proof / overload proof | Yes |
| Overload message ( $1>0.7$ A) | Yes, after ca. 100 ms |
| Output current limitation | Yes, automatic reactivation after short-circuit/overload |
| Resistance to feedback against 24 V signals | Yes |
| Max. cable length |  |
| Shielded | 1000 m |
| Unshielded | 600 m |

The following drawing shows the circuitry of a digital output with the varistors for demagnetization when inductive loads are switched off.


## Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm


The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 240 700 R0001 | DO524, digital output module, 32 DO, <br> 24 V DC / 0.5 A, 1-wire | Active |
| 1SAP 440 700 R0001 | DO524-XC, digital output module, <br> 32 DO, 24 V DC / 0.5 A, 1-wire, <br> XC version | Active |

${ }^{*}$ ) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

## DO526 - Digital output module

## Features

- 8 digital outputs 24 V DC ( $\mathrm{O} 0 \ldots \mathrm{O}$ ) in 2 groups without short circuit and without overload protection.
- Module and group-wise galvanically isolated
- XC version for use in extreme ambient conditions available


1 I/O bus
2 Allocation between terminal number and signal name
38 yellow LEDs to display the signal states of the outputs $\mathrm{O} 0 \ldots \mathrm{O} 7$
43 green LEDs to display the states of the process supply voltage UP, UP3 and UP4
52 red LEDs to display errors
6 Label
7 Terminal unit
8 DIN-rail
Sign for XC version

## Intended purpose

The device can be used as a decentralized I/O extension module for S500 communication interface modules (e. g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs.

The outputs are group-wise galvanically isolated from each other.
All other circuitry of the module is galvanically isolated from the outputs.
Potential separation between the channel groups.

| Parameter | Value |
| :--- | :--- |
| LED displays | For signal states, errors and supply voltages |
| Internal power supply | Via I/O bus |
| External power supply | Via the terminals ZP, ZP3, ZP4, UP, UP3 and UP4 <br> (process voltage 24 V DC) |
| Required terminal unit | TU542 « Chapter 5.2.5.2 "TU515, TU516, TU541 <br> and TU542 for I/O modules" on page 938 |

The output module is plugged on the terminal unit TU542. Properly position the module and press until it locks in place. The terminal unit is mounted on a DIN rail or with 2 screws plus the additional accessory for wall mounting ${ }^{\mu}$ ² Chapter 5.2.8.2.5 "TA526-Wall mounting accessory" on page 1324.

## Connections



For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

The connection of the I/O channels is carried out using the 40 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.
The terminals $1.8 \ldots 2.8$ and $1.9 \ldots 2.9$ are electrically interconnected within the I/O terminal unit and always have the same assignment, irrespective of the inserted module:

| Terminals 1.8 ... 2.8: | Process voltage UP $=+24 \mathrm{~V}$ DC |
| :--- | :--- |
| Terminals 1.9 ... 2.9: | Process voltage ZP $=0 \mathrm{~V}$ |
| Terminal 3.8: | Process voltage UP3 $=+24 \mathrm{~V}$ DC |
| Terminal 3.9: | Process voltage ZP3 $=0 \mathrm{~V}$ |
| Terminal 4.8: | Process voltage UP4 $=+24 \mathrm{~V}$ DC |
| Terminal 4.9: | Process voltage ZP4 $=0 \mathrm{~V}$ |


| Terminals | Signal | Description |
| :--- | :--- | :--- |
| $3.0,3.1,3.4,3.5$ | O0 ... O3 | 4 digital outputs |
| $4.0,4.1,4.4,4.5$ | $\mathrm{O} 4 \ldots \mathrm{O} 7$ | 4 digital outputs |

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a communication interface module or a CPU). Thus, the current consumption from 24 V DC power supply at the terminals L+/UP and M/ZP of the CPU/communication interface module increases by 2 mA per DO526.
The external power supply connection is carried out via the UP, UP3, UP4 (+24 V DC) and the ZP, ZP3, ZP4 (0 V DC) terminals.

## NOTICE!

## Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.


Fig. 78: Internal construction of the digital outputs


I/O bus
2 4.0 ... 4.7: Connected with UP (switch) -> Input;
Connected with ZP (load) -> Output
3 Control cabinet earth

## CAUTION!

The process supply voltage must be included in the grounding concept (e. g. grounding of the negative terminal).

The module provides several diagnosis functions.

## Internal data exchange

| Digital inputs (bytes) | 0 |
| :--- | :--- |
| Digital outputs (bytes) | 1 |

## I/O configuration

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or communication interface module) during power-up of the system.
Hence, replacing I/O modules is possible without any re-parameterization via software.


If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.

## Parameterization

The arrangement of the parameter data is performed with Automation Builder software, versions $\geq$ 1.2.3.
The parameter data directly influences the functionality of modules.
For non-standard applications, it is necessary to adapt the parameters to your system configuration.

Module: Module slot address: $\mathrm{Y}=1$... 7

| Name | Value | Internal value | Internal value, type | Default | Min. | Max. | Max. |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Module ID | Internal | $\begin{aligned} & 1105 \\ & \left.1{ }^{1}\right) \end{aligned}$ | WORD | $\begin{aligned} & \hline 1105 \\ & 0 \times 0451 \end{aligned}$ | 0 | 65535 | 0x0Y01 |
| Ignore module ${ }^{2}$ ) | No Yes | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | BYTE | $\begin{array}{\|l\|} \mathrm{No} \\ 0 \times 00 \end{array}$ |  |  | not for FBP |
| Parameter length | Internal | 6 | BYTE | $\begin{aligned} & \text { 6-CPU } \\ & 6-\mathrm{FBP} \end{aligned}$ | 0 | 6 | 0x0Y02 |
| Check supply | Off on | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | BYTE | $\begin{aligned} & \mathrm{On} \\ & 0 \times 01 \end{aligned}$ | 0 | 1 | 0x0Y03 |
| Reserve | $0 \ldots 255$ | $0 \ldots 0 x f f$ | BYTE | $\begin{aligned} & \mathrm{On} \\ & 0 \times 01 \end{aligned}$ | 0 | 1 | 0x0Y04 |
| Behaviour of outputs at com-munication errors | Off <br> Last value <br> Substitute value | $\begin{aligned} & \hline 0 \\ & 1+(n * 5) \\ & 2+(n * 5) \\ & n \leq 2 \end{aligned}$ | BYTE | $\begin{aligned} & \hline \text { Off } \\ & 0 \times 00 \end{aligned}$ | 0 | 2 | 0x0Y05 |


| Name | Value | Internal <br> value | Internal <br> value, <br> type | Default | Min. | Max. | Max. |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Substitute <br> value at <br> outputs <br> Bit $7=$ <br> Output 7 <br> Bit 0 <br> Output 0 | $0 \ldots 255$ | $0 \ldots 0 \times f f$ | BYTE | $0 \times 00$ | 0 | 255 | $0 x 0 Y 06$ |
| Reserve | $0 \ldots 255$ | $0 \ldots 0 x f$ | BYTE | $0 \times 00$ | 0 | 255 | $0 x 0 Y 07$ |
| Reserve | $0 \ldots 255$ | $0 \ldots 0 x f$ | BYTE | $0 x 00$ | 0 | 255 | $0 x 0 Y 08$ |
| 1) With CS31 and addresses smaller than 70 and FBP, the value is increased by 1 <br> 2) Not with FBP |  |  |  |  |  |  |  |

GSD file:

| Ext_User_Prm_Data_Len $=$ | 10 |
| :--- | :--- |
| Ext_User_Prm_Data_Const $(0)=$ | $0 \times 04,0 \times 51,0 \times 00,0 \times 06,0 \times 01,0 \times 01,0 \times 00$, <br> $0 \times 00,0 \times 00,0 \times 00$ |

State LEDs
During the power ON procedure, the module initializes automatically. All LEDs (except the channel LEDs) are ON during this time.

| LED |  | State | Color | LED = OFF | LED = ON | LED flashes |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| AH3 ${ }^{\text {D }}$ | Outputs O0 ... O7 | Digital output | Yellow | Output = OFF | $\begin{aligned} & \text { Output = ON } \\ & { }^{2} \text { ) } \end{aligned}$ | -- |
|  | UP | Process supply voltage 24 V DC via terminal | Green | Process supply voltage is missing | Process supply voltage OK | -- |
|  | UP3 | Process supply voltage outputs 0 ... 3 | Green | Process supply voltage is missing | Process supply voltage OK | -- |
| $]^{\text {UP } 2 \mathrm{NV}=400 \mathrm{~W}} \text { Outan 28V }=2 \mathrm{~m}$ |  | 24 V DC via terminal |  |  |  |  |
|  | UP4 | Process supply voltage outputs 4 ... 7 24 V DC via terminal | Green | Process supply voltage is missing | Process supply voltage OK | -- |



## Technical data

## Technical data of the module

The system data of AC500 and S500 are applicable to the standard version ${ }^{\leftrightarrows}$ Chapter 5.1.2 "System data AC500" on page 166.

The system data of AC500-XC are applicable to the XC version ${ }^{〔}$ Chapter 5.1.3 "System data AC500-XC" on page 169.

Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter | Value |
| :---: | :---: |
| Process supply voltage UP, UP3 and UP4 |  |
| Connections | Terminals 1.8 and 2.8 for +24 V (UP) as well as 1.9 and $2.90 \mathrm{~V}(\mathrm{ZP})$ <br> Terminals 3.8 for +24 V (UP3) as well as 3.9 for 0 V (ZP3) <br> Terminals 4.8 for +24 V (UP4) as well as 4.9 for 0 V (ZP4) |
| Rated value | 24 V DC |
| Max. ripple | 5 \% |
| Protection against reversed voltage | Yes |
| Rated protection fuse on UP, UP3 and UP4 | 10 A fast (for each process supply voltage) |
| Galvanic isolation | Yes, per module and per output channel groups |
| Current consumption |  |
| From 24 V DC power supply at the terminals UP/L+ and ZP/M of the CPU/communication interface module | Ca. 2 mA |
| From UP at normal operation / with outputs | Ca. $20 \mathrm{~mA}+1.5 \mathrm{~mA}$ per output |
| From UP3 or UP4 at normal operation / with outputs | Ca. 0.01 A + max. 2 A per output |
| Inrush current from UP (at power up) | $0.015 \mathrm{~A}^{2} \mathrm{~s}$ |


| Parameter |  |
| :--- | :--- |
| Inrush current from UP3 or UP4 (at <br> power up) | $0.005 \mathrm{~A}^{2}$ s (without output load) |
| Max. power dissipation within the module | 6 W |
| Weight (without terminal unit) | Ca. 135 g |
| Mounting position | Horizontal <br> Or vertical with derating (output load reduced to <br> $50 \%$ at $+40{ }^{\circ} \mathrm{C}$ per group) |
| Cooling | The natural convection cooling must not be hin- <br> dered by cable ducts or other parts in the con- <br> trol cabinet. |

No effects of multiple overloads

## NOTICE!

## Attention:

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply and continuous overvoltage up to 30 V DC.

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an external fuse.

## Technical data of the digital outputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 8 outputs (with transistors, non-latching type) |
| Distribution of the channels into groups | 2 groups of 4 channels |
| Connection of the channels |  |
| O0 ... O 3 | Terminals 3.0, 3.1, 3.4, 3.5 |
| O4 ... 07 | Terminals 4.0, 4.1, 4.4, 4.5 |
| Indication of the output signals | 1 yellow LED per channel, the LED is ON if the output signal is high (signal 1) |
| Power supply voltage for the module | Terminals 1.8 and 2.8 (positive pole of the process supply voltage, signal name UP) |
| Reference potential for module power supply | Terminals 1.9 and 2.9 (negative pole of the process supply voltage, signal name ZP) |
| Power supply voltage for the outputs O 0 to O3 | Terminal 3.8 (positive pole of the process supply voltage, signal name UP3) |
| Reference potential for the outputs O 0 to O 3 | Terminal 3.9 (negative pole of the process supply voltage, signal name ZP3) |
| Power supply voltage for the outputs O 4 to 07 | Terminal 4.8 (positive pole of the process supply voltage, signal name UP4) |
| Reference potential for the outputs O 4 to O 7 | Terminal 4.9 (negative pole of the process supply voltage, signal name ZP4) |
| Output voltage for signal 1 | UP (-0.4 V) |
| Output delay ( $0->1$ or 1->0) | On request |
| Output current |  |
| Rated value, per channel | 2 A at UP3 or UP4 $=24 \mathrm{~V}$ |


| Parameter |  | Value |
| :--- | :--- | :--- |
| Maximum value (channels O0 ... O3) | 8 A |  |
|  | Maximum value (channels O4 ... O7) | 8 A |
| Leakage current with signal 0 | $<0.1 \mathrm{~mA}$ |  |
| Rated protection fuse on UP | 10 A fast |  |
| Demagnetization when inductive loads are <br> switched off | With clamp diode in output high side driver |  |
| Switching frequency |  |  |
|  | With resistive load | On request |
|  | With inductive loads | Max. 2 Hz |
| With lamp loads |  | No (should be done externally) |
| Short-circuit proof / overload proof | No |  |
| Overload message | No (should be done externally) |  |
| Output current limitation | Yes to UP3 or UP4. No to outputs in same <br> group. |  |
| Resistance to feedback against 24 V signals |  |  |
| Max. cable length | 1000 m |  |
|  | Shielded | 600 m |
|  | Unshielded |  |

## Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm

The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 240 800 R0001 | DO526, digital output module, 8 DO, <br> 24 V DC / 2 A, 1-wire | Active |
| 1SAP 440 800 R0001 | DO526-XC, digital output module, <br> 8 DO, 24 V DC / 2 A, 1-wire, <br> XC version | Active |
| 1SAP 213 200 R0001 | TU542, I/O terminal unit, 24 V DC, <br> spring terminals | Active |
| 1SAP 413 200 R0001 | TU542-XC, I/O terminal unit, 24 V DC, <br> spring terminals, XC version | Active |

## DX522 - Digital input/output module

## Features

- 8 digital inputs 24 V DC, module-wise galvanically isolated
- 8 relay outputs
- Fast counter
- XC version for use in extreme ambient conditions available


1 I/O bus
2 Allocation between terminal number and signal name
38 yellow LEDs to display the signal states at the digital inputs ( 10 ... 17)
48 yellow LEDs to display the signal states at the digital relay outputs (R0 ... R7)
1 green LED to display the state of the process supply voltage UP
2 red LEDs to display errors
Label
Terminal unit
DIN rail
Sign for XC version

Intended purpose
The device can be used as a decentralized I/O extension module for S500 communication interface modules (e. g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs.

Digital configurable input/output unit.

- 8 digital inputs 24 V DC in 1 group (1.0...1.7)
- 8 digital relay outputs with one change-over contact each (R0...R7). All output channels are galvanically isolated from each other.
- Fast counter

The configuration is performed by software. The modules are supplied with a process supply voltage of 24 V DC.
All available inputs/outputs are galvanically isolated from all other circuitry of the module. There is no potential separation between the channels within the same group.

For use in extreme ambient conditions (e.g. wider temperature and humidity range), a special $X C$ version of the device is available.

## Functionality

| Parameter | Value |
| :--- | :--- |
| Fast counter | Integrated, many configurable operating modes (only with <br> AC500) |
| LED displays | For signal states, errors and supply voltage |
| Internal power supply | Through the I/O bus interface (I/O bus) |
| External power supply | Via the terminals ZP and UP (process supply voltage <br> $24 \mathrm{~V} \mathrm{DC)}$ |
| Required terminal units | TU531 or TU532 \& Chapter 5.2.5.3 "TU531 and TU532 for <br> I/O modules" on page 945 |

The device is plugged on a terminal unit $\Rightarrow$ Chapter 5.2.5.3 "TU531 and TU532 for I/O modules" on page 945 . Position the module properly and press until it locks in place. The terminal unit is either mounted on a DIN rail or to the wall using 2 screws plus the additional accessory for wall mounting ${ }^{\text {² }}$, Chapter 5.2.8.2.5 "TA526 - Wall mounting accessory" on page 1324.

## Connections

## WARNING!

## Risk of death by electric shock!

Hazardous voltages can be present at the terminals of the module.
Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

The connection of the I/O channels is carried out using the 40 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

The terminals $1.8 \ldots 4.8$ and $1.9 \ldots 4.9$ are electrically interconnected within the I/O terminal unit and have always the same assignment, irrespective of the inserted module:

- Terminals 1.8 ... 4.8: process supply voltage UP $=+24 \mathrm{~V}$ DC
- Terminals $1.9 \ldots 4.9$ : process supply voltage $\mathrm{ZP}=0 \mathrm{~V}$ DC

Table 137: Assignment of the other terminals:

| Terminals | Signal | Description |
| :--- | :--- | :--- |
| $1.0 \ldots 1.7$ | IO $\ldots$ I7 | Input signals of the 8 digital <br> inputs |
| $1.8 \ldots 4.8$ | UP | Process supply voltage +24 V <br> DC |
| $1.9 \ldots 4.9$ | RO | Reference potential for the 8 <br> digital inputs and the process <br> supply voltage |
| 2.0 | NO 0 | Common contact of the first <br> relay output |
| 3.0 | NC 0 | Normally-open contact of the <br> first relay output |
| 4.0 | NO 1 | Normally-closed contact of the <br> first relay output |
| 2.1 | NC 1 | Common contact of the <br> second relay output |
| 3.1 | : | Normally-open contact of the <br> second relay output |
| 4.1 | R7 | Normally-closed contact of the <br> second relay output |
| $:$ | NO 7 | NC 7 |
| 2.7 | Common contact of the eighth <br> relay output |  |
| 3.7 | Normally-open contact of the <br> eighth relay output |  |

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a communication interface module or a CPU). Thus, the current consumption from 24 V DC power supply at the terminals L+/UP and M/ZP of the CPU/communication interface module increases by 2 mA per DX522.

The external power supply connection is carried out via the UP (+24 V DC) and the ZP (0 V DC) terminals.

## WARNING!

## Removal/Insertion under power

Removal or insertion under power is permissible only if all conditions for hot swapping are fullfilled.

② Chapter 4.9.3 "Replace an I/O module with hot swap" on page 153
The devices are not designed for removal or insertion under power when the conditions for hot swap do not apply. Because of unforeseeable consequences, it is not allowed to plug in or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while they are energized in a hazardous location could result in an electric arc, which could create an ignition source resulting in fire or explosion.

Prior to proceeding, make sure that power is been disconnected and that the area has been thoroughly checked to ensure that flammable materials are not present.

The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

## Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

The module provides several diagnosis functions.


Fig. 79: Connection of the digital input/output module DX522
I/O bus
2 Control cabinet earth

## NOTICE!

- If the relay outputs have to switch inductive DC loads, free-wheeling diodes must be circuited in parallel to these loads.
- If the relay outputs have to switch inductive AC loads, spark suppressors are required.


## CAUTION!

The process supply voltage must be included in the grounding concept (e. g. grounding of the negative terminal).

## NOTICE!

## Risk of damaging the PLC module!

The following has to be considered when connecting input and output voltages to the module:

- All 230 V AC feeds must be single-phase from the same supply system.
- Connection of 2 or more relay contacts in series is possible; however, voltages above 230 V AC and 3-phase loads are not allowed.
- The 8 change-over contacts of the relays are galvanically isolated from channel to channel. This allows to connect loads of 24 V DC and 120/230 V AC to relay outputs of the same module. In such cases it is necessary that both supply voltages are grounded to prevent unsafe floating grounds.


## NOTICE!

## Risk of damaging the PLC module!

There is no internal short-circuit or overload protection for the relay outputs.
Protect the relay contacts by back-up fuses of 6 A max. (characteristic gG/gL). Depending on the application, fuses can be used for single channels or modulewise.

## Internal data exchange

|  | Without the fast counter | With the fast counter (only <br> with AC500) |
| :--- | :--- | :--- |
| Digital inputs (bytes) | 1 | 3 |
| Digital outputs (bytes) | 1 | 3 |
| Counter input data (words) | 0 | 4 |
| Counter output data (words) | 0 | 8 |

## I/O configuration

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or communication interface module) during power-up of the system.

Hence, replacing I/O modules is possible without any re-parameterization via software.


## Parameterization

| Firmware version | Configuration |
| :--- | :--- |
| Firmware version > V2.0.0 | The arrangement of the parameter data is per- <br> formed by Control Builder Plus/ Automation <br> Builder software. |

The parameter data directly influences the functionality of modules.
For non-standard applications, it is necessary to adapt the parameters to your system configuration.

Module: Module slot address: $\mathrm{Y}=1$... 10

| Name | Value | Internal value | Internal value, type | Default | Min. | Max. | EDS Slot/ Index |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Module ID | Internal | $\begin{aligned} & 1210 \\ & \left.{ }^{1}\right) \end{aligned}$ | Word | $\begin{aligned} & \hline 1210 \\ & 0 x 04 \mathrm{BA} \end{aligned}$ | 0 | 65535 | 0x0Y01 |
| Ignore module ${ }^{2}$ ) | No Yes | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | Byte | $\begin{aligned} & \mathrm{No} \\ & \mathrm{Nox} 00 \end{aligned}$ |  |  | Not for FBP |
| Parameter length | Internal | 5 | Byte | $\begin{aligned} & \text { 5-CPU } \\ & 4-\mathrm{FBP} \end{aligned}$ | 0 | 255 | 0x0Y02 |
| Check supply | $\begin{aligned} & \text { Off } \\ & \text { On } \end{aligned}$ | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | Byte | $\begin{aligned} & \mathrm{On} \\ & 0 \times 01 \end{aligned}$ | 0 | 1 | 0x0Y03 |
| Input delay | 0.1 ms <br> 1 ms <br> 8 ms <br> 32 ms | $\begin{aligned} & \hline 0 \\ & 1 \\ & 2 \\ & 3 \end{aligned}$ | Byte | $\begin{aligned} & 8 \mathrm{~ms} \\ & 0 \times 02 \end{aligned}$ | 0 | 3 | 0x0Y04 |
| Fast Counter ${ }^{4}$ ) | 0 <br> 10 <br> ${ }^{3}$ ) | $\begin{aligned} & 0 \\ & : \\ & 10 \end{aligned}$ | Byte | $\begin{aligned} & \text { Mode } 0 \\ & 0 \times 00 \end{aligned}$ |  |  | Not for FBP |
| Behaviour of outputs at com-munication errors | Off <br> Last value <br> Substitute value | $\begin{aligned} & 0 \\ & 1+\left(n^{*} 5\right) \\ & 2+\left(n^{*} 5\right), \\ & n \leq 2 \end{aligned}$ | Byte | $\begin{array}{\|l\|} \hline \text { Off } \\ 0 \times 00 \end{array}$ | 0 | 2 | 0x0Y05 |
| Substitute value at outputs) Bit $7=$ Output 7 <br> Bit $0=$ Output 0 | $\begin{aligned} & \hline 0 \ldots \\ & 255 \end{aligned}$ | $\begin{aligned} & 0 \ldots \\ & 0 x f f \end{aligned}$ | Byte | $\begin{aligned} & \hline 0 \\ & 0 \times 00 \end{aligned}$ | 0 | 255 | 0x0Y06 |

## Remarks:

| $\left.{ }^{1}\right)$ | With CS31 and addresses smaller than 70 and FBP, the value is increased <br> by 1 |
| :--- | :--- |
| $\left.{ }^{2}\right)$ | Not with FBP |
| $\left.{ }^{3}\right)$ | For a description of the counter operating modes, please refer to the 'Fast <br> Counter' section $\star$ Chapter 5.2.4.3.2.9 "Fast counter" on page 606 |
| $\left.{ }^{4}\right)$ | With FBP and without the parameter Fast Counter |

GSD file:

| Ext_User_Prm_Data_Len $=$ | 7 |
| :--- | :--- |
| Ext_User_Prm_Data_Const | $0 \times 04,0 \times b b, 0 \times 04$, । |
| $(0)=$ | $0 \times 01,0 \times 02,0 \times 00,0 \times 00 ;$ |

## State LEDs

During the power ON procedure, the module initializes automatically. All LEDs (except the channel LEDs) are ON during this time.


## Technical data

## Technical data of the module

The system data of AC500 and S500 are applicable to the standard version $\Leftrightarrow$ Chapter 5.1.2 "System data AC500" on page 166.
The system data of AC500-XC are applicable to the XC version $\Leftrightarrow$ Chapter 5.1.3 "System data AC500-XC" on page 169.
Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter |  | Value |
| :--- | :--- | :--- |
| Process supply voltage UP |  |  |
|  | Connections | Terminals $1.8,2.8,3.8$ and 4.8 for +24 V (UP) <br> as well as $1.9,2.9,3.9$ and 4.9 for 0 V (ZP) |
|  | Rated value | 24 V DC |
|  | Max. ripple | $5 \%$ |
|  | Protection against reversed voltage | Yes |
|  | Rated protection fuse on UP | 10 A fast |
|  | Galvanic isolation | Yes, per module |
| Current consumption | From 24 V DC power supply at the <br> terminals UP/L+ and ZP/M of the CPU/ <br> communication interface module | ca. 2 mA |
|  | From UP at normal operation / with out- <br> puts | $0.05 \mathrm{~A}+$ output loads |
|  | Inrush current from UP (at power up) | 0.010 A² |
| Max. power dissipation within the module | 6 W (outputs OFF) |  |
| Weight (without terminal unit) | ca. 300 g |  |
| Mounting position | Horizontal or vertical with derating (output <br> load reduced to $50 \%$ at $+40{ }^{\circ} \mathrm{C}$ per group) |  |
| Cooling | The natural convection cooling must not be <br> hindered by cable ducts or other parts in the <br> control cabinet. |  |

No effects of multiple overloads

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an external fuse.

## Technical data of the digital inputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 8 |
| Distribution of the channels into groups | 1 group of 8 channels |
| Terminals of the channels I0 ... I7 | $1.0 \ldots 1.7$ |
| Reference potential for all inputs | Terminals $1.9,2.9,3.9$ and 4.9 (negative pole of <br> the process supply voltage, signal name ZP) |
| Galvanic isolation | From the rest of the module (I/O bus) |
| Indication of the input signals | One yellow LED per channel, the LED is ON <br> when the input signal is high (signal 1) |
| Monitoring point of input indicator | LED is part of the input circuitry |
| Input type acc. to EN 61131-2 | Type 1 |
| Input delay (0->1 or 1->0) | Typ. 8 ms, configurable $0.1 \ldots 32 \mathrm{~ms}$ |


| Parameter | Value |
| :--- | :--- |
| Input signal voltage | 24 V DC |
| Signal 0 | $-3 \mathrm{~V} \ldots+5 \mathrm{~V}$ |
| Undefined signal | $>+5 \mathrm{~V} \ldots<+15 \mathrm{~V}$ |
| Signal 1 | $+15 \mathrm{~V} \ldots+30 \mathrm{~V}$ |
| Ripple with signal 0 | Within $-3 \mathrm{~V} \ldots+5 \mathrm{~V}$ |
| Ripple with signal 1 | Within $+15 \mathrm{~V} . .+30 \mathrm{~V}$ |
| Input current per channel |  |
|  | Input voltage +24 V |
|  | Input voltage +5 V |
| Input voltage +15 V | $>1 \mathrm{~mA}$ |
|  | Input voltage +30 V |
| Max. cable length | $>5 \mathrm{~mA}$ |
|  | Shielded |
|  | Unshielded |

## Technical data of the relay outputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 8 relay outputs |
| Distribution of channels into groups | 8 groups of 1 channel each |
| Connection of the channel R0 | Terminal 2.0 (common), 3.0 (NO) and 4.0 (NC) |
| Connection of the channel R1 | Terminal 2.1 (common), 3.1 (NO) and 4.1 (NC) |
| Connection of the channel R6 | Terminal 2.6 (common), 3.6 (NO) and 4.6 (NC) |
| Connection of the channel R7 | Terminal 2.7 (common), 3.7 (NO) and 4.7 (NC) |
| Galvanic isolation | Between the channels and from the rest of the module |
| Indication of the output signals | One yellow LED per channel, the LED is ON when the <br> relay coil is energized |
| Monitoring point of output indicator | LED is controlled by process CPU |
| Way of operation | Non-latching type |
| Output delay (0->1 or 1->0) | On request |
| Relay power supply | By UP process supply voltage |
| Relay outputs |  |
|  | Output short circuit protection |
| Rated protection fuse | Should be provided externally with a fuse or circuit <br> breaker |
| Min. switching current | $6 \mathrm{~A} \mathrm{gL/gG} \mathrm{per} \mathrm{channel}$ |
| Output switching capacity | 10 mA |
|  | Resistive load, max. |
| Inductive load, max. | $3 \mathrm{~A} ; 3 \mathrm{~A} \mathrm{(230} \mathrm{~V} \mathrm{AC)} ,2 \mathrm{~A} \mathrm{(24} \mathrm{~V} \mathrm{DC)}$ |
|  | Lamp load |


| Parameter | Value |
| :--- | :--- |
| Output switching capacity (XC ver- <br> sion above $\left.+60^{\circ} \mathrm{C}\right)$ | On request |
| Lifetime (cycles) | Mechanical: $300000 ;$ <br> Under load: $300000(24 \mathrm{~V} \mathrm{DC} \mathrm{at} 2 \mathrm{~A}), 200000(120 \mathrm{~V}$ <br> AC at 2 A), $100000(230 \mathrm{~V} \mathrm{AC} \mathrm{at} \mathrm{3} \mathrm{A)}$ |
| Spark suppression with inductive AC <br> load | Must be performed externally according to driven load <br> specifications |
| Demagnetization with inductive DC <br> load | A free-wheeling diode must be circuited in parallel to <br> the inductive load |
| Switching frequency |  |
|  | With resistive load |
|  | With inductive load |
| With lamp load | Max. 10 Hz |
| Max. cable length | On request |
|  | Shielded |
|  | Unshielded |

## Technical data of the fast counter

The fast counter of the module does not work if the module is connected to a

- FBP interface module
- CS31 bus module
- CANopen communication interface module

| Parameter | Value |
| :--- | :--- |
| Used inputs | I0 / I1 |
| Used outputs | None |
| Counting frequency | 50 kHz max. |

Chapter 6.8.2.12 "Fast counters in AC500 devices" on page 4536

## Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm

The dimensions are in mm and in brackets in inch.

Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 245 200 R0001 | DX522, digital input/output module, <br> 8 DI, 24 V DC, 8 DO relays | Active |
| 1SAP 445 200 R0001 | DX522-XC, digital input/output <br> module, 8 DI, 24 V DC, 8 DO relays, <br> XC version | Active |

*) Modules in lifecycle Classic are available from stock but not recommended
for planning and commissioning of new installations.

## DX531 - Digital input/output module

## Features

- 8 digital inputs $120 / 230$ V AC
- 4 relay outputs with one change-over contacts each
- Module-wise galvanically isolated


1 I/O bus
2 Allocation between terminal number and signal name
38 yellow LEDs to display the signal states at the digital inputs (IO ... I7)
44 yellow LEDs to display the signal states at the digital relay outputs (R0 ... R3)
51 green LED to display the state of the process supply voltage UP
62 red LEDs to display errors
7 Label
8 Terminal unit
9 DIN rail
$\stackrel{{ }_{c}^{*}+\underset{k}{*}}{\boldsymbol{c}_{k}}$ Sign for XC version

## Intended purpose

The device can be used as a decentralized I/O extension module for S500 communication interface modules (e. g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs.

Digital configurable input / output unit.

- 8 digital inputs $120 / 230 \mathrm{~V}$ AC in 1 group (2.0 ... 2.3 and $3.0 \ldots 3.3$ )
- 4 digital relay outputs with one change-over contact each (R0 ... R3). All output channels are galvanically isolated from each other.

The configuration is performed by software. The modules are supplied with a process supply voltage of 24 V DC.

All available inputs/outputs are galvanically isolated from all other circuitry of the module. There is no potential separation between the channels within the same group.
For use in extreme ambient conditions (e.g. wider temperature and humidity range), a special $X C$ version of the device is available.

## Functionality

| Parameter | Value |
| :--- | :--- |
| LED displays | For signal states, errors and supply voltage |
| Internal power supply | Through the I/O bus interface (I/O bus) |
| External power supply | Via the terminals ZP and UP (process supply voltage <br> 24 V DC) |
| Required terminal units | TU531 or TU532 4. Chapter 5.2.5.3 "TU531 and <br> TU532 for I/O modules" on page 945 |

The device is plugged on a terminal unit ${ }^{\wedge} \triangleq$ Chapter 5.2.5.3 "TU531 and TU532 for I/O modules" on page 945 . Position the module properly and press until it locks in place. The terminal unit is either mounted on a DIN rail or to the wall using 2 screws plus the additional accessory for wall mounting ${ }^{\wedge}>$ Chapter 5.2.8.2.5 "TA526 - Wall mounting accessory" on page 1324.

## Connections

## WARNING!

## Risk of death by electric shock!

Hazardous voltages can be present at the terminals of the module.
Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

The connection of the I/O channels is carried out using the 40 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.
The terminals $1.8 \ldots 4.8$ and $1.9 \ldots 4.9$ are electrically interconnected within the I/O terminal unit and always have the same assignment, irrespective of the inserted module:

- Terminals $1.8 \ldots$.. 4.8: process supply voltage UP $=+24 \mathrm{~V}$ DC
- Terminals $1.9 \ldots 4.9$ : process supply voltage $\mathrm{ZP}=0 \mathrm{~V} \mathrm{DC}$

Table 138: Assignment of the other terminals

| Terminals | Signal | Description |
| :---: | :---: | :---: |
| 1.0 ... 1.7 | unused |  |
| 2.0 and 3.0 | 10 and I1 | Input signals for the digital inputs IO and I1 |
| 4.0 | N01 | Neutral conductor for the digital inputs I0 and I1 |
| 2.1 and 3.1 | 12 and I3 | Input signals for the digital inputs I2 and I3 |
| 4.1 | N23 | Neutral conductor for the digital inputs I2 and I3 |
| 2.2 and 3.2 | 14 and 15 | Input signals for the digital inputs 14 and I5 |
| 4.2 | N45 | Neutral conductor for the digital inputs 14 and I5 |
| 2.3 and 3.3 | 16 and 17 | Input signals for the digital inputs I6 and I7 |
| 4.3 | N67 | Neutral conductor for the digital inputs 16 and I7 |
| 2.4 | R0 | Common contact of the first relay output |
| 3.4 and 4.4 | NOO and NCO | NO and NC contacts of the first relay output |
| 2.5 | R1 | Common contact of the second relay output |
| 3.5 and 4.5 | NO1 and NC1 | NO and NC contacts of the second relay output |
| 2.6 | R2 | Common contact of the third relay output |
| 3.6 and 4.6 | NO2 and NC2 | NO and NC contacts of the third relay output |
| 2.7 | R3 | Common contact of the fourth relay output |
| 3.7 and 4.7 | NO3 and NC3 | NO and NC contacts of the fourth relay output |

## Digital inputs


2.316
$3.3 \quad 17$
4.3 N67

Digital outputs



$3.7 \mathrm{NO}_{2-7}^{4.7}$

Fig. 80: Internal construction
The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a communication interface module or a CPU). Thus, the current consumption from 24 V DC power supply at the terminals $L+/ U P$ and M/ZP of the CPU/communication interface module increases by 2 mA per $\mathrm{DX531}$. The external power supply connection is carried out via the UP (+24 V DC) and the ZP (0 V DC) terminals.

## WARNING!

## Removal/Insertion under power

Removal or insertion under power is permissible only if all conditions for hot swapping are fullfilled.
« Chapter 4.9.3 "Replace an I/O module with hot swap" on page 153
The devices are not designed for removal or insertion under power when the conditions for hot swap do not apply. Because of unforeseeable consequences, it is not allowed to plug in or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while they are energized in a hazardous location could result in an electric arc, which could create an ignition source resulting in fire or explosion.

Prior to proceeding, make sure that power is been disconnected and that the area has been thoroughly checked to ensure that flammable materials are not present.

The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

## Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.


Fig. 81: Connection of the module
I/O bus
2 Control cabinet earth

## NOTICE!

- If the relay outputs have to switch inductive DC loads, free-wheeling diodes must be circuited in parallel to these loads.
- If the relay outputs have to switch inductive AC loads, spark suppressors are required.


## CAUTION!

The process supply voltage must be included in the grounding concept (e. g. grounding of the negative terminal).

## NOTICE!

## Risk of damaging the PLC module!

The following has to be considered when connecting input and output voltages to the module:

- All 230 V AC feeds must be single phase from the same supply system.
- Connection of 2 or more relay contacts in series is possible; however, voltages above 230 VAC and 3 -phase loads are not allowed.
- The 4 change-over contacts of the relays are galvanically isolated from channel to channel. This allows to connect loads of 24 V DC and $120 / 230 \mathrm{~V}$ AC to relay outputs of the same module. In such cases it is necessary that both supply voltages are grounded to prevent unsafe floating grounds.
- All input signals must come from the same phase of the same supply system (together with the used neutral conductor). The module is designed for 120/230 V AC max., not for 400 V AC , not even between two input terminals.
- All neutral conductor connections must be common to the same supply system, since the terminals $4.0 \ldots 4.3$ are interconnected within the module. Otherwise, accidental energization could occur.


## NOTICE!

## Risk of damaging the PLC module!

There is no internal short-circuit or overload protection for the relay outputs.
Protect the relay contacts by back-up fuses of 6 A max. (characteristic $\mathrm{gG} / \mathrm{gL}$ ). Depending on the application, fuses can be used for single channels or modulewise.

The module provides several diagnosis functions.

## Internal data exchange

| Digital inputs (bytes) | 1 |
| :--- | :--- |
| Digital outputs (bytes) | 1 |
| Counter input data (words) | 0 |
| Counter output data (words) | 0 |

## I/O configuration

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or communication interface module) during power-up of the system.
Hence, replacing I/O modules is possible without any re-parameterization via software.


If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.

## Parameterization

| Firmware version | Configuration |
| :--- | :--- |
| Firmware version > V2.0.0 | The arrangement of the parameter data is per- <br> formed by Control Builder Plus/ Automation <br> Builder software. |

The parameter data directly influences the functionality of modules.
For non-standard applications, it is necessary to adapt the parameters to your system configuration.

Module: Module slot address: $\mathrm{Y}=1$... 10

| Name | Value | Internal value | Internal value, type | Default | Min. | Max. | EDS <br> Slot/ Index |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Module ID | Internal | $\begin{aligned} & \hline 1205 \\ & \left.{ }^{1}\right) \end{aligned}$ | Word | $\begin{aligned} & \hline 1205 \\ & 0 \times 04 \mathrm{~B} 5 \end{aligned}$ | 0 | 65535 | 0x0Y01 |
| Ignore module ${ }^{2}$ ) | $\begin{aligned} & \text { No } \\ & \text { Yes } \end{aligned}$ | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | Byte | $\begin{array}{\|l} \hline \text { No } \\ 0 \times 00 \end{array}$ |  |  | not for FBP |
| Parameter length | Internal | 4 | Byte | $\begin{aligned} & \text { 4-CPU } \\ & 4-\mathrm{FBP} \end{aligned}$ | 0 | 255 | 0x0Y02 |
| Check supply | Off on | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | Byte | $\begin{array}{\|l\|} \hline \text { On } \\ 0 \times 01 \end{array}$ | 0 | 1 | 0x0Y03 |
| Input delay | $\begin{aligned} & 20 \mathrm{~ms} \\ & 100 \mathrm{~ms} \end{aligned}$ | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | Byte | $\begin{aligned} & 20 \mathrm{~ms} \\ & 0 \times 00 \end{aligned}$ | 0 | 1 | 0x0Y04 |
| Behaviour of outputs at com-munication errors | Off <br> Last value Substitute value | $\begin{aligned} & 0 \\ & 1+(n * 5) \\ & 2+\left(n^{*} 5\right), \\ & n \leq 2 \end{aligned}$ | Byte | $\begin{array}{\|l\|} \hline \text { Off } \\ 0 \times 00 \end{array}$ | 0 | 2 | 0x0Y05 |
| Substitute value at outputs <br> Bit 3 = Output 3 <br> Bit $0=$ Output 0 | $0 \ldots 15$ | $\begin{aligned} & 0 \ldots \\ & 0 \times 0 f \end{aligned}$ | Byte | $\begin{array}{\|l\|} \hline 0 \\ 0 \times 00 \end{array}$ | 0 | 15 | 0x0Y06 |
| ${ }^{1}$ ) With CS31 and addresses smaller than 70 and FBP, the value is increased by 1 <br> ${ }^{2}$ ) Not with FBP |  |  |  |  |  |  |  |

GSD file:

| Ext_User_Prm_Data_Len $=$ | 7 |
| :--- | :--- |
| Ext_User_Prm_Data_Const | $0 \times 04,0 \times b 6,0 \times 04,1$ |
| $(0)=$ | $0 \times 01,0 \times 00,0 \times 00,0 \times 00 ;$ |

## State LEDs

During the power ON procedure, the module initializes automatically. All LEDs (except the channel LEDs) are ON during this time.

| LED |  | State | Color | LED = OFF | LED = ON | LED flashes |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| A\#B DX531 | Inputs IO ... I7 | Digital input | Yellow | Input = OFF | Input $=\mathrm{ON}$ | -- |
|  | Outputs <br> R0 ... R3 <br> (relays) | Digital output | Yellow | Relay output = OFF | Relay output = ON | -- |
|  | UP | Process supply voltage 24 V DC via terminal | Green | Process supply voltage is missing | Process supply voltage OK | -- |
| 801400.R |  |  |  |  |  |  |
| Relay 230 V 3 A | $\begin{aligned} & \text { CH-ERR2 } \\ & \hline \mathrm{CH} \text {-ERR3 } \end{aligned}$ | Channel error, error messages in groups (digital inputs/ outputs combined into the groups 2 and 3) | Red | No error or process supply voltage is missing | Severe error within the corresponding group | Error on one channel of the corresponding group |
|  |  |  | Red |  |  |  |
|  | CH-ERR *) | Module Error | Red | -- | Internal error | -- |
|  | *) All of the LEDs $\mathrm{CH}-\mathrm{ERR} 2$ to $\mathrm{CH}-E R R 3$ light up together |  |  |  |  |  |

## Technical data

## Technical data of the module

The system data of AC500 and S500 are applicable to the standard version $\Leftrightarrow$ Chapter 5.1.2 "System data AC500" on page 166.

The system data of AC500-XC are applicable to the XC version ${ }^{\aleph}$ Chapter 5.1.3 "System data AC500-XC" on page 169.
Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter |  | Value |
| :--- | :--- | :--- |
| Process supply voltage UP |  |  |
|  | Connections | Terminals 1.8, 2.8, 3.8 and 4.8 for <br> +24 V DC (UP) as well as 1.9, 2.9, <br> 3.9 and 4.9 for 0 V DC (ZP) |
|  | Rated value | 24 V DC |
|  | Max. ripple | $5 \%$ |
|  | Protection against reversed voltage | Yes |
|  | Rated protection fuse on UP | 10 A fast |
| Galvanic isolation | Yes, per module |  |
| Current consumption |  |  |


| Parameter |  | Value |
| :--- | :--- | :--- |
|  | From 24 V DC power supply at the terminals <br> UP/L+ and ZP/M of the CPU/communication inter- <br> face module | ca. 2 mA |
|  | From UP at normal operation / with outputs | $0.15 \mathrm{~A}+$ output loads |
| Inrush current from UP (at power up) | $0.004 \mathrm{~A}^{2} \mathrm{~s}$ |  |
| Max. power dissipation within the module | 6 W (outputs OFF) |  |
| Weight (without terminal unit) | Ca. 300 g |  |
| Mounting position | Horizontal or vertical with derating <br> (output load reduced to $50 \%$ at <br> $+40{ }^{\circ} \mathrm{C}$ per group) |  |
| Cooling | The natural convection cooling <br> must not be hindered by cable <br> ducts or other parts in the control <br> cabinet. |  |

No effects of multiple overloads

## NOTICE!

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and temporary overvoltage up to 30 V DC.

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an external fuse.

## Technical data of the digital inputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 8 |
| Distribution of the channels into groups | 4 groups of 2 channels each |
| Terminals of the channels 10 to I7 | ③ Chapter 5.2.4.3.2.8.4 "Connections" on page 595 |
| Galvanic isolation | 2500 V AC from the rest of the module (I/O bus) |
| Indication of the input signals | 1 yellow LED per channel <br> The LEDs are only operating if the module is initialized |
| Monitoring point of input indicator | LED is controlled by process CPU |
| Input type acc. to EN 61131-2 | Type 2 |
| Input delay (0->1 or 1->0) | Typ. 20 ms |
| Input signal voltage | 230 V AC or 120 V AC |
| Input signal range | 0 V AC ... 265 V AC |
| Input signal frequency | 47 Hz ... 63 Hz |
| Input characteristic | According EN 61132-2 Type 2 |
| Signal 0 | 0 V AC ... 40 V AC |
| Undefined signal | > 40 V AC ... $<74 \mathrm{~V}$ AC |
| Signal 1 | 74 V AC ... 265 V AC |


| Parameter |  | Value |
| :--- | :--- | :--- |
| Input current per channel |  |  |
|  | Input voltage $=159 \mathrm{~V} \mathrm{AC}$ | $>7 \mathrm{~mA}$ |
|  | Input voltage $=40 \mathrm{~V} \mathrm{AC}$ | $<5 \mathrm{~mA}$ |
| Overvoltage protection | Yes |  |
| Max. cable length |  |  |
|  | Shielded | 1000 m |
|  | Unshielded | 600 m |

## Technical data of the relay outputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 4 relay outputs |
| Distribution of channels into groups | 4 groups of 1 channel each |
| Connection of the four relays | ⓨ Chapter 5.2.4.3.2.8.4 "Connections" on page 595 |
| Galvanic isolation | Between the channels and from the rest of the module |
| Indication of the output signals | 1 yellow LED per channel, the LED is ON when the relay coil is energized |
| Monitoring point of output indicator | LED is controlled by process CPU |
| Way of operation | Non-latching type |
| Output delay (0->1 or 1->0) | On request |
| Relay power supply | By UP process supply voltage |
| Relay outputs |  |
| Output short circuit protection | Must be provided externally with a fuse or circuit breaker |
| Rated protection fuse | $6 \mathrm{~A} \mathrm{gL/gG} \mathrm{per} \mathrm{channel}$ |
| Output switching capacity |  |
| Resistive load, max. | $3 \mathrm{~A} ; 3 \mathrm{~A}(230 \mathrm{~V}$ AC), $2 \mathrm{~A}(24 \mathrm{~V}$ DC) |
| Inductive load, max. | 1.5 A; 1.5 A (230 V AC), 1.5 A (24 V DC) |
| Lamp load | 60 W (230 V AC), 10 W (24 V DC) |
| Lifetime (cycles) | Mechanical: 300 000; <br> Under load: 300000 (24 V DC at 2 A), 200000 ( 120 V AC at 2 A ), 100000 ( 230 V AC at 3 A ) |
| Spark suppression with inductive AC load | Must be performed externally according to driven load specifications |
| Demagnetization with inductive DC load | A free-wheeling diode must be circuited in parallel to the inductive load |
| Switching frequency |  |
| With resistive load | Max. 10 Hz |
| With inductive load | Max. 2 Hz |
| With lamp load | On request |
| Max. cable length |  |


| Parameter |  | Value |
| :--- | :--- | :--- |
|  | Shielded | 1000 m |
|  | Unshielded | 600 m |

## Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm

The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 245 000 R0001 | DX531, digital input/output module, <br> 8 DI, 120 / 230 V AC, 4 DO relays, <br> 2-wires | Active |

${ }^{*}$ ) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

## Fast counter

More information can be found in the Automation Builder chapter, "Fast counters in AC500 devices".
(2) Chapter 6.8.2.12 "Fast counters in AC500 devices" on page 4536

### 5.2.4.4 Analog I/O modules

### 5.2.4.4.1 S500-eCo

## AI561 - Analog input module

## Features

- 4 configurable analog inputs ( $10 \ldots$ I3) in 1 group
- Resolution: 12 bits including sign


1 I/O bus
21 green LED to display power supply, 1 red LED to display error
3 Terminal number
4 Allocation of signal name
5 Terminal block for input signals (9-pin)
6 Terminal block for input signals (11-pin)
72 holes for wall-mounting with screws
8 DIN rail

## Intended purpose

The device can be used as a decentralized I/O extension module for S 500 communication interface modules (e. g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs.

The inputs are not galvanically isolated from each other.
All other circuitry of the module is not galvanically isolated from the inputs or from the I/O bus.

The I/O module must not be used as a decentralized I/O module with CI590-CS31-HA communication interface modules.

## Functionality

4 analog inputs, individually configurable for

- Not used (default setting)
- $-2.5 \mathrm{~V} . . .+2.5 \mathrm{~V}$
- $-5 \mathrm{~V} \ldots+5 \mathrm{~V}$
- $0 \mathrm{~V} . .+5 \mathrm{~V}$
- $0 \mathrm{~V} \ldots+10 \mathrm{~V}$
- $0 \mathrm{~mA} . .220 \mathrm{~mA}$
- $4 \mathrm{~mA} . .20 \mathrm{~mA}$

| Parameter | Value |
| :---: | :---: |
| Resolution of the analog channels |  |
| Voltage bipolar (-2.5 V ... +2.5 V ; $-5 \mathrm{~V} \ldots+5$ V) | 12 bits including sign |
| Voltage unipolar ( $0 \mathrm{~V} \ldots 5 \mathrm{~V} ; 0 \mathrm{~V} \ldots 10 \mathrm{~V}$ ) | 12 bits |
| Current (0 mA ... 20 mA ; $4 \mathrm{~mA} . . .20 \mathrm{~mA}$ ) | 12 bits |
| LED displays | 2 LEDs for process voltage and error messages |
| Internal supply | Via I/O bus |
| External supply | Via the terminals L+ (process voltage 24 $V D C)$ and $M(0 \vee D C)$; the $M$ terminal is connected to the M terminal of the CPU via the I/O bus |

## Connections

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

The connection is carried out by using a removable 9-pin and 11-pin terminal block. These terminal blocks differ in their connection system (spring terminals or screw terminals, cable mounting from the front or from the side). The terminal blocks are not included in the module's scope of delivery and must be ordered separately.

| R0 | $\bigcirc$ |
| :---: | :---: |
| 10＋ | 2 － |
| 10－ | 3 － |
| R1 | 4 |
| 11＋ | 5 － |
| 11－ | 6 － |
| R2 | $7 \bigcirc$ |
| 12＋ | 8 |
| 12－ | 9 － |
| R3 | 10 － |
| 13＋ | 11 － |
| 13－ | 12 － |
| － | 13 。 |
| －－－ | 14 。 |
| －－－ | 15 。 |
| －－－ | 16 。 |
| －－－ | 17 。 |
| SG | 18 |
| L＋ | 19 － |
| M | 20 － |

Fig．82：Internal construction of the analog inputs
The assignment of the terminals：

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 1 | R0 | Burden resistor for input <br> signal 0 for current sensing |
| 2 | I0＋ | Positive pole of input signal 0 |
| 3 | I0－ | Negative pole of input signal 0 |
| 4 | R1 | Burden resistor for input <br> signal 1 for current sensing |
| 5 | I1＋ | Positive pole of input signal 1 |
| 6 | R2－ | Negative pole of input signal 1 |
| 7 | I2＋ | Burden resistor for input <br> signal 2 for current sensing |
| 8 | I2－ | Positive pole of input signal 2 |
| 9 | R3 | Negative pole of input signal 2 |
| 10 | Burden resistor for input <br> signal 3 for current sensing |  |


| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 11 | I3+ | Positive pole of input signal 3 |
| 12 | I3- | Negative pole of input signal 3 |
| 13 | --- | Reserved |
| 14 | --- | Reserved |
| 15 | --- | Reserved |
| 16 | --- | Reserved |
| 17 | --- | Reserved |
| 18 | SG | Shield grounding |
| 19 | L+ | Process voltage L+ (24 V DC) |
| 20 | M | Process voltage M (0 V DC) |

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a communication interface module or a CPU). Thus, the current consumption from 24 V DC power supply at the terminals L+/UP and M/ZP of the CPU/communication interface module increases by 10 mA per AI561.
The external power supply connection is carried out via the $\mathrm{L}+(+24 \mathrm{~V} \mathrm{DC}$ ) and the M ( 0 V DC) terminals. The $M$ terminal is interconnected to the M/ZP terminal of the CPU/communication interface module.

## NOTICE!

Risk of imprecise and faulty measurements!
Analog signals may be distorted seriously by external electromagnetic influences.

Use shielded wires when wiring analog signal sources. The cable shield must be grounded at both ends of the cable. Provide a potential equalisation of a low resistance to avoid high potential differences between different parts of the plant.

## NOTICE!

Risk of damaging the PLC modules!
The PLC modules must not be removed while the plant is connected to a power supply.
Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove or replace a module.


## NOTICE!

## Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

The module provides several diagnosis functions ${ }^{*} \Rightarrow$ Chapter 5.2.4.4.1.1.7 "Diagnosis" on page 615 .

The following figure is an example of the internal construction of the analog input AIO. The analog inputs Al1 ... Al3 are designed in the same way.


## CAUTION!

## Risk of damaging the analog input!

The $250 \Omega$ input resistor can be damaged by overcurrent.
Make sure that the current through the resistor never exceeds 30 mA .

Table 139: Example of the connection of analog sensors (voltage) to the input 10 of the analog input module AI561 (Proceed with the inputs I1 ... I3 in the same way)


24 VDC

| Connection of active-type analog sensors <br> (voltage) | Connection of passive-type analog sen- <br> sors (voltage) |
| :--- | :--- |
| $-2.5 \mathrm{~V} \ldots 2.5 \mathrm{~V}$ | $-2.5 \mathrm{~V} \ldots 2.5 \mathrm{~V}$ |
| $-5 \mathrm{~V} \ldots 5 \mathrm{~V}$ | $-5 \mathrm{~V} \ldots 5 \mathrm{~V}$ |
| $0 \mathrm{~V} \ldots 5 \mathrm{~V}$ | $0 \mathrm{~V} \ldots 5 \mathrm{~V}$ |
| $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ | $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ |

Table 140: Example of the connection of analog sensors to the Input IO of the analog input module AI561 ( Proceed with the inputs I1 ... I3 in the same way.)


| Connection of active-type analog sensors <br> (voltage) | Connection of passive-type analog sen- <br> sors (voltage) |
| :--- | :--- |
| $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ |
| $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ |  |

The meaning of the LEDs is described in the Displays section $\Rightarrow$ Chapter 5.2.4.4.1.1.8 "State LEDs" on page 616.

## I/O configuration

The analog input module AI561 does not store configuration data itself.

## Parameterization

The arrangement of the parameter data is performed with Automation Builder software.
The parameter data directly influences the functionality of modules.
For non-standard applications, it is necessary to adapt the parameters to your system configuration.

| Name | Value | Internal <br> Value | Internal <br> value, <br> Type | Default | Min. | Max. | EDS Slot <br> Index |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Module ID | Intern | $6500^{1}$ ) | WORD | $0 \times 1964$ | 0 | 65535 | xx01 |
| Ignore <br> module | No <br> Yes | 0 <br> 1 | BYTE | No <br> $0 x 00$ |  |  |  |
| Parameter <br> length | Internal | 6 | BYTE | 0 | 0 | 255 | $x x 02^{2}$ ) |
| Check <br> Supply | Off <br> On | 0 | BYTE | On <br> $0 x 01$ |  |  |  |
| Analog <br> Data <br> Format | Default | 0 | BYTE | Default <br> $0 x 00$ |  | 255 |  |

${ }^{1}$ ) with CS31 and addresses smaller than 70 , the value is increased by 1
${ }^{2}$ ) Value is hexadecimal: HighByte is slot ( $x x: 0 \ldots 7$ ), LowByte is index ( $1 \ldots n$ )

| GSD file: | Ext_User_Prm_Data_Len $=$ | $0 \times 09$ |
| :--- | :--- | :--- |
|  | Ext_User_Prm_Data_Const(0  <br> $)=$ $0 \times 65,0 \times 19,0 \times 06,1$ <br> $0 \times 01,0 \times 00,1$  <br> $0 x 00,0 \times 00,0 \times 00,0 \times 00 ;$  |  |

Input channel (4x)

| Name | Value | Internal <br> value | Internal <br> value, Type | Default | Min. | Max. |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Channel <br> configura- <br> tion | see table ${ }^{2}$ ) | see table ${ }^{2}$ ) | BYTE | 0 | 0 | 65535 |

Table 141: Channel configuration ${ }^{2}$ )

| Internal value | Operating modes for the analog inputs, individu- <br> ally configurable |
| :--- | :--- |
| 0 | Not used (default) |
| 1 | $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ |
| 3 | $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ |
| 4 | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ |
| 6 | $0 \mathrm{~V} \ldots 5 \mathrm{~V}$ |
| 7 | $-5 \mathrm{~V} \ldots+5 \mathrm{~V}$ |
| 20 | $-2,5 \mathrm{~V} \ldots+2,5 \mathrm{~V}$ |

Diagnosis

| E1 ... E4 | d1 | d2 | d3 | d4 | $\begin{array}{\|l\|} \hline \text { Identifier } \\ 000 \ldots 063 \end{array}$ | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err |  |  |
| Byte 6 <br> Bit 6 ... 7 | - | Byte 3 | Byte 4 | Byte 5 | Byte 6 <br> Bit 0 ... 5 |  |  |
| Class | Interface | Device | Module | Channel | Error Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | $\left.{ }^{3}\right)$ | ${ }^{4}$ ) |  |  |  |
| Module error |  |  |  |  |  |  |  |
| 3 | 14 | $1 \ldots 10$ | 31 | 31 | 19 | Checksum error in the I/O module | Replace I/O module |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |
| 3 | 14 | $1 \ldots 10$ | 31 | 31 | 9 | Overflow diagnosis buffer | Restart |
|  | 11 / 12 | ADR | 1... 10 |  |  |  |  |
| 3 | 14 | $1 . . .10$ | 31 | 31 | 26 | Parameter error | Check master |
|  | 11/12 | ADR | 1... 10 |  |  |  |  |
| 3 | 14 | $1 \ldots 10$ | 31 | 31 | 11 | Process voltage too low | Check process voltage |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |
| Channel error |  |  |  |  |  |  |  |
| 4 | 14 | $1 \ldots 10$ | 1 | $0 \ldots 3$ | 48 | Analog value overflow at an analog input | Check input value or terminal |
|  | 11 / 12 | ADR | $1 \ldots 0$ |  |  |  |  |


| E1 ... E4 | d1 | d2 | d3 | d4 | $\begin{array}{\|l\|} \hline \text { Identifier } \\ 000 \ldots 063 \end{array}$ | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err |  |  |
| Byte 6 <br> Bit 6 ... 7 | - | Byte 3 | Byte 4 | Byte 5 | Byte 6 <br> Bit 0 ... 5 |  |  |
| Class | Interface | Device | Module | Channel | Error Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | $\left.{ }^{3}\right)$ | $\left.{ }^{4}\right)$ |  |  |  |
| 4 | 14 | $1 \ldots 10$ | 1 | 0... 3 | 7 | Analog value underflow at an analog input | Check input value |
|  | 11 / 12 | ADR | $1 \ldots 0$ |  |  |  |  |

Remarks:

| ${ }^{1}$ ) | In AC500, the following interface identifier applies: $14=$ I/O bus, 11 = COM1 (e.g. CS31 bus), $12=$ COM2. <br> The PNIO diagnosis block does not contain this identifier. |
| :---: | :---: |
| ${ }^{2}$ ) | With "Device" the following allocation applies: <br> 31 = module itself, $1 \ldots 10$ = expansion module $1 \ldots 10$, ADR = hardware address (e. g. of the DC551-CS31) |
| ${ }^{3}$ ) | With "Module" the following allocation applies depending on the master: <br> Module error: I/O bus or PNIO: 31 = module itself; COM1/COM2: $1 \ldots 10=$ expansion 1 ... 10 <br> Channel error: I/O bus or PNIO = module type ( 1 = AI); COM1/COM2: 1 ... 10 = expansion 1 ... 10 |
| ${ }^{4}$ ) | In case of module errors, with channel "31 = Module itself" is output. |

## State LEDs

| LED |  | State | Color | LED = OFF | LED = ON | LED flashes |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | PWR | Process voltage 24 V DC via terminal | Green | CPU module voltage or external 24 V DC supply voltage is missing | 3.3 V system voltage (I/O bus) and external 24 V DC supply voltage are present | --- |
|  | ERR | Channel or module error | Red | No error or process voltage is missing | Severe error in the module | Error on 1 or more channels of the module |

## Risk of invalid analog input values!

The analog input values may be invalid if the measuring range of the inputs is exceeded.
Make sure that the analog signal at the connection terminals is always within the signal range.

| Range | $\begin{aligned} & -2.5 \ldots \\ & +25 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & -5 \ldots+5 \\ & V \end{aligned}$ | $0 \ldots 5 \mathrm{~V}$ | $0 . .10 \mathrm{~V}$ | $0 \ldots 20$ | $4 \ldots 20$ | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  |  | Decimal | Hex. |
| Overflow | >2.9397 | >5.8795 | >5.8795 | $\begin{aligned} & >11.758 \\ & 9 \end{aligned}$ | $\begin{aligned} & >23.517 \\ & 8 \end{aligned}$ | $\begin{aligned} & >22.814 \\ & 2 \end{aligned}$ | 32767 | 7FFF |
| Meas- <br> ured <br> value too high | $\begin{aligned} & 2.9397 \\ & : \\ & 2.5014 \end{aligned}$ | $\begin{aligned} & 5.8795 \\ & : \\ & 5.0029 \end{aligned}$ | $\begin{aligned} & 5.8795 \\ & : \\ & : \\ & : \\ & 5.0015 \end{aligned}$ | $\begin{aligned} & 11.7589 \\ & : \\ & : \\ & : \\ & 10.0029 \end{aligned}$ | $\begin{aligned} & 23.5178 \\ & : \\ & : \\ & : \\ & 20.0058 \end{aligned}$ | $\begin{aligned} & 22.8142 \\ & : \\ & : \\ & 20.0058 \end{aligned}$ | $\begin{aligned} & 32511 \\ & : \\ & 27664 \\ & 27658 \\ & 27656 \end{aligned}$ | $\begin{aligned} & 7 \mathrm{EFF} \\ & : \\ & 6 \mathrm{C} 10 \\ & 6 \mathrm{C} 0 \mathrm{~A} \\ & 6 \mathrm{C} 08 \end{aligned}$ |
| Normal range <br> Normal range or measured value too | $\begin{aligned} & 2.5000 \\ & : \\ & 0.0014 \end{aligned}$ | $\begin{aligned} & \hline 5.0000 \\ & : \\ & 0.0029 \end{aligned}$ | $\begin{aligned} & 5.0000 \\ & : \\ & : \\ & : \\ & 0.0015 \end{aligned}$ | $\begin{aligned} & 10.0000 \\ & : \\ & : \\ & : \\ & 0.0029 \end{aligned}$ | $\begin{aligned} & 20.0000 \\ & : \\ & : \\ & : \\ & 0.0058 \end{aligned}$ | $\begin{aligned} & 20.0000 \\ & : \\ & : \\ & 4.0058 \end{aligned}$ | $\begin{aligned} & 27648 \\ & : \\ & 16 \\ & 10 \\ & 8 \end{aligned}$ | $\begin{aligned} & \text { 6C00 } \\ & : \\ & 0010 \\ & 000 \mathrm{~A} \\ & 0008 \end{aligned}$ |
| low | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0 | 4 | 0 | 0000 |
|  | -0.0014 $-2.5000$ | -0.0029 \|-5.0000 |  |  |  | $\begin{aligned} & 3.9942 \\ & : \\ & : \\ & 0 \end{aligned}$ | $\begin{aligned} & -10 \\ & -16 \\ & -4864 \\ & -6912 \\ & : \\ & -27648 \end{aligned}$ | $\begin{array}{\|l} \text { FFF6 } \\ \text { FFF0 } \\ \text { ED00 } \\ \text { E500 } \\ : \\ 9400 \end{array}$ |
| Meas- <br> ured value too low | $\begin{aligned} & -2.5014 \\ & : \\ & -2.9398 \end{aligned}$ | $\begin{aligned} & -5.0029 \\ & : \\ & -5.8795 \end{aligned}$ |  |  |  |  | $\begin{aligned} & -27664 \\ & : \\ & -32512 \end{aligned}$ | $\begin{aligned} & \text { 93F0 } \\ & : \\ & 8100 \end{aligned}$ |
| Underflow | <-2.9398 | <-5.8795 | <-0.0300 | <-0.0600 | <-0.1200 | <-0.1200 | -32768 | 8000 |

The represented resolution corresponds to 12 bits including sign.

## Technical data

## Technical data of the module

The system data of AC500-eCo apply.
« Chapter 5.1.1 "System data AC500-eCo" on page 159
Only additional details are therefore documented below.

| Parameter |  | Value |
| :--- | :--- | :--- |
| Process supply voltage L+ |  |  |
|  | Connections | Terminal 19 for L+ (+24 V DC) and terminal 20 <br> for M (0 V) |
|  | Rated value | 24 V DC |
|  | Current consumption via L+ terminal | 0.1 A |
|  | Inrush current (at power up) | $0.05 \mathrm{~A}^{2} \mathrm{~s}$ |
|  | Max. ripple | $5 \%$ |
|  | Protection against reversed voltage | Yes |
|  | Protection fuse for L+ | Recommended |
| Current consumption from 24 V DC power <br> supply at the terminals UP/L+ and ZP/M of the <br> CPU/communication interface module | Ca. 10 mA |  |
| Galvanic isolation | No |  |
| Surge-voltage (max.) | 35 V DC for 0.5 s |  |
| Max. power dissipation within the module | 2.7 W |  |
| Weight | Ca. 120 g |  |
| Mounting position | Horizontal or vertical |  |
| Cooling | The natural convection cooling must not be <br> hindered by cable ducts or other parts in the <br> control cabinet. |  |

## NOTICE!

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and temporary overvoltage up to 30 V DC.

## Technical data of the analog inputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 4 individually configurable voltage or current <br> inputs |
| Distribution of channels into groups | 1 (4 channels per group) |
| Resolution |  |
| Unipolar | Voltage: $0 \mathrm{~V} \ldots+5 \mathrm{~V} ; 0 \mathrm{~V} \ldots+10 \mathrm{~V}: 12$ bits <br> Current $0 \mathrm{~mA} \ldots 20 \mathrm{~mA} ; 4 \mathrm{~mA} \ldots 20 \mathrm{~mA}: 12$ bits |
| Bipolar | Voltage $-2.5 \mathrm{~V} \ldots+2.5 \mathrm{~V} ;-5 \mathrm{~V} \ldots+5 \mathrm{~V}: 12$ bits <br> including sign |
| Connection of the signals I0- to I3- | Terminals $3,6,9,12$ |
| Connection of the signals I0+ to I3+ | Terminals $2,5,8,11$ |
| Input type | Differential |
| Galvanic isolation | No galvanic isolation between the inputs and <br> the I/O bus |
| Common mode input range | Signal voltage plus common mode voltage <br> must be within $\pm 12 \mathrm{~V}$ |


| Parameter | Value |
| :---: | :---: |
| Indication of the input signals | No |
| Channel input resistance | Voltage: > $1 \mathrm{M} \Omega$ <br> Current: ca. $250 \Omega$ |
| Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range | Typ. $\pm 0.5 \%$ of full scale (voltage) <br>  $\pm 0.5 \%$ of full scale (current 0 <br>  $\mathrm{~mA} \ldots 20 \mathrm{~mA}$ ) <br> $\pm 0.7 \%$ of full scale (current 4  <br> $\mathrm{~mA} \ldots 20 \mathrm{~mA}$ )  <br> at $+25^{\circ} \mathrm{C}$  |
|  | Max. $\pm 2 \%$ of full scale (all ranges) <br> at $0^{\circ} \mathrm{C} \ldots+60^{\circ} \mathrm{C}$ or EMC disturb- <br> ance |
| Time constant of the input filter | Voltage: $300 \mu \mathrm{~s}$ <br> Current: $300 \mu \mathrm{~s}$ |
| Relationship between input signal and hex code | Chapter 5.2.4.4.1.1.9 "Measuring ranges" on page 617 |
| Analog to digital conversion time | Typ. $500 \mu$ s per channel |
| Unused inputs | Can be left open and should be configured as "unused" |
| Input data length | 8 bytes |
| Overvoltage protection | Yes, up to 30 V DC only for voltage input |
| Max. cable length (conductor cross section $\left.>0,14 \mathrm{~mm}^{2}\right)$ |  |
| Unshielded wire | 10 m |
| Shielded wire | 100 m |

## Dimensions



The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1TNE 968 902 R1101 | Al561, analog input module, 4 AI, U/I | Active |
| 1TNE 968 901 R3101 | Terminal block TA563-9, 9 pins, screw <br> front, cable side, 6 pieces per unit | Active |
| 1TNE 968 901 R3102 | Terminal block TA563-11, 11 pins, <br> screw front, cable side, 6 pieces per <br> unit | Active |
| 1TNE 968 901 R3103 | Terminal block TA564-9, 9 pins, screw <br> front, cable front, 6 pieces per unit | Active |
| 1TNE 968901 R3104 | Terminal block TA564-11, 11 pins, <br> screw front, cable front, 6 pieces per <br> unit | Active |
| 1TNE 968 901 R3105 | Terminal block TA565-9, 9 pins, spring <br> front, cable front, 6 pieces per unit | Active |
| 1TNE 968 901 R3106 | Terminal block TA565-11, 11 pins, <br> spring front, cable front, 6 pieces per <br> unit | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

## Al562-Analog input module

## Features

- 2 configurable analog resistance temperature detector (RTD) inputs (IO and I1) in 1 group
- Resolution: 16 bits including sign


1 I/O bus
21 green LED to display power supply, 1 red LED to display error
3 Terminal number
4 Allocation of signal name
5 Terminal block for input signals (11-pin)
62 holes for wall-mounting with screws
7 DIN rail

## Intended purpose

The device can be used as a decentralized I/O extension module for S500 communication interface modules (e. g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs.

The inputs are not galvanically isolated from each other.
All other circuitry of the module is galvanically isolated from the inputs.

## Functionality

2 analog RTD-inputs, individually configurable for

- Not used (default)
- Pt100, $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$, 2-wire
- Pt100, $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$, 3-wire
- Pt1000, $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$, 2-wire
- Pt1000, $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$, 3-wire
- Ni1000, $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$, 2-wire
- Ni1000, $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$, 3-wire
- Ni100, $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$, 2-wire
- Ni100, $-50^{\circ} \mathrm{C} . . .+150{ }^{\circ} \mathrm{C}$, 3-wire
- Analog input resistance $0 \Omega \ldots 150 \Omega$
- Analog input resistance $0 \Omega \ldots 300 \Omega$

| Parameter |  |
| :--- | :--- |
| Resolution of the analog channels |  |
|  | Temperature |
| LED displays | $+0.1^{\circ} \mathrm{C}$ |
| Internal supply | 2 LEDs for process voltage and error messages |
| External supply | Via I/O bus | | Via the terminals UP (process voltage 24 V DC) and |
| :--- |
| ZP (0 V DC) |.

## Connections

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

The connection is carried out by using a removable 9-pin and 11-pin terminal block. These terminal blocks differ in their connection system (spring terminals or screw terminals, cable mounting from the front or from the side). The terminal blocks are not included in the module's scope of delivery and must be ordered separately.


Fig. 83: Internal construction of the analog inputs
The assignment of the terminals:

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 10 | O0+ | Current source of channel 0 |
| 11 | I0+ | Sense input of channel 0 |
| 12 | I0- | Return input of channel 0 |
| 13 | O1+ | Current source of channel 1 |
| 14 | I1+ | Sense input of channel 1 |
| 15 | I1- | Return input of channel 1 |
| 16 | SG | Reserved |
| 17 | SG | Shield grounding |
| 18 | UP | Shield grounding |
| 19 | ZP | Process voltage UP (24 V DC) |
| 20 | Process voltage ZP (0 V DC) |  |

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a communication interface module or a CPU). Thus, the current consumption from 24 V DC power supply at the terminals L+/UP and M/ZP of the CPU/communication interface module increases by 5 mA per AI562.
The external power supply connection is carried out via the UP (+24 V DC) and the ZP (0 V DC) terminals.

## NOTICE!

Risk of imprecise and faulty measurements!
Analog signals may be distorted seriously by external electromagnetic influences.

Use shielded wires when wiring analog signal sources. The cable shield must be grounded at both ends of the cable. Provide a potential equalisation of a low resistance to avoid high potential differences between different parts of the plant.

## NOTICE!

## Risk of damaging the PLC modules!

The PLC modules must not be removed while the plant is connected to a power supply.

Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove or replace a module.


## NOTICE!

## Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

The module provides several diagnosis functions $\Rightarrow$ Chapter 5.2.4.4.1.2.7 "Diagnosis" on page 626.

Table 142: Connection of RTDs to the inputs of the analog input module A1562


With 2-wires connection, the resistance of the connection wires influences the accuracy of the measured value. Use 3-wires connection to achieve the guaranteed measuring accuracy.

The meaning of the LEDs is described in the Displays section ${ }^{*} \Rightarrow$ Chapter 5.2.4.4.1.2.8 "State LEDs" on page 627.

## I/O configuration

The analog input module AI562 does not store configuration data itself.

## Parameterization

The arrangement of the parameter data is performed with Automation Builder software.
The parameter data directly influences the functionality of modules.
For non-standard applications, it is necessary to adapt the parameters to your system configuration.

| Name | Value | Internal <br> value | Internal <br> value, <br> Type | Default | Min. | Max. | EDS Slot <br> Index |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Module ID | Intern | $6505^{1}$ ) | WORD | $0 x 1969$ | 0 | 65535 | xx01 |
| Ignore <br> module | No <br> Yes | 0 <br> 1 | BYTE | No <br> $0 x 00$ |  |  |  |
| Parameter <br> length | Intern | 4 | BYTE | 0 | 0 | 255 | xx02 ${ }^{2}$ ) |
| Check <br> Supply | Off <br> On | 0 | BYTE | On <br> $0 x 01$ |  |  |  |
| Analog <br> Data <br> Format | Default | 0 | BYTE | Default <br> $0 x 00$ |  | 255 |  |

${ }^{1}$ ) with CS31 and addresses less than 70, the value is increased by 1
${ }^{2}$ ) Value is hexadecimal: HighByte is slot ( $x x: 0 \ldots 7$ ), LowByte is index ( $1 \ldots n$ ) GSD file:

| Ext_User_Prm_Data_Len $=$ | $0 x 07$ |
| :--- | :--- |
| Ext_User_Prm_Data_Const $(0)=$ | $0 x 6 \mathrm{~A}, 0 \times 19,0 \times 04,1$ |
|  | $0 \times 01,0 \times 00,1$ |
|  | $0 x 00,0 \times 00 ;$ |

## Input channel (2x)

| Name | Value | Internal <br> value | Internal <br> value, Type | Default | Min. | Max. |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Channel <br> configura- <br> tion | see table ${ }^{2}$ ) | see table ${ }^{2}$ ) | BYTE | 0 <br> $0 \times 00$ see <br> table $\left.{ }^{3}\right)$ | 0 | 65535 |

Table 143: Channel configuration ${ }^{2}$ )

| Internal value | Operating modes for the analog inputs, <br> individually configurable |
| :--- | :--- |
| 0 | Not used (default) <br> 3 |
| 8 | 2-wire $\operatorname{Pt} 100-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ |
| 9 | 3-wire $\operatorname{Pt} 100-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ |
| 16 | 2-wire $\mathrm{Pt} 1000,-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ |


| Internal value | Operating modes for the analog inputs, <br> individually configurable |
| :--- | :--- |
| 17 | 3-wire Pt1000, $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ |
| 18 | 2-wire Ni1000 $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ |
| 19 | 3-wire Ni1000 $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ |
| 22 | 2-wire Ni100, $-50^{\circ} \mathrm{C} \ldots+150{ }^{\circ} \mathrm{C}$ |
| 23 | 3-wire Ni100, $-50^{\circ} \mathrm{C} \ldots+150{ }^{\circ} \mathrm{C}$ |
| 32 | Analog input resistor $0 \Omega \ldots 150 \Omega$ |
| 33 | Analog input resistor $0 \Omega \ldots 300 \Omega$ |

## Diagnosis

| E1 ... E4 | d1 | d2 | d3 | d4 | Identifier $000 \ldots 063$ |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err |  |  |
| Byte 6 <br> Bit $6 . . .7$ | - | Byte 3 | Byte 4 | Byte 5 | Byte 6 <br> Bit 0 ... 5 | PNIO diagnosis block |  |
| Class | Interface | Device | Module | Channel | Error Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) | ${ }^{4}$ ) |  |  |  |
| Module err |  |  |  |  |  |  |  |
| 3 | 14 | $1 . .10$ | 31 | 31 | 19 | Checksum error in the | Replace |
|  | $11 / 12$ | ADR | $1 . .10$ |  |  | I/O module | I/O module |
| 3 | 14 | $1 . .10$ | 31 | 31 | 9 | Overflow diagnosis | Restart |
|  | 11 / 12 | ADR | 1... 10 |  |  | buf |  |
| 3 | 14 | $1 . .10$ | 31 | 31 | 26 | Parameter error | Check |
|  | 11/12 | ADR | 1... 10 |  |  |  | master |
| 3 | 14 | 1... 10 | 31 | 31 | 11 | Process voltage too low | Check |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  | process voltage |
| Channel er |  |  |  |  |  |  |  |
| 4 | 14 | 1... 10 | 1 | 0... 1 | 48 | Analog value overflow | Check |
|  | 11 / 12 | ADR | 1 ... 10 |  |  | at an analog input | input value or terminal |
| 4 | 14 | 1... 10 | 1 | 0... 1 | 7 | Analog value underflow | Check |
|  | 11 / 12 | ADR | 1 ... 10 |  |  | at an analog input | input value |

Remarks:

| ${ }^{1}$ ) | In AC500 the following interface identifier applies: $14=\mathrm{I} / \mathrm{O}$ bus, $11=$ COM1 (e.g. CS31 bus), $12=$ COM2. <br> The PNIO diagnosis block does not contain this identifier |
| :---: | :---: |
| ${ }^{2}$ ) | With "Device" the following allocation applies: <br> 31 = module itself, $1 \ldots 10$ = expansion module $1 \ldots 10$, ADR = hardware address (e. g. of the DC551-CS31) |
| ${ }^{3}$ ) | With "Module" the following allocation applies dependent of the master: <br> Module error: I/O bus or PNIO: 31 = module itself; COM1/COM2: 1 ... $10=$ expansion 1... 10 <br> Channel error: I/O bus or PNIO = module type ( 1 = AI); COM1/COM2: 1 ... 10 = expansion 1 ... 10 |
| ${ }^{4}$ ) | In case of module errors, with channel "31 = Module itself" is output. |

State LEDs

| LED |  | State <br> Process voltage 24 V DC via terminal | Color <br> Green | LED = OFF <br> CPU module voltage or external 24 V DC supply voltage is missing | LED = ON <br> 3.3 V system voltage (I/O bus) and external 24 V DC supply voltage are present | LED flashes |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | PWR |  |  |  |  |  |
|  | ERR | Channel or module error | Red | No error or process voltage is missing | Severe error in the module | Error on 1 or more channels of the module |

## Measuring ranges

## Risk of invalid analog input values!

The analog input values may be invalid if the measuring range of the inputs is exceeded.

Make sure that the analog signal at the connection terminals is always within the signal range.

## Resistance temperature detectors

| Range | $\mathrm{Pt100} / \mathrm{Pt1000}$ <br> $-50{ }^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | $\mathrm{Ni} 1000 / \mathrm{Ni100}$ <br> $-50^{\circ} \mathrm{C} \ldots+150$ |  |
| :--- | :--- | :--- | :--- | :--- |
|  |  |  | Digital value |


| Range | $\begin{aligned} & \mathrm{Pt} 100 / \mathrm{Pt} 1000 \mathrm{O} \\ & -50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \text { Ni1000 / Ni100 } \\ & -50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C} \end{aligned}$ | Digital value |  |
| :---: | :---: | :---: | :---: | :---: |
|  |  |  | Decimal | Hex. |
|  |  | $\begin{aligned} & +160.0^{\circ} \mathrm{C} \\ & : \\ & +150.1^{\circ} \mathrm{C} \end{aligned}$ | $\begin{array}{\|l\|} \hline 1600 \\ : \\ 1501 \\ \hline \end{array}$ | $\begin{array}{\|l\|} \hline 0640 \\ : \\ \text { 05DD } \\ \hline \end{array}$ |
| Normal range | $+400.0^{\circ} \mathrm{C}$ $:$ $:$ $:$ $+0.1^{\circ} \mathrm{C}$ | $\begin{aligned} & +150.0^{\circ} \mathrm{C} \\ & : \\ & : \\ & +0.1^{\circ} \mathrm{C} \end{aligned}$ | 4000 2000 1500 700 $:$ 1 | $\begin{aligned} & \text { 0FAO } \\ & \text { 07DO } \\ & \text { 05DC } \\ & \text { 02BC } \\ & : \\ & 1 \end{aligned}$ |
|  | $0,0^{\circ} \mathrm{C}$ | $0.0{ }^{\circ} \mathrm{C}$ | 0 | 0000 |
|  | $\begin{aligned} & -0.1^{\circ} \mathrm{C} \\ & : \\ & -50.0^{\circ} \mathrm{C} \end{aligned}$ | $-0.1^{\circ} \mathrm{C}$ <br> $-50.0^{\circ} \mathrm{C}$ | $\begin{aligned} & \hline-1 \\ & : \\ & -500 \\ & -2000 \end{aligned}$ | $\begin{aligned} & \text { FFFF } \\ & \text { : } \\ & \text { FE0C } \\ & \text { F830 } \end{aligned}$ |
| Measured value too low | $\begin{aligned} & -50.1^{\circ} \mathrm{C} \\ & : \\ & -60.0^{\circ} \mathrm{C} \\ & \hline \end{aligned}$ | $-50.1^{\circ} \mathrm{C}$ <br> $-60.0^{\circ} \mathrm{C}$ | $\begin{aligned} & \hline-501 \\ & : \\ & -600 \end{aligned}$ | FE0B F FDA8 |
| Underflow | $<-60.0^{\circ} \mathrm{C}$ | $<-60.0{ }^{\circ} \mathrm{C}$ | -32768 | 8000 |

## Resistances

| Range | Resistance <br> $\mathbf{0} \Omega \ldots \mathbf{1 5 0} \Omega$ | Resistance <br> $\mathbf{0} \Omega \ldots \mathbf{3 0 0} \boldsymbol{\Omega}$ | Digital value |  |
| :--- | :--- | :--- | :--- | :--- |
|  |  |  | Decimal | Hex. |
|  | $>176.383$ | $>352.767$ | 32767 | 7FFF |
| Overflow | 176.383 | 352.767 | 32511 | 7 FFF |
| Measured value <br> too high | 150.005 | 300.011 | 27649 | 6 C01 |
| Normal range | 150.000 | 300.000 | 27648 | 6 C00 |
|  | $:$ | $:$ | $:$ | $\vdots$ |
|  | 0.005 | 0.011 | 1 | 0001 |
|  | 0 | 0 | 0 | 0000 |

## Technical data

## Technical data of the module

The system data of AC500-eCo apply.

* Chapter 5.1.1 "System data AC500-eCo" on page 159

Only additional details are therefore documented below.

| Parameter |  | Value |
| :--- | :--- | :--- |
| Process supply voltage UP |  |  |
|  | Connections | Terminal 19 for UP (+24 V DC) and terminal 20 <br> for ZP (0 V) |
|  | Rated value | 24 V DC |
|  | Current consumption | 0.04 A |
|  | Inrush current (at power-up) | $0.05 \mathrm{~A}^{2} \mathrm{~s}$ |
|  | Max. ripple | $5 \%$ |
|  | Protection against reversed voltage | Yes |
|  | Protection fuse for UP | Recommended |
| Current consumption from 24 V DC power <br> supply at the terminals UP/L+ and ZP/M of <br> the CPU/communication interface module | Ca. 5 mA |  |
| Galvanic isolation | Yes, between the input group and the rest of the <br> module |  |
|  | Isolated groups | 1 (2 channels per group) |
| Surge-voltage (max.) | 35 V DC for 0.5 s |  |
| Max. power dissipation within the module | 1.1 W |  |
| Weight | Ca. 120 g |  |
| Mounting position | Horizontal or vertical |  |
| Cooling | The natural convection cooling must not be hin- <br> dered by cable ducts or other parts in the con- <br> trol cabinet. |  |

NOTICE!
All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and temporary overvoltage up to 30 V DC.

## Technical data of the analog inputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 2 configurable RTD (resistance temperature detector) <br> inputs |
| Distribution of channels into groups | 1 (2 channels per group) |
| Resolution |  |
|  | RTD |
| Resistance |  |
| Connection of the signals O0+ and <br> O1+ | Terminals 10 and 13 |
| Connection of the signals I0- and I1- | Terminals 11 and 14 |
| Connection of the signals I0+ and I1+ | Terminals 12 and 15 |
| Input type | Module ground referenced RTD for 2-wire and 3-wire <br> resistance temperature detectors |
| Galvanic isolation | Against internal power supply and other modules |


| Parameter | Value |  |
| :---: | :---: | :---: |
| Input ranges | Pt100, Pt1000, Ni100, Ni1000 |  |
|  | $150 \Omega, 300 \Omega$ |  |
| Indication of the input signals | No |  |
| Module update time | All channels: <1 s |  |
| Channel input resistance | > $100 \mathrm{k} \Omega$ |  |
| Input filter attenuation | -3 dB at 3.6 kHz |  |
| Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range | Typ. | Depending on RTD max. $\pm 0.6 \%$ of full scale (guaranteed for 3 -wires connection only) $\text { at }+25^{\circ} \mathrm{C}$ |
|  | Max. | $\pm 2 \%$ of full scale (guaranteed for 3-wires connection only) <br> at $0^{\circ} \mathrm{C} \ldots+60^{\circ} \mathrm{C}$ or EMC disturbances |
| Measuring range | « Chapter 5.2.4.4.1.2.9 "Measuring ranges" on page 627 |  |
| Analog to digital conversion time | Typ. 140 ms per channel |  |
| Unused inputs | Can be left open and should be configured as "unused" |  |
| Input data length | 4 bytes |  |
| Power dissipation inside the sensor (max.) | 1 mW |  |
| Suppression of interference | On request |  |
| Maximum input voltage | 30 V DC (sense), 5 V DC (source) |  |
| Basic error (resistance) | 0.1 \% of full-scale |  |
| Repeatability | 0.05 \% of full-scale |  |
| Overvoltage protection | Yes, up to 30 V DC |  |
| Wire loop resistance | < $20 \Omega$ |  |
| Max. cable length (conductor cross section > $0.14 \mathrm{~mm}^{2}$ ) |  |  |
| Unshielded wire | 10 m |  |
| Shielded wire | 100 m |  |

## Dimensions



The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1TNE 968 902 R1102 | Al562, analog input module, 2 AI, RTD | Active |
| 1TNE 968 901 R3102 | Terminal block TA563-11, 11 pins, <br> screw front, cable side, 6 pieces per <br> unit | Active |
| 1TNE 968 901 R3104 | Terminal block TA564-11, 11 pins, <br> screw front, cable front, 6 pieces per <br> unit | Active |
| 1TNE 968 901 R3106 | Terminal block TA565-11, 11 pins, <br> spring front, cable front, 6 pieces per <br> unit | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

## Al563 - Analog input module

## Features

- 4 configurable thermocouple (TC) / $-80 \mathrm{mV} \ldots+80 \mathrm{mV}$ inputs (I0 ... I3) in 1 group
- Resolution: 16 bits including sign


1 I/O bus
21 green LED to display power supply, 1 red LED to display error
3 Terminal number
4 Allocation of signal name
5 Terminal block for input signals (9-pin)
6 Terminal block for input signals (11-pin)
72 holes for wall-mounting with screws
8 DIN rail

Intended purpose
The device can be used as a decentralized I/O extension module for S 500 communication interface modules (e. g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs.

The inputs are group-wise galvanically isolated from each other.

The other electronic circuitry of the module is galvanically isolated from the inputs.

The I/O module must not be used as a decentralized I/O module with CI590-CS31-HA communication interface modules.

## Functionality

4 analog TC inputs, individually configurable for

- Not used (default)
- Voltage $-80 \mathrm{mV} \ldots+80 \mathrm{mV}$
- Thermocouple J-type $-210^{\circ} \mathrm{C} \ldots+1200^{\circ} \mathrm{C}$
- Thermocouple K-type $-270^{\circ} \mathrm{C} \ldots+1372{ }^{\circ} \mathrm{C}$
- Thermocouple R-type $-50^{\circ} \mathrm{C} \ldots+1768^{\circ} \mathrm{C}$
- Thermocouple S-type $-50^{\circ} \mathrm{C} \ldots+1768^{\circ} \mathrm{C}$
- Thermocouple T-type $-270^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$
- Thermocouple E-type $-270^{\circ} \mathrm{C} \ldots+1000^{\circ} \mathrm{C}$
- Thermocouple N-type $-270^{\circ} \mathrm{C} \ldots+1300^{\circ} \mathrm{C}$

| Parameter | Value |
| :--- | :--- |
| Resolution of the analog channels |  |
| Temperature | $+0.1^{\circ} \mathrm{C}$ |
| LED displays | 2 LEDs for process voltage and error mes- <br> sages |
| Internal supply | Via I/O bus |
| External supply | Via the terminals UP (process voltage 24 V <br> DC $)$ and ZP (0 V DC) |

## Connections

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

After powering up the system, input channels, which are configured will have undefined values /diagnosis message for typically 45 seconds, if the wires of all configured channels are broken.

If the AI563 is connected to a PROFINET communication interface module, the firmware version of PROFINET communication interface module must be 1.2 or above.

The connection is carried out by using a removable 9-pin and 11-pin terminal block. These terminal blocks differ in their connection system (spring terminals or screw terminals, cable mounting from the front or from the side). The terminal blocks are not included in the module's scope of delivery and must be ordered separately.


Fig. 84: Internal construction of the analog inputs

Table 144: Assignment of the terminals

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 1 | IO+ | Positive pole of channel 0 |
| 2 | IO- | Negative pole of channel 0 |
| 3 | I1+ | Positive pole of channel 1 |
| 4 | I1- | Negative pole of channel 1 |
| 5 | I2+ | Positive pole of channel 2 |
| 6 | I2- | Negative pole of channel 2 |
| 7 | I3+ | Positive pole of channel 3 |
| 8 | I3- | Negative pole of channel 3 |
| 9 | --- | Reserved |
| 10 | --- | Reserved |
| 11 | --- | Reserved |
| 12 | --- | Reserved |
| 13 | --- | Reserved |


| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 14 | --- | Reserved |
| 15 | SG | Shield grounding |
| 16 | SG | Shield grounding |
| 17 | SG | Shield grounding |
| 18 | SG | Shield grounding |
| 19 | UP | Process voltage UP (24 V DC) |
| 20 | ZP | Process voltage ZP (0 V DC) |

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a communication interface module or a CPU). Thus, the current consumption from 24 V DC power supply at the terminals UP/L+ and ZP/M of the CPU/communication interface module increases by 5 mA per AI563.

The external power supply connection is carried out via the UP (+24 V DC) and the ZP (0 V DC) terminals.

## NOTICE!

Risk of imprecise and faulty measurements!
Analog signals may be distorted seriously by external electromagnetic influences.

Use shielded wires when wiring analog signal sources. The cable shield must be grounded at both ends of the cable. Provide a potential equalisation of a low resistance to avoid high potential differences between different parts of the plant.

## NOTICE!

Risk of damaging the PLC modules!
The PLC modules must not be removed while the plant is connected to a power supply.

Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove or replace a module.


## NOTICE!

## Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

The module provides several diagnosis functions ${ }^{\circ} \Rightarrow$ Chapter 5.2.4.4.1.3.7 "Diagnosis" on page 638.
The following figure shows the connection of thermocouples to the inputs of the module:


Fig. 85: Connection of thermocouples to the inputs of the module
The meaning of the LEDs is described in Displays chapter ${ }^{〔}$ Chapter 5.2.4.4.1.3.8 "State LEDs" on page 639.

I/O configuration
The analog input module AI563 does not store configuration data itself.

## Parameterization

The arrangement of the parameter data is performed with Automation Builder software.
The parameter data directly influences the functionality of modules.
For non-standard applications, it is necessary to adapt the parameters to your system configuration.

| Name | Value | Internal <br> value | Internal <br> value, <br> Type | Default | Min. | Max. | EDS Slot <br> Index |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Module ID | Intern | $6510^{1}$ ) | WORD | $0 x 196 E$ | 0 | 65535 | xx01 |
| Ignore <br> module | No <br> Yes | 0 <br> 1 | BYTE | No <br> $0 x 00$ |  |  |  |
| Parameter <br> length | Intern | 6 | BYTE | 0 | 0 | 255 | $x x 02^{2}$ ) |
| Check <br> Supply | Off <br> On | 0 | BYTE | On <br> $0 x 01$ |  | 255 |  |
| Analog <br> Data <br> Format | Default | 0 | BYTE | Default <br> $0 x 00$ |  |  |  |
| 1) with CS31 and addresses less than 70, the value is increased by 1 <br> 2) Value is hexadecimal: HighByte is slot (xx: 0 ... 7), LowByte is index (1 ...n) |  |  |  |  |  |  |  |

GSD file:

| Ext_User_Prm_Data_Len $=$ | $0 x 09$ |
| :--- | :--- |
| Ext_User_Prm_Data_Const $(0)=$ | $0 x 6 \mathrm{~F}, 0 \times 19,0 \times 06,1$ |
|  | $0 \times 01,0 \times 00,1$ |
|  | $0 x 00,0 \times 00,0 \times 00,0 \times 00 ;$ |

Input channel (4x)

| Name | Value | Internal <br> value | Internal <br> value, Type | Default | Min. | Max. |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Channel <br> configura- <br> tion | see table ${ }^{2}$ ) | see table ${ }^{2}$ ) | BYTE | 0 <br> 0x00 see <br> table $^{2}$ ) | 0 | 65535 |

Table 145: Channel configuration ${ }^{2}$ )

| Internal value | Operating modes for the analog inputs, individually configurable |
| :--- | :--- |
| 0 | Not used (default) |
| 21 | Voltage $-80 \mathrm{mV} \ldots+80 \mathrm{mV}$ |
| 24 | Thermocouple J-type $-210^{\circ} \mathrm{C} \ldots+1200{ }^{\circ} \mathrm{C}$ |
| 25 | Thermocouple K-type $-270{ }^{\circ} \mathrm{C} \ldots+1372{ }^{\circ} \mathrm{C}$ |
| 26 | Thermocouple R-type $-50^{\circ} \mathrm{C} \ldots+1768^{\circ} \mathrm{C}$ |
| 27 | Thermocouple S-type $-50^{\circ} \mathrm{C} \ldots+1760^{\circ} \mathrm{C}$ |
| 28 | Thermocouple T-type $-270^{\circ} \mathrm{C} \ldots+400{ }^{\circ} \mathrm{C}$ |
| 29 | Thermocouple E-type $-270{ }^{\circ} \mathrm{C} \ldots+1000{ }^{\circ} \mathrm{C}$ |
| 30 | Thermocouple N-type $-270^{\circ} \mathrm{C} \ldots+1300^{\circ} \mathrm{C}$ |

Diagnosis

| E1 ... E4 | d1 | d2 | d3 | d4 | Identifier $000 \ldots 063$ | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err |  |  |
| Byte 6 <br> Bit 6 ... 7 | - | Byte 3 | Byte 4 | Byte 5 | Byte 6 <br> Bit 0 ... 5 |  |  |
| Class | Interface | Device | Module | Channel | Error Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) | ${ }^{4}$ ) |  |  |  |
| Module error |  |  |  |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 19 | Checksum error in the I/O module | Replace I/O module |
|  | 11/12 | ADR | 1... 10 |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 9 | Overflow diagnosis buffer | Restart |
|  | 11/12 | ADR | $1 . .10$ |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 26 | Parameter error | Check master |
|  | 11 / 12 | ADR | 1... 10 |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 11 | Process voltage too low | Check process voltage |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |
| Channel error |  |  |  |  |  |  |  |
| 4 | 14 | 1... 10 | 1 | 0... 3 | 48 | Analog value overflow or broken wire at an analog input | Check input value or terminal |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |
| 4 | 14 | 1... 10 | 1 | 0... 3 | 7 | Analog value underflow at an analog input | Check input value |
|  | 11/12 | ADR | $1 . .10$ |  |  |  |  |

Remarks:

| ${ }^{1}$ ) | In AC500 the following interface identifier applies: 14 = I/O bus, 11 = COM1 (e.g. CS31 bus), $12=$ COM2. <br> The PNIO diagnosis block does not contain this identifier. |
| :---: | :---: |
| ${ }^{2}$ ) | With "Device" the following allocation applies: <br> 31 = module itself, $1 \ldots 10=$ expansion module $1 . . .10$, ADR $=$ hardware address (e. g. of the DC551-CS31) |
| ${ }^{3}$ ) | With "Module" the following allocation applies dependent of the master: <br> Module error: I/O bus or PNIO: 31 = module itself; COM1/COM2: 1 ... 10 = expansion 1 ... 10 <br> Channel error: I/O bus or PNIO = module type ( $1=\mathrm{AI}$ ); COM1/ COM2: 1 ... 10 = expansion 1 ... 10 |
| ${ }^{4}$ ) | In case of module errors, with channel "31 = Module itself" is output. |

## State LEDs

| LED |  | State | Color | LED = OFF | LED = ON | LED flashes |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | PWR | Process voltage 24 V DC via terminal | Green | CPU module voltage or external 24 V DC supply voltage is missing | 3.3 V system voltage (I/O bus) and external 24 V DC supply voltage are present | --- |
|  | ERR | Channel or module error | Red | No error or process voltage is missing | Severe error in the module | Error on 1 or more channels of the module |

## Measuring ranges

Al563 needs typ. 6 to 8 seconds for initialization after applying the process supply voltage to clamp UP/ZP. During this time, the accuracy of the measurement values is not within specification. After that, valid measurement values are provided by the module. After that, valid measurement values are provided by the module.

After an interruption of the process supply voltage > 10 ms , a re-initialization is performed by Al563.

## Risk of invalid analog input values!

The analog input values may be invalid if the measuring range of the inputs is exceeded.

Make sure that the analog signal at the connection terminals is always within the signal range.

When a wire break occurs on a sensor wire, the temperature measurement value of the corresponding channel changes to Overflow (Hexadecimal 7FFF).

| Range | $\begin{aligned} & \text { Type J } \\ & -210^{\circ} \mathrm{C} \ldots \\ & +1200^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \text { Type K } \\ & 270^{\circ} \mathrm{C} \ldots \\ & +1372^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \hline \text { Type N } \\ & 270^{\circ} \mathrm{C} \ldots . \\ & +1300^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \hline \text { Type } \mathrm{T} \\ & -270^{\circ} \mathrm{C} \ldots \\ & +400^{\circ} \mathrm{C} \end{aligned}$ | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  | Decimal | Hex. |
| Overflow | $\begin{aligned} & >+1200.0 \\ & { }^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & >+1372.0 \\ & { }^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & >+1300.0 \\ & { }^{\circ} \mathrm{C} \end{aligned}$ | >+400.0 ${ }^{\circ} \mathrm{C}$ | 32767 | 7FFF |
| Normal range |  |  |  |  | 17680 | 4510 |
|  |  | $+1372.0{ }^{\circ} \mathrm{C}$ |  |  | 13720 | 3598 |
|  |  | : | $+1300.0{ }^{\circ} \mathrm{C}$ |  | 13000 | 32C8 |
|  | $+1200.0{ }^{\circ} \mathrm{C}$ | : | : |  | 12000 | 2EE0 |
|  | : | : | : | $+400.0{ }^{\circ} \mathrm{C}$ | 4000 | OFAO |
|  | : | : | : | : | : | : |


| Range | $\begin{aligned} & \hline \text { Type J } \\ & -210^{\circ} \mathrm{C} \ldots \\ & +1200^{\circ} \mathrm{C} \end{aligned}$ | Type K $270^{\circ} \mathrm{C}$... $+1372^{\circ} \mathrm{C}$ | Type N $270^{\circ} \mathrm{C}$... $+1300{ }^{\circ} \mathrm{C}$ | $\begin{aligned} & \text { Type T } \\ & -270^{\circ} \mathrm{C} \ldots \\ & +400^{\circ} \mathrm{C} \end{aligned}$ | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  | Decimal | Hex. |
|  | $+0.1{ }^{\circ} \mathrm{C}$ | $+0.1{ }^{\circ} \mathrm{C}$ | $+0.1{ }^{\circ} \mathrm{C}$ | $+0.1{ }^{\circ} \mathrm{C}$ | 1 | 1 |
|  | $+0.0{ }^{\circ} \mathrm{C}$ | $+0.0{ }^{\circ} \mathrm{C}$ | $+0.0{ }^{\circ} \mathrm{C}$ |  | 0 | 0000 |
|  | $-0.1{ }^{\circ} \mathrm{C}$ | $-0.1{ }^{\circ} \mathrm{C}$ | $-0.1{ }^{\circ} \mathrm{C}$ | $-0.1{ }^{\circ} \mathrm{C}$ | -1 | FFFF |
|  | : | : | . |  | : | - |
|  | : | : | : | . | -500 | FEOC |
|  | $-210.0^{\circ} \mathrm{C}$ | : | . | . | -2100 | F7CC |
|  |  | $-270.0^{\circ} \mathrm{C}$ | $-270.0^{\circ} \mathrm{C}$ | $-270.0^{\circ} \mathrm{C}$ | -2700 | F574 |
| Underflow | $<-210.0{ }^{\circ} \mathrm{C}$ | $<-270.0{ }^{\circ} \mathrm{C}$ | $<-270.0{ }^{\circ} \mathrm{C}$ | $<-270.0{ }^{\circ} \mathrm{C}$ | -32768 | 8000 |


| Range | $\begin{aligned} & -80 \mathrm{mV} \ldots+80 \\ & \mathrm{mV} \end{aligned}$ | $\begin{aligned} & \hline \text { Type E } \\ & -270^{\circ} \mathrm{C} \ldots \\ & +1000^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \text { Types R, S } \\ & -50^{\circ} \mathrm{C} \ldots \\ & +1768{ }^{\circ} \mathrm{C} \end{aligned}$ | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  | Decimal | Hex. |
| Overflow | > +90 mV | $>+1000.0{ }^{\circ} \mathrm{C}$ | >+1768.0 ${ }^{\circ} \mathrm{C}$ | 32767 | 7FFF |
| Normal range | +80 mV |  |  | 27648 | 6C00 |
|  |  |  | $+1768.0{ }^{\circ} \mathrm{C}$ | 17680 | 4510 |
|  |  | $+1000.0{ }^{\circ} \mathrm{C}$ |  | 10000 | 2710 |
|  |  |  |  | 9000 | 2328 |
|  | : | : | . | . | . |
|  | $3 \mu \mathrm{~V}$ | $+0.1{ }^{\circ} \mathrm{C}$ | $+0.1{ }^{\circ} \mathrm{C}$ | 1 | 1 |
|  | $0 \mu \mathrm{~V}$ | $0.0{ }^{\circ} \mathrm{C}$ | $0.0^{\circ} \mathrm{C}$ | 0 | 0000 |
|  | $-3 \mu \mathrm{~V}$ | $-0.1{ }^{\circ} \mathrm{C}$ | $-0.1{ }^{\circ} \mathrm{C}$ | -1 | FFFF |
|  | : | : |  |  |  |
|  | : | : | $-50.0{ }^{\circ} \mathrm{C}$ | -500 | FEOC |
|  | : | $-270.0{ }^{\circ} \mathrm{C}$ |  | -2700 | F574 |
|  | -80 mV |  |  | -27648 | 9400 |
| Underflow | $<-90 \mathrm{mV}$ | $<-270.0{ }^{\circ} \mathrm{C}$ | $<-50.0{ }^{\circ} \mathrm{C}$ | -32768 | 8000 |

## Technical data

## Technical data of the module

The system data of AC500-eCo apply.
年 Chapter 5.1.1 "System data AC500-eCo" on page 159
Only additional details are therefore documented below.

| Parameter |  | Value |
| :--- | :--- | :--- |
| Process supply voltage UP |  |  |
|  | Connections | Terminal 19 for UP (+24 V DC) and terminal 20 <br> for ZP (0 V) |
|  | Rated value | 24 V DC |
|  | Current consumption | 0.10 A |
|  | Inrush current (at power-up) | $0.07 \mathrm{~A}^{2} \mathrm{~s}$ |
|  | Max. ripple | $5 \%$ |
|  | Protection against reversed voltage | Yes |
|  | Rated protection fuse for UP | Not necessary |
| Current consumption from 24 V DC power <br> supply at the terminals UP/L+ and ZP/M of <br> the CPU/communication interface module | Ca. 5 mA |  |
| Galvanic isolation | Yes, between the channels and the rest of the <br> module |  |
|  | Isolated groups | 1 (4 channels per group) |
| Surge-voltage (max.) | 35 V DC for 0.5 s |  |
| Max. power dissipation within the module | 2.6 W |  |
| Weight | Ca. 120 g |  |
| Mounting position | Horizontal or vertical |  |
| Cooling | The natural convection cooling must not be hin- <br> dered by cable ducts or other parts in the con- <br> trol cabinet. |  |

NOTICE!
All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and temporary overvoltage up to 30 V DC.

## Technical data of the analog inputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 4 configurable thermocouple (TC) inputs |
| Distribution of channels into groups | 1 (4 channels per group) |
| Resolution |  |
|  | Temperature |
| Voltage | $0.1^{\circ} \mathrm{C}$ |
| Connection of the signals I0+ to I3+ | Terminals 1, 3, 5 and 7 |
| Connection of the signals I0- to I3- | Terminals 2, 4, 6 and 8 |
| Input type | Floating thermocouple |
| Galvanic isolation | Against internal power supply and other modules |
| Common mode rejection | $>120$ dB at 120 V AC |
| Indication of the input signals | No |
| Module update time | All channels: < 1.6 s |


| Parameter | Value |  |
| :--- | :--- | :--- |
| Channel input resistance | On request |  |
| Input filter attenuation | -3 dB at 15 kHz |  |
| Cold junction error | $\pm 1.5^{\circ} \mathrm{C}$ |  |
| Conversion error of the analog values <br> caused by non-linearity, adjustment <br> error at factory and resolution within <br> the normal range | Typ. |  |

## Accuracy of thermocouple ranges at $25^{\circ} \mathrm{C}$ (with cold junction compensation)

| Thermocouple Type | Range | Accuracy |
| :--- | :--- | :--- |
| E | $-270^{\circ} \mathrm{C} \ldots-220^{\circ} \mathrm{C}$ | $\pm 2 \%$ |
|  | $-220^{\circ} \mathrm{C} \ldots+1000^{\circ} \mathrm{C}$ | $\pm 0.6 \%$ |
| J | $-210^{\circ} \mathrm{C} \ldots+1200^{\circ} \mathrm{C}$ | $\pm 0.6 \%$ |
| K | $-270^{\circ} \mathrm{C} \ldots-220^{\circ} \mathrm{C}$ | $\pm 1.5 \%$ |
|  | $-220^{\circ} \mathrm{C} \ldots+1372{ }^{\circ} \mathrm{C}$ | $\pm 0.6 \%$ |
| N | $-270^{\circ} \mathrm{C} \ldots-150^{\circ} \mathrm{C}$ | $\pm 2 \%$ |
|  | $-150^{\circ} \mathrm{C} \ldots+1300^{\circ} \mathrm{C}$ | $\pm 0.6 \%$ |
| R | $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ | $\pm 1.5 \%$ |
| S | $+150^{\circ} \mathrm{C} \ldots+1768{ }^{\circ} \mathrm{C}$ | $\pm 0.6 \%$ |
| T | $-50^{\circ} \mathrm{C} \ldots+150{ }^{\circ} \mathrm{C}$ | $\pm 1.5 \%$ |
|  | $+150^{\circ} \mathrm{C} \ldots+1768{ }^{\circ} \mathrm{C}$ | $\pm 0.6 \%$ |
|  | $-270^{\circ} \mathrm{C} \ldots-240^{\circ} \mathrm{C}$ | $\pm 3 \%$ |
|  | $-240^{\circ} \mathrm{C} \ldots-0^{\circ} \mathrm{C}$ | $\pm 2 \%$ |
| $0^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | $\pm 0.6 \%$ |  |

These accuracy values are valid only for stable module temperatures.

## Dimensions



The dimensions are in mm and in brackets in inch.

Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1TNE 968 902 R1103 | Al563, analog input module, 4 AI, <br> thermocouple | Active |
| 1TNE 968 901 R3101 | Terminal block TA563-9, 9 pins, screw <br> front, cable side, 6 pieces per unit | Active |
| 1TNE 968 901 R3102 | Terminal block TA563-11, 11 pins, <br> screw front, cable side, 6 pieces per <br> unit | Active |
| 1TNE 968901 R3103 | Terminal block TA564-9, 9 pins, screw <br> front, cable front, 6 pieces per unit | Active |
| 1TNE 968 901 R3104 | Terminal block TA564-11, 11 pins, <br> screw front, cable front, 6 pieces per <br> unit | Active |


| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1TNE 968 901 R3105 | Terminal block TA565-9, 9 pins, spring <br> front, cable front, 6 pieces per unit | Active |
| 1TNE 968 901 R3106 | Terminal block TA565-11, 11 pins, <br> spring front, cable front, 6 pieces per <br> unit | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

## AO561-Analog output module

## Features

- 2 configurable analog outputs (O0 ... O1) in 1 group
- Resolution: 12 bits including sign


1 I/O bus
21 green LED to display power supply, 1 red LED to display error
3 Terminal number
4 Allocation of signal name
5 Terminal block for output signals (11-pin)
62 holes for wall-mounting with screws
7 DIN rail

Intended purpose
The device can be used as a decentralized I/O extension module for S 500 communication interface modules (e. g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs.

The outputs are not galvanically isolated from each other.
The other electronic circuitry of the module is not galvanically isolated from the outputs or from the I/O bus

The I/O module must not be used as communication interface module at CI590-CS31-HA bus modules.

## Functionality

2 analog outputs, individually configurable for

- Not used (default setting)
-     - 10 V ... +10 V
- $0 \mathrm{~mA} . .20 \mathrm{~mA}$
- 4 mA ... 20 mA

| Parameter | Value |
| :--- | :--- |
| Resolution of the analog channels |  |
|  | Voltage bipolar $(-10 \mathrm{~V} \ldots+10 \mathrm{~V})$ |
|  | Current $(0 \mathrm{~mA} \ldots 20 \mathrm{~mA} ; 4 \mathrm{~mA} \ldots 20 \mathrm{~mA})$ |
| LED displays | 12 bits including sign |
| Internal supply | 2 LEDs for process voltage and error messages |
| External supply | Via I/O bus |
|  | Via the terminals $\mathrm{L}+$ (process voltage 24 V DC$)$ <br> and $\mathrm{M}(0 \mathrm{~V}$ DC $) ;$ the M terminal is connected to <br> the M terminal of the CPU via the I/O bus |

## Connections

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

If the output is configured as not used, the voltage and current output signals are undefined and must not be connected.

The connection is carried out by using a removable 9-pin and 11-pin terminal block. These terminal blocks differ in their connection system (spring terminals or screw terminals, cable mounting from the front or from the side). The terminal blocks are not included in the module's scope of delivery and must be ordered separately.


Fig. 86: Internal construction of the analog outputs
The assignment of the terminals:

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 10 | --- | Reserved |
| 11 | --- | Reserved |
| 12 | --- | Reserved |
| 13 | O0U+ | Voltage output of channel 0 |
| 14 | O0I+ | Current output of channel 0 |
| 15 | O1U+ | Voltage output of channel 1 |
| 16 | O1I+ | Current output of channel 1 |
| 17 | O01- | Negative pole of channels O0 and O1 |
| 18 | L+ | Shield grounding |
| 19 | M | Process voltage L+ (24 V DC) |
| 20 | Process voltage M (0 V DC) |  |

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a communication interface module or a CPU). Thus, the current consumption from $24 \vee$ DC power supply at the terminals UP/L+ and ZP/M of the CPU/communication interface module increases by 5 mA per AO561.

The external power supply connection is carried out via the $\mathrm{L}+(+24 \mathrm{~V} \mathrm{DC}$ ) and the $\mathrm{M}(0 \mathrm{~V}$ $D C$ ) terminals. The $M$ terminal is electrically interconnected to the $M / Z P$ terminal of the CPU/ communication interface module.

## NOTICE!

Risk of imprecise and faulty measurements!
Analog signals may be distorted seriously by external electromagnetic influences.

Use shielded wires when wiring analog signal sources. The cable shield must be grounded at both ends of the cable. Provide a potential equalisation of a low resistance to avoid high potential differences between different parts of the plant.

## NOTICE!

## Risk of damaging the PLC modules!

The PLC modules must not be removed while the plant is connected to a power supply.

Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove or replace a module.


## NOTICE!

## Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

The module provides several diagnosis functions Chapter 5.2.4.4.1.4.7 "Diagnosis" on page 650.

Table 146: Connection of analog actuators to the analog output module AO561


The output signal is undefined if the supply voltage at the $L+$ terminal is below 10 V . This can, for example, occur if the supply voltage has a slow ramp-up / ramp-down behavior and must be foreseen when planning the installation.

If the output is configured in current mode, the voltage output signal is undefined and must not be connected.

If the output is configured in voltage mode, the current output signal is undefined and must not be connected.

## I/O configuration

The analog output module A0561 does not store configuration data itself.

## Parameterization

The arrangement of the parameter data is performed with Automation Builder software.
The parameter data directly influences the functionality of modules.
For non-standard applications, it is necessary to adapt the parameters to your system configuration.

| Name | Value | Internal value | Internal value, Type | Default | Min. | Max. | EDS Slot Index |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Module ID | Intern | $6515{ }^{1}$ ) | WORD | 0x1973 | 0 | 65535 | xx01 |
| Ignore module | $\begin{aligned} & \text { No } \\ & \text { Yes } \end{aligned}$ | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | BYTE | $\begin{array}{\|l\|} \hline \text { No } \\ 0 \times 00 \end{array}$ |  |  |  |
| Parameter length | Intern | 4 | BYTE | 0 | 0 | 255 | $\mathrm{xx} 02{ }^{2}$ ) |
| Check Supply | $\begin{aligned} & \text { Off } \\ & \text { On } \end{aligned}$ | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | BYTE | $\begin{aligned} & \mathrm{On} \\ & 0 \times 01 \end{aligned}$ |  |  |  |
| Analog <br> Data <br> Format | Default | 0 | BYTE | Default 0x00 |  | 255 |  |
| ${ }^{1}$ ) with CS3 <br> ${ }^{2}$ ) Value is | 1 and ad | : HighByt | an 70, the | alue is in | te is in | $1 \ldots \mathrm{n})$ |  |

GSD file:

| Ext_User_Prm_Data_Len $=$ | $0 \times 07$ |
| :--- | :--- |
| Ext_User_Prm_Data_Const $(0)=$ | $0 \times 74,0 \times 19,0 \times 04,1$ |
|  | $0 \times 01,0 \times 00,1$ |
|  | $0 \times 00,0 \times 00,0 \times 00,0 \times 00 ;$ |

Output channel (2x)

| Name | Value | Internal <br> value | Internal <br> value, Type | Default | Min. | Max. |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Channel <br> configura- <br> tion | see table ${ }^{2}$ ) | see table ${ }^{2}$ ) | BYTE | 0 <br> $0 x 00$ see <br> table $\left.{ }^{2}\right)$ | 0 | 65535 |

Table 147: Channel configuration ${ }^{2}$ )

| Internal value | Operating modes for the analog outputs, individually configu- <br> rable |
| :--- | :--- |
| 0 | Not used (default) |
| 128 | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ |
| 129 | $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ |
| 130 | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ |

## Diagnosis

| E1 ... E4 | d1 | d2 | d3 | d4 | $\begin{aligned} & \text { Identifier } \\ & 000 \ldots 063 \end{aligned}$ | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err |  |  |
| Byte 6 <br> Bit 6 ... 7 | - | Byte 3 | Byte 4 | Byte 5 | Byte 6 <br> Bit 0 ... 5 |  |  |
| Class | Interface | Device | Module | Channel | Error Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) | ${ }^{4}$ ) |  |  |  |
| Module error |  |  |  |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 19 | Checksum error in the I/O module | Replace I/O module |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 9 | Overflow diagnosis buffer | Restart |
|  | 11 / 12 | ADR | 1... 10 |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 26 | Parameter error | Check master |
|  | 11 / 12 | ADR | 1... 10 |  |  |  |  |
| 3 | 14 | $1 \ldots 10$ | 31 | 31 | 11 | Process voltage too low | Check process voltage |
|  | 11/12 | ADR | $1 . .10$ |  |  |  |  |
| Channel error |  |  |  |  |  |  |  |
| 4 | 14 | 1... 10 | 3 | 0... 1 | 48 | Analog value overflow at an analog output | Check output value or terminal |
|  | 11/12 | ADR | 1... 10 |  |  |  |  |
| 4 | 14 | 1... 10 | 3 | $0 \ldots 1$ | 7 | Analog value underflow at an analog output | Check output value |
|  | 11/12 | ADR | $1 . .10$ |  |  |  |  |

Remarks:

| ${ }^{1}$ ) | In AC500 the following interface identifier applies: 14 = I/O bus, 11 = COM1 (e.g. CS31 bus), $12=$ COM2. <br> The PNIO diagnosis block does not contain this identifier. |
| :---: | :---: |
| ${ }^{2}$ ) | With "Device" the following allocation applies: <br> 31 = module itself, $1 \ldots 10=$ expansion module $1 \ldots 10$, ADR = hardware address (e. g. of the DC551-CS31) |
| $\left.{ }^{3}\right)$ | With "Module" the following allocation applies depending on the master: <br> Module error: I/O bus or PNIO: 31 = module itself; COM1/COM2: 1 ... $10=$ expansion 1 ... 10 <br> Channel error: I/O bus or PNIO = module type ( 3 = AO); COM1/COM2: 1 ... 10 = expansion 1 ... 10 |
| ${ }^{4}$ ) | In case of module errors, with channel "31 = Module itself" is output. |

## State LEDs

| LED |  | State | Color | LED = OFF | LED = ON | LED flashes |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |

## Output ranges

| Range | -10 ... +10 V | 0 ... 20 mA | $4 \ldots 20 \mathrm{~mA}$ | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  | Decimal | Hex. |
| Overflow | >11.7589 | >23.5178 | >22.8142 | 32767 | 7FFF |
| Value too high | $\begin{aligned} & 11.7589 \\ & : \\ & 10.0058 \end{aligned}$ | $\begin{aligned} & 23.5178 \\ & : \\ & : \\ & : \\ & 20.0058 \end{aligned}$ | $\begin{aligned} & 22.8142 \\ & : \\ & : \\ & 20.0058 \end{aligned}$ | $\begin{aligned} & 32511 \\ & : \\ & 27664 \\ & 27658 \\ & 27656 \end{aligned}$ | $\begin{aligned} & \text { 7EFF } \\ & : \\ & 6 \mathrm{C} 10 \\ & 6 \mathrm{COA} \\ & 6 \mathrm{C} 08 \end{aligned}$ |
| Normal range <br> Normal range or value too low | $\begin{aligned} & 10.0000 \\ & : \\ & 0.0058 \end{aligned}$ | $\begin{aligned} & 20.0000 \\ & : \\ & : \\ & : \\ & 0.0058 \end{aligned}$ | $\begin{aligned} & 20.0000 \\ & : \\ & : \\ & 4.0058 \end{aligned}$ | $\begin{aligned} & 27648 \\ & : \\ & 16 \\ & 10 \\ & 8 \end{aligned}$ | $\begin{aligned} & 6 \mathrm{C} 00 \\ & : \\ & 0010 \\ & 000 \mathrm{~A} \\ & 0008 \end{aligned}$ |
|  | 0.0000 | 0 | 4 | 0 | 0000 |


| Range | -10 ... +10 V | 0 ... 20 mA | 4 ... 20 mA | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  | Decimal | Hex. |
|  | \|-0.0058 |-10.0000 |  | $\begin{aligned} & 3.9942 \\ & \vdots \\ & \vdots \\ & 0 \end{aligned}$ | -10 <br> -16 <br> -4864 <br> -6912 <br> -27648 | $\begin{aligned} & \text { FFF6 } \\ & \text { FFF0 } \\ & \text { ED00 } \\ & \text { E500 } \\ & : \\ & 9400 \end{aligned}$ |
| Value too low | $\begin{aligned} & -10.0058 \\ & : \\ & -11.7589 \end{aligned}$ |  |  |  | $\begin{aligned} & 93 \text { F0 } \\ & : \\ & 8100 \end{aligned}$ |
| Underflow | <-11.7589 |  | <0.0000 | -32768 | 8000 |

The represented resolution corresponds to 12 bits including sign.

## Technical data

## Technical data of the module

The system data of AC500-eCo apply.
*) Chapter 5.1.1 "System data AC500-eCo" on page 159
Only additional details are therefore documented below.

| Parameter | Value |  |
| :--- | :--- | :--- |
| Process supply voltage L+ |  |  |
|  | Connections | Terminal 19 for L+ (+24 V DC) and terminal 20 <br> for M (0 V) |
|  | Rated value | 24 V DC |
|  | Current consumption | $0.1 \mathrm{~A}+$ output load |
|  | Inrush current (at power-up) | $0.05 \mathrm{~A}^{2} \mathrm{~s}$ |
|  | Max. ripple | $5 \%$ |
|  | Protection against reversed voltage | Yes |
|  | Protection fuse for L+ | Recommended |
| Current consumption from 24 V DC power <br> supply at the terminals UP/L+ and ZP/M of <br> the CPU/communication interface module | Ca. 5 mA |  |
| Galvanic isolation | No |  |
| Surge-voltage (max.) | $35 \mathrm{~V} \mathrm{DC} \mathrm{for} \mathrm{0.5} \mathrm{~s}$ |  |
| Max. power dissipation within the module | 3.1 W |  |
| Weight | Ca. 120 g |  |
| Mounting position | Horizontal or vertical |  |
| Cooling | The natural convection cooling must not be hin- <br> dered by cable ducts or other parts in the con- <br> trol cabinet. |  |

## NOTICE!

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and temporary overvoltage up to 30 V DC.

## Technical data of the analog outputs

| Parameter | Value |  |
| :---: | :---: | :---: |
| Number of channels per module | 2 configurable voltage or current outputs |  |
| Distribution of channels into groups | 1 (2 channels per group) |  |
| Connection of the signals O0U- and O1U+ | Terminals 13 and 15 |  |
| Connection of the signals $\mathrm{OOI}+$ and $\mathrm{O} 11+$ | Terminals 14 and 16 |  |
| Output type | Bipolar with voltage, unipolar with current |  |
| Resolution | 12 bits including sign |  |
| Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range | Typ. | $\pm 0.5 \%$ of full scale at $+25^{\circ} \mathrm{C}$ |
|  | Max. | $\pm 2 \%$ of full scale at $0^{\circ} \mathrm{C} \ldots+60^{\circ} \mathrm{C}$ or EMC disturbances |
| Indication of the output signals | No |  |
| Output Resistance (load) as current output | $0 \Omega \ldots 500 \Omega$ |  |
| Output load ability as voltage output | $\pm 2 \mathrm{~mA}$ max. |  |
| Output data length | 4 bytes |  |
| Relationship between output signal and hex code | ® Chapter 5.2.4.4.1.4.9 "Output ranges" on page 651 |  |
| Unused outputs | Must not be connected and must be configured as "unused" |  |
| Overvoltage protection | Yes, up to 30 V DC |  |
| Max. cable length (conductor cross section $>0.14 \mathrm{~mm}^{2}$ ) |  |  |
| Unshielded wire | 10 m |  |
| Shielded wire | 100 m |  |

## Dimensions



The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1TNE 968 902 R1201 | AO561, analog output module, 2 AO, <br> U/I | Active |
| 1TNE 968 901 R3102 | Terminal block TA563-11, 11 pins, <br> screw front, cable side, 6 pieces per <br> unit | Active |
| 1TNE 968 901 R3104 | Terminal block TA564-11, 11 pins, <br> screw front, cable front, 6 pieces per <br> unit | Active |
| 1TNE 968 901 R3106 | Terminal block TA565-11, 11 pins, <br> spring front, cable front, 6 pieces per <br> unit | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

## AX561 - Analog input/output module

## Features

- 4 configurable analog inputs (IO ... 13 ) in 1 group
- 2 configurable analog outputs ( O 0 and O 1 ) in 1 group
- Resolution: 12 bits including sign


1 I/O bus
21 green LED to display power supply, 1 red LED to display error
3 Terminal number
4 Allocation of signal name
5 Terminal block for input signals (9-pin)
6 Terminal block for output signals (11-pin)
72 holes for wall-mounting with screws
8 DIN rail

## Intended purpose

The device can be used as a decentralized I/O extension module for S 500 communication interface modules (e. g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs.

The inputs are not galvanically isolated from each other.
The outputs are not galvanically isolated from each other.
All other circuitry of the module is not galvanically isolated from the inputs/outputs or from the I/O bus.

The I/O module must not be used as a decentralized I/O module with CI590-CS31-HA communication interface modules.

## Functionality

4 analog inputs, individually configurable for

- Not used (default)
- $-2.5 \mathrm{~V} . .+2.5 \mathrm{~V}$
- $-5 \mathrm{~V} \ldots+5 \mathrm{~V}$
- $0 \mathrm{~V} \ldots+5 \mathrm{~V}$
- $0 \mathrm{~V} \ldots+10 \mathrm{~V}$
- 0 mA ... 20 mA
- $4 \mathrm{~mA} . .20 \mathrm{~mA}$

2 analog outputs, individually configurable for

- Not used (default)
- -10 V ... +10 V
- 0 mA ... 20 mA
- 4 mA ... 20 mA

| Parameter | Value |
| :---: | :---: |
| Resolution of the analog channels |  |
| Voltage bipolar (-2.5 V ... +2.5 V; $-5 \vee \ldots+5 \vee)$ | 12 bits including sign |
| Voltage unipolar ( $0 \mathrm{~V} \ldots 5 \mathrm{~V}$; $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ ) | 12 bits |
| Current (0 mA ... 20 mA ; 4 mA ... 20 mA ) | 12 bits |
| LED displays | 2 LEDs for process voltage and error messages |
| Internal supply | Via I/O bus |
| External supply | Via the terminals L+ (process voltage 24 V $D C$ ) and $M(0 \vee D C)$; the $M$ terminal is connected to the M terminal of the CPU via the I/O bus |

## Connections

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

If the output is configured as not used, the voltage and current output signals are undefined and must not be connected.

The connection is carried out by using a removable 9-pin and 11-pin terminal block. These terminal blocks differ in their connection system (spring terminals or screw terminals, cable mounting from the front or from the side). The terminal blocks are not included in the module's scope of delivery and must be ordered separately.


Fig. 87: Internal construction of the analog inputs and outputs

Table 148: Assignment of the terminals

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 1 | R0 | Burden resistor for input signal 0 for current sensing |
| 2 | I0+ | Positive pole of input signal 0 |
| 3 | I0- | Negative pole of input signal 0 |
| 4 | R1 | Burden resistor for input signal 1 for current sensing |
| 5 | I1+ | Positive pole of input signal 1 |
| 6 | I1- | Negative pole of input signal 1 |


| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 7 | R2 | Burden resistor for input signal 2 for current sensing |
| 8 | I2+ | Positive pole of input signal 2 |
| 9 | I2- | Negative pole of input signal 2 |
| 10 | R3 | Burden resistor for input signal 3 for current sensing |
| 11 | I3+ | Positive pole of input signal 3 |
| 12 | I3- | Negative pole of input signal 3 |
| 13 | O0I+ | Voltage output of channel 0 |
| 14 | O1U+ | Current output of channel 0 |
| 15 | O1I+ | Voltage output of channel 1 |
| 16 | SG | Current output of channel 1 |
| 17 | L+ | Negative pole of channels O0 and O1 |
| 18 | M | Shield grounding |
| 19 | Process voltage L+ (24 V DC) |  |
| 20 | Process voltage M (0 V DC) |  |

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a communication interface module or a CPU). Thus, the current consumption from 24 V DC power supply at the terminals UP/L+ and ZP/M of the CPU/communication interface module increases by 5 mA per AX561.
The external power supply connection is carried out via the $\mathrm{L}+(+24 \mathrm{~V} D C$ ) and the M ( 0 V DC) terminals. The M terminal is interconnected to the M/ZP terminal of the CPU/communication interface module.

## NOTICE!

Risk of imprecise and faulty measurements!
Analog signals may be distorted seriously by external electromagnetic influences.
Use shielded wires when wiring analog signal sources. The cable shield must be grounded at both ends of the cable. Provide a potential equalisation of a low resistance to avoid high potential differences between different parts of the plant.

WARNING!

## Removal/Insertion under power

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.
Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.
The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

Risk of damaging the PLC modules!
Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

The module provides several diagnosis functions ${ }^{\star}>$ Chapter 5.2.4.4.1.5.7 "Diagnosis" on page 665.


Fig. 88: Example of the internal construction of the analog input AIO (analog inputs Al1 ... Al3 are designed in the same way)

## CAUTION!

## Risk of damaging the analog input!

The $250 \Omega$ input resistor can be damaged by overcurrent.
Make sure that the current through the resistor never exceeds 30 mA .

The following figures are an example of the connection of analog sensors (voltage) to the input IO of the analog input/output module AX561. Proceed with the inputs I1 ... I3 in the same way.

Table 149: Example of the connection of analog sensors (voltage) to the input IO of the analog input/output module AX561 (Proceed with the inputs I1 to I3 in the same way)


| Connection of active-type analog sensors <br> (voltage) | Connection of passive-type analog sen- <br> sors (voltage) |
| :--- | :--- |
| $-2.5 \mathrm{~V} \ldots 2.5 \mathrm{~V}$ | $-2.5 \mathrm{~V} \ldots 2.5 \mathrm{~V}$ |
| $-5 \mathrm{~V} \ldots 5 \mathrm{~V}$ | $-5 \mathrm{~V} \ldots 5 \mathrm{~V}$ |
| $0 \mathrm{~V} \ldots 5 \mathrm{~V}$ | $0 \mathrm{~V} \ldots 5 \mathrm{~V}$ |
| $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ | $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ |

Table 150: Example of the connection of analog sensors (current) to the input 10 of the analog input/output module AX561 (Proceed with the inputs I1 ... I3 in the same way)


| Connection of active-type analog sensors <br> (voltage) | Connection of passive-type analog sen- <br> sors (voltage) |
| :--- | :--- |
| $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ |
| $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ |  |

Table 151: Example of the connection of analog actuators to the analog input/output module AX561


The output signal is undefined if the supply voltage at the $L+$ terminal is below 10 V . This can, for example, occur if the supply voltage has a slow ramp-up / ramp-down behavior and must be foreseen when planning the installation.

If the output is configured in current mode, the voltage output signal is undefined and must not be connected.

If the output is configured in voltage mode, the current output signal is undefined and must not be connected.

The meaning of the LEDs is described in the displays chapter ${ }^{\$}$ Chapter 5.2.4.4.1.5.8 "State LEDs" on page 666.

## I/O configuration

The I/O module does not store configuration data itself.

## Parameterization

The arrangement of the parameter data is performed with Automation Builder software.
The parameter data directly influences the functionality of modules.
For non-standard applications, it is necessary to adapt the parameters to your system configuration.
$\left.\begin{array}{|l|l|l|l|l|l|l|l|}\hline \text { Name } & \text { Value } & \begin{array}{l}\text { Internal } \\ \text { Value }\end{array} & \begin{array}{l}\text { Internal } \\ \text { value, } \\ \text { Type }\end{array} & \text { Default } & \text { Min. } & \text { Max. } & \begin{array}{l}\text { EDS SIot } \\ \text { Index }\end{array} \\ \hline \text { Module ID } & \text { Internal } & 6520^{1} \text { ) } & \text { WORD } & 0 \times 1978 & 0 & 65535 & \text { xx01 } \\ \hline \begin{array}{l}\text { Ignore } \\ \text { module }\end{array} & \begin{array}{l}\text { No } \\ \text { Yes }\end{array} & 0 & \text { BYTE } \\ 1 & \text { No } \\ 0 x 00\end{array}\right]$
${ }^{1}$ ) With CS31 and addresses less than 70, the value is increased by 1
${ }^{2}$ ) Value is hexadecimal: HighByte is slot ( $x x: 0 \ldots 7$ ), LowByte is index ( $1 \ldots n$ ) GSD file:

| Ext_User_Prm_Data_Len $=$ | $0 x 0 B$ |
| :--- | :--- |
| Ext_User_Prm_Data_Const $(0)=$ | $0 x 79,0 \times 19,0 \times 08,1$ |
|  | $0 \times 01,0 \times 00,1$ |
|  | $0 \times 00,0 \times 00,0 \times 00,0 \times 00,1$ |
|  | $0 x 00,0 \times 00 ;$ |

## Input channel (4x)

| Name | Value | Internal <br> value | Internal <br> value, Type | Default | Min. | Max. |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Channel <br> configura- <br> tion | see table ${ }^{2}$ ) | see table ${ }^{2}$ ) | BYTE | 0 <br> $0 \times 00$ see <br> table $\left.^{2}\right)$ | 0 | 65535 |

Table 152: Channel configuration ${ }^{2}$ )

| Internal value | Operating modes for the analog inputs, individually configu- <br> rable |
| :--- | :--- |
| 0 | Not used (default) |
| 1 | $0 \mathrm{~V} \ldots+10 \mathrm{~V}$ |
| 3 | $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ |
| 4 | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ |
| 6 | $0 \mathrm{~V} \ldots+5 \mathrm{~V}$ |
| 7 | $-5 \mathrm{~V} \ldots+5 \mathrm{~V}$ |
| 20 | $-2.5 \mathrm{~V} \ldots+2.5 \mathrm{~V}$ |

Output channel (2x)

| Name | Value | Internal <br> value | Internal <br> value, Type | Default | Min. | Max. |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Channel <br> configura- <br> tion | see see <br> table $^{2}$ ) | see see <br> table ${ }^{2}$ ) | BYTE | 0 <br> $0 \times 00$ see <br> table $\left.^{2}\right)$ | 0 | 65535 |

Table 153: Channel configuration ${ }^{2}$ )

| Internal value | Operating modes for the analog outputs, individually configurable |
| :--- | :--- |
| 0 | Not used (default) |
| 128 | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ |
| 129 | $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ |
| 130 | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ |

## Diagnosis

| E1 ... E4 | d1 | d2 | d3 | d4 | $\begin{array}{\|l\|} \hline \text { Identifier } \\ 000 \ldots 063 \end{array}$ | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err |  |  |
| Byte 6 <br> Bit 6 ... 7 | - | Byte 3 | Byte 4 | Byte 5 | Byte 6 <br> Bit 0 ... 5 |  |  |
| Class | Interface | Device | Module | Channel | Error Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | $\left.{ }^{3}\right)$ | ${ }^{4}$ ) |  |  |  |
| Module error |  |  |  |  |  |  |  |
| 3 | 14 | $1 . .10$ | 31 | 31 | 19 | Checksum error in the I/O module | Replace I/O module |
|  | 11/12 | ADR | $1 \ldots 10$ |  |  |  |  |
| 3 | 14 | $1 \ldots 10$ | 31 | 31 | 9 | Overflow diagnosis buffer | Restart |
|  | 11/12 | ADR | 1... 10 |  |  |  |  |
| 3 | 14 | $1 \ldots 10$ | 31 | 31 | 26 | Parameter error | Check master |
|  | 11 / 12 | ADR | 1... 10 |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 11 | Process voltage too low | Check process voltage |
|  | 11/12 | ADR | $1 \ldots 10$ |  |  |  |  |
| Channel error |  |  |  |  |  |  |  |
| 4 | 14 | $1 \ldots 10$ | 1 | $0 \ldots 3$ | 48 | Analog value overflow at an analog input | Check input value or terminal |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |
| 4 | 14 | $1 \ldots 10$ | 1 | $0 \ldots 3$ | 7 | Analog value underflow at an analog input | Check input value |
|  | 11 / 12 | ADR | 1... 10 |  |  |  |  |


| E1 ... E4 | d1 | d2 | d3 | d4 | $\begin{array}{\|l\|} \hline \text { Identifier } \\ 000 \ldots 063 \end{array}$ | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err |  |  |
| Byte 6 <br> Bit 6 ... 7 | - | Byte 3 | Byte 4 | Byte 5 | Byte 6 <br> Bit 0 ... 5 |  |  |
| Class | Interface | Device | Module | Channel | Error Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | $\left.{ }^{3}\right)$ | $\left.{ }^{4}\right)$ |  |  |  |
| 4 | 14 | $1 \ldots 10$ | 3 | $0 \ldots 1$ | 48 | Analog value overflow at an analog output | Check output value or terminal |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |
| 4 | 14 | $1 \ldots 10$ | 3 | $0 \ldots 1$ | 7 | Analog value underflow at an analog output | Check output value |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |

Remarks:

| $\left.{ }^{1}\right)$ | In AC500 the following interface identifier applies: 14 = I/O bus, 11 = COM1 (e.g. CS31 bus), 12 = COM2. <br> The PNIO diagnosis block does not contain this identifier. |
| :---: | :---: |
| ${ }^{2}$ ) | With "Device" the following allocation applies: <br> 31 = module itself, 1 ... $10=$ expansion module 1 ... 10, ADR = hardware address (e. g. of the DC551-CS31) |
| ${ }^{3}$ ) | With "Module" the following allocation applies dependent of the master: <br> Module error: I/O bus or PNIO: 31 = module itself; COM1/COM2: 1 ... $10=$ expansion 1 ... 10 <br> Channel error: I/O bus or PNIO = module type ( 1 = AI, 3 = AO); COM1/ COM2: 1 ... 10 = expansion 1 ... 10 |
| ${ }^{4}$ ) | In case of module errors, with channel "31 = Module itself" is output. |

## State LEDs

| LED |  | State | Color | LED = OFF | LED = ON | LED flashes |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | PWR | Process voltage 24 V DC via terminal | Green | CPU module voltage or external 24 V DC supply voltage is missing | 3.3 V system voltage (l/O bus) and external 24 V DC supply voltage are present | --- |
|  | ERR | Channel or module error | Red | No error or process voltage is missing | Severe error in the module | Error on 1 or more channels of the module |

## CAUTION!

Risk of wrong analog input values!
The analog input values may be wrong if the measuring range of the inputs are exceeded.
Make sure that the analog signal at the connection terminals is always within the signal range.

| Range | $-2.5 \mathrm{~V} . .$ | $-5 \text { V ... }$ | $0 \mathrm{~V} \ldots 5$ | $0 \mathrm{~V} . .10$ | $0 \mathrm{~mA} \ldots$ | $4 \mathrm{~mA} \ldots$ | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  |  | Decimal | Hex. |
| Overflow | >2.9397 | >5.8795 | >5.8795 | $\begin{aligned} & >11.758 \\ & 9 \end{aligned}$ | $\begin{aligned} & >23.517 \\ & 8 \end{aligned}$ | $\begin{aligned} & >22.814 \\ & 2 \end{aligned}$ | 32767 | 7FFF |
| Measured value too high | $\begin{aligned} & 2.9397 \\ & : \\ & 2.5014 \end{aligned}$ | $\begin{aligned} & 5.8795 \\ & : \\ & 5.0029 \end{aligned}$ | $\begin{aligned} & 5.8795 \\ & : \\ & : \\ & : \\ & 5.0015 \end{aligned}$ | $\begin{aligned} & 11.7589 \\ & : \\ & : \\ & : \\ & 10.0029 \end{aligned}$ | $\begin{aligned} & 23.5178 \\ & : \\ & : \\ & : \\ & 20.0058 \end{aligned}$ | $\begin{aligned} & 22.8142 \\ & : \\ & : \\ & 20.0058 \end{aligned}$ | $\begin{aligned} & 32511 \\ & : \\ & 27664 \\ & 27658 \\ & 27656 \end{aligned}$ | 7EFF <br> 6C10 <br> 6C0A <br> 6C08 |
| Normal range <br> Normal range or measured value too low | $\begin{aligned} & 2.5000 \\ & : \\ & 0.0014 \end{aligned}$ | $\begin{aligned} & 5.0000 \\ & : \\ & 0.0029 \end{aligned}$ | $\begin{aligned} & 5.0000 \\ & : \\ & : \\ & : \\ & 0.0015 \end{aligned}$ | $\begin{aligned} & 10.0000 \\ & : \\ & : \\ & : \\ & 0.0029 \end{aligned}$ | $\begin{aligned} & 20.0000 \\ & : \\ & : \\ & : \\ & 0.0058 \end{aligned}$ | $\begin{aligned} & 20.0000 \\ & : \\ & : \\ & 4.0058 \end{aligned}$ | $\begin{aligned} & 27648 \\ & : \\ & 16 \\ & 10 \\ & 8 \end{aligned}$ | $\begin{aligned} & 6 \mathrm{C} 00 \\ & : \\ & 0010 \\ & 000 \mathrm{~A} \\ & 0008 \end{aligned}$ |
|  | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0 | 4 | 0 | 0000 |
|  | -0.0014 \|-2.5000 | -0.0029 -5.0000 |  |  |  | $\begin{array}{\|l} \hline 3.9942 \\ : \\ : \\ 0 \end{array}$ | $\begin{array}{\|l} \hline-10 \\ -16 \\ -4864 \\ -6912 \\ : \\ -27648 \end{array}$ | FFF6 <br> FFF0 <br> ED00 <br> E500 <br> 9400 |
| Measured value too low | $\begin{aligned} & -2.5014 \\ & : \\ & -2.9398 \end{aligned}$ | $\begin{aligned} & -5.0029 \\ & : \\ & -5.8795 \end{aligned}$ |  |  |  |  | $\begin{aligned} & -27664 \\ & : \\ & -32512 \end{aligned}$ | $\begin{aligned} & \text { 93F0 } \\ & : \\ & 8100 \end{aligned}$ |
| Underflow | <-2.9398 | <-5.8795 | <-0.0300 | <-0.0600 | <-0.1200 | <-0.1200 | -32768 | 8000 |

The represented resolution corresponds to 12 bits including sign.

## Output ranges

| Range | -10 V ... +10 V | $\begin{aligned} & 0 \mathrm{~mA} . . .20 \\ & \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 4 \mathrm{~mA} \ldots 20 \\ & \mathrm{~mA} \end{aligned}$ | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  | Decimal | Hex. |
| Overflow | > 11.7589 | > 23.5178 | > 22.8142 | 32767 | 7FFF |
| Output value too high | $\begin{aligned} & 11.7589 \\ & : \\ & 10.0058 \end{aligned}$ | $\begin{aligned} & 23.5178 \\ & : \\ & : \\ & : \\ & 20.0058 \end{aligned}$ | $\begin{aligned} & 22.8142 \\ & : \\ & : \\ & 20.0058 \end{aligned}$ | $\begin{aligned} & 32511 \\ & : \\ & 27664 \\ & 27658 \\ & 27656 \end{aligned}$ | $\begin{aligned} & \hline 7 \mathrm{EFF} \\ & : \\ & 6 \mathrm{C} 10 \\ & 6 \mathrm{C} 0 \mathrm{~A} \\ & 6 \mathrm{C} 08 \end{aligned}$ |
| Normal range <br> Normal range or output value too low | $\begin{aligned} & 10.0000 \\ & : \\ & 0.0058 \end{aligned}$ | $\begin{aligned} & 20,0000 \\ & : \\ & : \\ & 0.0058 \end{aligned}$ | $\begin{aligned} & 20.0000 \\ & : \\ & : \\ & 4.0058 \end{aligned}$ | $\begin{aligned} & 27648 \\ & : \\ & 16 \\ & 10 \\ & 8 \end{aligned}$ | $\begin{aligned} & \text { 6C00 } \\ & : \\ & 0010 \\ & 000 \mathrm{~A} \\ & 0008 \end{aligned}$ |
|  | 0.0000 | 0 | 4 | 0 | 0000 |
|  | $\begin{aligned} & -0.0058 \\ & : \\ & : \\ & : \\ & -10.0000 \end{aligned}$ |  | 3.9942 $:$ $:$ 0 | $\begin{aligned} & \hline-10 \\ & -16 \\ & -4864 \\ & -6912 \\ & : \\ & -27648 \\ & \hline \end{aligned}$ | FFF6 FFF0 ED00 E500 $:$ 9400 |
| Output value too low | $\begin{array}{\|l} -10.0058 \\ : \\ -11.7589 \end{array}$ |  |  | $\begin{aligned} & -27664 \\ & : \\ & -32512 \end{aligned}$ | $\begin{aligned} & 93 F 0 \\ & : \\ & 8100 \end{aligned}$ |
| Underflow | <-11.7589 |  | <0.0000 | -32768 | 8000 |

The represented resolution corresponds to 12 bits including sign.

## Technical data

## Technical data of the module

The system data of AC500-eCo apply.
«4 Chapter 5.1.1 "System data AC500-eCo" on page 159
Only additional details are therefore documented below.

| Parameter |  | Value |
| :--- | :--- | :--- |
| Process supply voltage L+ |  |  |
|  | Connections | Terminal 19 for L+ (+24 V DC) and terminal 20 <br> for M (0 V) |
|  | Rated value | 24 V DC |
|  | Current consumption via L+ terminal | $0.14 \mathrm{~A}+$ output load |
|  | Inrush current (at power-up) | 0.05 A |


| Parameter |  | Value |
| :--- | :--- | :--- |
|  | Max. ripple | $5 \%$ |
|  | Protection against reversed voltage | Yes |
|  | Protection fuse for L+ | Recommended |
| Current consumption from 24 V DC power <br> supply at the terminals UP/L+ and ZP/M of <br> the CPU/communication interface module | Ca. 5 mA |  |
| Galvanic isolation | No |  |
| Surge-voltage (max.) | 35 V DC for 0.5 s |  |
| Max. power dissipation within the module | 4.9 W |  |
| Weight | Ca. 120 g |  |
| Mounting position | Horizontal or vertical |  |
| Cooling | The natural convection cooling must not be hin- <br> dered by cable ducts or other parts in the control <br> cabinet. |  |

## NOTICE!

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and temporary overvoltage up to 30 V DC.

## Technical data of the analog inputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per <br> module | 4 individually configurable voltage or current inputs |
| Distribution of channels into <br> groups | 1 (4 channels per group) |
| Resolution | Unipolar |
| Voltage: $0 \mathrm{~V} \ldots+5 \mathrm{~V} ; 0 \mathrm{~V} \ldots+10 \mathrm{~V}: 12$ bits <br> Current $0 \mathrm{~mA} \ldots 20 \mathrm{~mA} ; 4 \mathrm{~mA} \ldots 20 \mathrm{~mA}: 12$ bits |  |
| Connection of the signals I0- to <br> I3- | Terminals $3,6,9,12$ |
| Connection of the signals I0+ to <br> I3+ | Terminals $2,5,8,11$ |
| Input type | Differential |
| Galvanic isolation | No galvanic isolation between the inputs and the I/O bus |
| Common mode input range | Signal voltage plus common mode voltage must be within $\pm$ <br> 12 V |
| Indication of the input signals | No |
| Channel input resistance | Voltage: $>12$ bits including sign |
|  | Current: ca. $250 \Omega$ |


| Parameter | Value |  |
| :---: | :---: | :---: |
| Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range | Typ. | $\begin{aligned} & \pm 0.5 \% \text { of full scale (voltage) } \\ & \pm 0.5 \% \text { of full scale (current } 0 \mathrm{~mA} \ldots 20 \mathrm{~mA} \text { ) } \\ & \pm 0.7 \% \text { of full scale (current } 4 \mathrm{~mA} \ldots 20 \mathrm{~mA} \text { ) } \\ & \text { at }+25^{\circ} \mathrm{C} \end{aligned}$ |
|  | Max. | $\pm 2$ \% of full scale (all ranges) at $0^{\circ} \mathrm{C} \ldots+60^{\circ} \mathrm{C}$ or EMC disturbance |
| Time constant of the input filter | Voltage: $300 \mu \mathrm{~s}$ Current: $300 \mu \mathrm{~s}$ |  |
| Relationship between input signal and hex code | $\stackrel{\text { ® Table on page } 667}{ }$ |  |
| Analog to digital conversion time | Typ. $500 \mu$ s per channel |  |
| Unused inputs | Can be left open and should be configured as "unused" |  |
| Input data length | 8 bytes |  |
| Overvoltage protection | Yes, up to 30 V DC only for voltage input |  |
| Max. cable length (conductor cross section > $0.14 \mathrm{~mm}^{2}$ ) |  |  |
| Unshielded wire | 10 m |  |
| Shielded wire | 100 m |  |

## Technical data of the analog outputs

| Parameter | Value |  |
| :---: | :---: | :---: |
| Number of channels per module | 2 configurable voltage or current outputs |  |
| Distribution of channels into groups | 1 (2 channels per group) |  |
| Connection of the signals O0U- and O1U+ | Terminals 13 and 15 |  |
| Connection of the signals $\mathrm{O} 01+$ and O11+ | Terminals 14 and 16 |  |
| Output type | Bipolar with voltage, unipolar with current |  |
| Resolution | 12 bits including sign |  |
| Indication of the output signals | No |  |
| Output resistance (load) as current output | $0 \Omega \ldots 500 \Omega$ |  |
| Output load ability as voltage output | 2 mA max. |  |
| Relationship between input signal and hex code | Table Output Ranges $\stackrel{y}{*}$ Table on page 668 |  |
| Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range | Typ. | ```\pm0.5 % of full scale (voltage) \pm0.5 % of full scale (current 0mA ... }20\textrm{mA} \pm0.7 % of full scale (current 4 mA ... 20 mA) at +25*}\textrm{C``` |
|  | Max. | $\pm 2 \%$ of full scale (all ranges) at $0^{\circ} \mathrm{C} \ldots+60^{\circ} \mathrm{C}$ or EMC disturbance |


| Parameter | Value |
| :--- | :--- |
| Unused outputs | Can be left open and should be configured <br> as "unused" |
| Output data length | 4 bytes |
| Overvoltage protection | Yes, up to 30 V DC |
| Max. cable length (conductor cross section <br> $>0.14 \mathrm{~mm}^{2}$ ) |  |
|  | Unshielded wire |
| Shielded wire | 10 m |

## Dimensions



The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1TNE 968 902 R1301 | AX561, analog input/output module, <br> 4 AI, 2 AO, U/l | Active |
| 1TNE 968 901 R3101 | Terminal block TA563-9, 9 pins, screw <br> front, cable side, 6 pieces per unit | Active |
| 1TNE 968 901 R3102 | Terminal block TA563-11, 11 pins, <br> screw front, cable side, 6 pieces per <br> unit | Active |


| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1TNE 968 901 R3103 | Terminal block TA564-9, 9 pins, screw <br> front, cable front, 6 pieces per unit | Active |
| 1TNE 968 901 R3104 | Terminal block TA564-11, 11 pins, <br> screw front, cable front, 6 pieces per <br> unit | Active |
| 1TNE 968 901 R3105 | Terminal block TA565-9, 9 pins, spring <br> front, cable front, 6 pieces per unit | Active |
| 1TNE 968 901 R3106 | Terminal block TA565-11, 11 pins, <br> spring front, cable front, 6 pieces per <br> unit | Active |

> *) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.4.4.2 S500 and S500-XC

## AC522 - Analog input/output module

## Features

- 8 channels configurable as analog inputs/outputs in one group (2.0 ... 2.7 and 3.0 ... 3.7)
- Resolution 12 bits including sign
- Module-wise galvanically isolated
- XC version for use in extreme ambient conditions available


1 I/O bus
2 Allocation between terminal number and signal name
38 yellow LEDs to display the signal states at the analog inputs/outputs (C0 ... C7)
41 green LED to display the state of the process supply voltage UP
51 red LED to display errors
6 Label
7 Terminal unit
8 DIN rail
Sign for XC version

## Intended purpose

The device can be used as a decentralized I/O extension module for S500 communication interface modules (e. g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs.

The configuration is performed by software. The modules are supplied with a process voltage of 24 V DC.

The inputs and outputs are galvanically isolated from all other circuitry of the module.

## Functionality

> If used as inputs, the following signal ranges are individually configurable:

- Unused (default setting)
- 0 V ... 10 V
- $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$
- $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$
- $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$
- Pt100, $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ (2-wire)
- Pt100, $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ (3-wire), requires 2 channels
- Pt100, $-50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ (2-wire)
- Pt100, $-50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ (3-wire), requires 2 channels
- Pt1000, $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ (2-wire)
- Pt1000, $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ (3-wire), requires 2 channels
- Ni1000, $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ (2-wire)
- Ni1000, $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ (3-wire), requires 2 channels
- 0 V ... 10 V with differential inputs, requires 2 channels
- $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ with differential inputs, requires 2 channels
- Digital signals (digital input)

| Parameter | Value |
| :---: | :---: |
| Resolution of the analog channels |  |
| Voltage -10 V ... +10 V | 12 bits including sign |
| Voltage 0 V ... 10 V | 12 bits |
| $\begin{aligned} & \text { Current } 0 \mathrm{~mA} \ldots 20 \mathrm{~mA}, \\ & 4 \mathrm{~mA} \ldots 20 \mathrm{~mA} \end{aligned}$ | 12 bits |
| Temperature | $+0.1^{\circ} \mathrm{C}$ |
| LED displays | 10 LEDs for signals and error messages |
| Internal power supply | Via the I/O bus interface (I/O bus) |
| External power supply | Via the terminals ZP and UP (process voltage 24 V DC) |
| Required terminal unit | TU515 or TU516 \& Chapter 5.2.5.2 "TU515, TU516, TU541 and TU542 for I/O modules" on page 938 |

## Connections

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

The modules are plugged on an I/O terminal unit $\Leftrightarrow$ Chapter 5.2.5.2 "TU515, TU516, TU541 and TU542 for I/O modules" on page 938. Properly position the modules and press until they lock in place. The terminal units are mounted on a DIN rail or with 2 screws plus the additional accessory for wall mounting ${ }^{\star}$ ch Chapter 5.2.8.2.5 "TA526-Wall mounting accessory" on page 1324.

The connection of the I/O channels is carried out using the 40 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

The terminals $1.8,2.8,3.8$ and 4.8 as well as $1.9,2.9,3.9$ and 4.9 are electrically interconnected within the I/O terminal units and always have the same assignment, independent of the inserted module:

Terminals 1.8, 2.8, 3.8 and 4.8: process voltage UP $=+24 \mathrm{~V}$ DC
Terminals 1.9, 2.9, 3.9 and 4.9: process voltage $\mathrm{ZP}=0 \mathrm{~V}$ DC
The assignment of the other terminals:

| Terminals | Signal | Description |
| :--- | :--- | :--- |
| $1.0 \ldots 1.7$ | Unused | Unused |
| $2.0 \ldots 2.7$ | C0- ... C7- | Negative poles of the 8 analog <br> inputs/outputs |
| $3.0 \ldots 3.7$ | C0+ .. C7+ | Positive poles of the analog <br> inputs/outputs |
| $4.0 \ldots 4.7$ | Unused | Unused |

The negative poles of the analog inputs are connected to each other to form an "Analog Ground" signal for the module.

The negative poles of the analog outputs are connected to each other to form an "Analog Ground" signal for the module.

There is no galvanic isolation between the analog circuitry and ZP/UP. Therefore, the analog sensors must be galvanically isolated in order to avoid loops via the ground potential or the supply voltage.

Because of their common reference potential, analog current inputs cannot be circuited in series, neither within the module nor with channels of other modules.

For the open-circuit detection (wire break), each analog input channel is pulled up to "plus" by a high-resistance resistor. If nothing is connected, the maximum voltage will be read in then.

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a communication interface module or a CPU). Thus, the current consumption from 24 V DC power supply at the terminals L+/UP and M/ZP of the CPU/communication interface module increases by 2 mA per I/O module. The external power supply connection is carried out via the UP (+24 V DC) and the $\mathrm{ZP}(0 \mathrm{~V} \mathrm{DC})$ terminals.

## WARNING!

## Removal/Insertion under power

Removal or insertion under power is permissible only if all conditions for hot swapping are fullfilled.
« Chapter 4.9.3 "Replace an I/O module with hot swap" on page 153
The devices are not designed for removal or insertion under power when the conditions for hot swap do not apply. Because of unforeseeable consequences, it is not allowed to plug in or unplug devices with the power being ON.
Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while they are energized in a hazardous location could result in an electric arc, which could create an ignition source resulting in fire or explosion.

Prior to proceeding, make sure that power is been disconnected and that the area has been thoroughly checked to ensure that flammable materials are not present.

The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

## Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

Generally, analog signals must be laid in shielded cables. The cable shields must be grounded at both sides of the cables. In order to avoid unacceptable potential differences between different parts of the installation, low resistance equipotential bonding conductors must be laid.


Fig. 89: Connection of the I/O module
14 analog I/O channels as inputs for 0 V ... $10 \mathrm{~V},-10 \mathrm{~V} \ldots+10 \mathrm{~V}, 0 \mathrm{~mA} . .22 \mathrm{~mA}, 4 \mathrm{~mA} . . .20 \mathrm{~mA}, \mathrm{Pt} 100 / \mathrm{Pt} 1000 /$ Ni1000 digital signals as outputs for -10 V ... $+10 \mathrm{~V}, 0 \mathrm{~mA} . . .20 \mathrm{~mA}, 4 \mathrm{~mA} . . .20 \mathrm{~mA}$
24 analog I/O channels as inputs for $0 \mathrm{~V} \ldots 10 \mathrm{~V},-10 \mathrm{~V} \ldots+10 \mathrm{~V}, 0 \mathrm{~mA} \ldots 20 \mathrm{~mA}, 4 \mathrm{~mA} . . .20 \mathrm{~mA}, \mathrm{Pt} 100 / \mathrm{Pt} 1000 /$ Ni1000 digital signals as outputs for $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$

The process voltage must be included in the grounding concept of the control system (e.g. grounding the negative pole).

By installing equipotential bonding conductors between the different parts of the system, it must be made ensured that the potential difference between ZP and AGND never exeeds 1 V .

## Connection of resistance thermometers in 2-wire configuration

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the I/O module provides a constant current source which is multiplexed over the 8 analog channels.


| Pt100 | $-50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ | 2-wire configuration, one <br> channel used |
| :--- | :--- | :--- |
| $\operatorname{Pt100}$ | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 2-wire configuration, one <br> channel used |
| Pt1000 | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 2-wire configuration, one <br> channel used |
| Ni1000 | $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ | 2-wire configuration, one <br> channel used |

The I/O module performs a linearization of the resistance characteristic.
In order to avoid error messages from unused analog input channels, it is useful to configure them as "unused".

## Connection of resistance thermometers in 3-wire configuration

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the I/O module provides a constant current source which is multiplexed over the max. 8 (depending on the configuration) analog channels.


1 Return line
2 Twisted pair within the cable

If several measuring points are adjacent to each other, only one return line is necessary. This saves wiring costs.

With the 3-wire configuration, two adjacent analog channels belong together (e.g. the channels 0 and 1). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0 ), the next higher address must be the odd address (channel 1).

The constant current of one channel flows through the resistance thermometer. The constant current of the other channel flows through one of the cores. The module calculates the measured value from the two voltage drops and stores it under the input with the higher channel number (e.g. C1).
In order to keep measuring errors as small as possible, it is necessary to have all the involved conductors in the same cable. All the conductors must have the same cross section.

| Pt100 | $-50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ | 3-wire configuration, two <br> channels used |
| :--- | :--- | :--- |
| Pt100 | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 3-wire configuration, two <br> channels used |
| Pt1000 | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 3-wire configuration, two <br> channels used |
| Ni1000 | $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ | 3-wire configuration, two <br> channels used |

The I/O module performs a linearization of the resistance characteristic.
In order to avoid error messages from unused analog input channels, it is useful to configure them as "unused".

## Connection of active-type analog sensors (Voltage) with galvanically isolated power supply



By connecting the sensor's negative pole of the output voltage to AGND, the galvanically isolated voltage source of the sensor is referred to ZP.

By connecting to AGND the galvanically isolated voltage source of the sensor is referred to ZP. The following measuring ranges can be configured:

| Voltage | $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ | 1 channel used |
| :--- | :--- | :--- |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | 1 channel used |

In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as "unused".

## Connection of active-type analog sensors (Current) with galvanically isolated power supply



The following measuring ranges can be configured:

| Current | $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 1 channel used |
| :--- | :--- | :--- |
| Current | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 1 channel used |

Unused input channels can be left open-circuited, because they are of low resistance.

## Connection of active-type analog sensors (Voltage) with no galvanically isolated power supply



## CAUTION!

The potential difference between AGND and ZP at the module must not be greater than 1 V , not even in case of long lines (see figure Terminal Assignment).

If AGND does not get connected to $Z P$, the sensor current flows to $Z P$ via the AGND line. The measuring signal is distorted, as a very small current flows through the voltage line. The total current through the PTC should not exceed 50 mA . This measuring method is therefore only suitable for short lines and small sensor currents. If there are bigger distances, the difference measuring method should be applied.

| Voltage | $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ | 1 channel used |
| :--- | :--- | :--- |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V} *)$ | 1 channel used |
| $*)$ if the sensor can provide this signal range |  |  |

In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as "unused".

## Connection of passive-type analog sensors (Current)



| Current | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 1 channel used |
| :--- | :--- | :--- |

## CAUTION!

If, during initialization, an analog current sensor supplies more than 25 mA for more than 1 second to an analog input, this input is switched off by the module (input protection). In such cases, it is recommended to protect the analog input by a 10 -volt Zener diode (in parallel to I+ and ZP). But, in general, sensors with fast initialization or without current peaks higher than 25 mA are preferrable.

Unused input channels can be left open-circuited because they are of low resistance.

## Connection of active-type analog sensors (Voltage) to differential inputs

Differential inputs are very useful if analog sensors are used which are remotely non-isolated (e.g. the minus terminal is remotely grounded).

The use of differential inputs helps to considerably increase the measuring accuracy and to avoid ground loops.

With differential input configurations, two adjacent analog channels belong together (e.g. the channels 0 and 1). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0 ), the next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1).
The analog value is calculated by subtraction of the input value with the higher address from the input value of the lower address.

The converted analog value is available at the odd channel (higher address).

## CAUTION!

The ground potential at the sensors must not have too large a potential difference with respect to ZP (max. $\pm 1 \mathrm{~V}$ within the full signal range). Otherwise, problems may occur concerning the common-mode input voltages of the involved analog inputs.


The negative pole of the sensor must be grounded next to the sensor.

| Voltage | $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ | with differential inputs, 2 chan- <br> nels used |
| :--- | :--- | :--- |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | with differential inputs, 2 chan- <br> nels used |

In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as "unused".

## Use of analog inputs as digital inputs

Several (or all) analog inputs can be configured as digital inputs. The inputs are not galvanically isolated against the other analog channels.


| Digital input | 24 V | 1 channel used |
| :--- | :--- | :--- |
| Effect of incorrect input ter- <br> minal connection |  | Wrong or no signal detected, <br> no damage up to 35 V |

## Connection of analog output loads (Voltage, current)



| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | Load max. $\pm 10 \mathrm{~mA}$ | 1 channel used |
| :--- | :--- | :--- | :--- |
| Current | $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | Load $0 \Omega \ldots 500 \Omega$ | 1 channel used |
| Current | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | Load $0 \Omega \ldots 500 \Omega$ | 1 channel used |

Only the channels $0 \ldots 3$ can be configured as current output ( $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ or $4 \mathrm{~mA} . .20 \mathrm{~mA}$ ).

Unused analog outputs can be left open-circuited.

Internal data exchange

| Analog inputs (words) | 8 |
| :--- | :--- |
| Analog outputs (words) | 8 |

## I/O configuration

The module does not store configuration data itself. The 8 configurable analog channels are defined as inputs or outputs by the configuration, i.e. each of the configurable channels can used as input or output (or re-readable output in case of voltage input/output).

When a channel is used as input, the corresponding output must be configured unused.
When a channel is used as output, the corresponding input must be configured unused.

| Firmware version | Configuration |
| :--- | :--- |
| Firmware version > V2.0.0 | The arrangement of the parameter data is per- <br> formed by Control Builder Plus/ Automation <br> Builder software. |

The parameter data directly influences the functionality of modules.
For non-standard applications, it is necessary to adapt the parameters to your system configuration.
Module: Module slot address: $\mathrm{Y}=1 \ldots 10$

| No. | Name | Value | Internal value | Internal <br> value, type | Default | Min. | Max. | EDS <br> Slot/ Index |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 1 | Module ID | Internal | $\begin{aligned} & \hline 1520 \\ & \left.1{ }^{1}\right) \end{aligned}$ | Word | $\begin{aligned} & \hline 1520 \\ & 0 \times 05 f 0 \end{aligned}$ | 0 | 65535 | 0x0Y01 |
| 2 | Ignore module ${ }^{2}$ ) | $\begin{aligned} & \text { No } \\ & \text { Yes } \end{aligned}$ | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | Byte | $\begin{array}{\|l} \hline \text { No } \\ 0 \times 00 \end{array}$ |  |  | not for FBP |
| 3 | Parameter length in bytes | Internal | 37 | Byte | $\begin{aligned} & \hline 37-\mathrm{CPU} \\ & 37-\mathrm{FBP} \end{aligned}$ | 0 | 255 | 0x0Y02 |
| 4 | Check supply | $\begin{aligned} & \text { Off } \\ & \text { On } \end{aligned}$ | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | Byte | $\begin{array}{\|l\|} \hline \text { On } \\ 0 \times 01 \end{array}$ | 0 | 1 | 0x0Y03 |
| 5 | Analog data format | Default | 0 | Byte | Default $0 \times 00$ |  |  | 0x0Y04 |
| 6 | Behaviour of outputs at com-munication errors | Off <br> Last value <br> Substitute value | $\begin{array}{\|l} \hline 0 \\ 1+(n * 5) \\ 2+(n * 5) \\ n \leq 2 \end{array}$ | Byte | $\begin{aligned} & \text { Off } \\ & 0 \times 00 \end{aligned}$ | 0 | 2 | 0x0Y05 |
| 7 | Channel configuration Input channel 0 | see table <br> Channel tion | onfigura- | Byte | Default $0 \times 00$ | 0 | 19 | 0x0Y06 |
| 8 | Channel monitoring Input channel 0 | see table <br> Channel | monitoring | Byte | Default $0 \times 00$ | 0 | 3 | 0x0Y07 |


| No. | Name | Value | Internal <br> value | Internal <br> value, type | Default | Min. | Max. | EDS <br> Slot Index |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $\begin{aligned} & \text { to } \\ & 22 \end{aligned}$ | Channel configuration and channel monitoring of the input channels 1 to 7 | see tables <br> channel configuration and channel monitoring |  | $\begin{aligned} & \hline \text { Byte } \\ & \text { Byte } \end{aligned}$ | $\begin{array}{\|l\|} \hline \text { Default } \\ 0 \times 00 \\ 0 \times 00 \end{array}$ |  | $\begin{aligned} & \hline 19 \\ & 3 \end{aligned}$ | 0x0Y08 to 0x0Y15 |
| 23 | Channel configuration <br> Output channel 0 | see table <br> Channel configuration |  | Byte | Default <br> 0x00 | 0 | 130 | 0x0Y16 |
| 24 | Channel monitoring <br> Output channel 0 | see table <br> Channel monitoring |  | Byte | Default 0x00 | 0 | 3 | 0x0Y17 |
| 25 | Substitute value Output channel 0 | only valid for output channel 0 | 0 ... 0xffff | Word | $\begin{array}{\|l\|l\|} \hline \text { Default } \\ 0 \times 0000 \end{array}$ | 0 | 65535 | 0x0Y18 |
| 26 to 31 | Channel configu- <br> ration <br> and <br> channel <br> moni- <br> toring of <br> the <br> output <br> channels <br> 1 to 3 | see tables channel configuration and channel monitoring |  | $\begin{array}{\|l\|l\|} \hline \text { Byte } \\ \text { Byte } \end{array}$ | $\begin{array}{\|l\|} \hline \text { Default } \\ 0 \times 00 \\ 0 \times 00 \end{array}$ | $0$ | $\begin{aligned} & 130 \\ & 3 \end{aligned}$ | 0x0Y19 <br> to <br> 0x0Y1E |
| 32 | Channel configuration <br> Output channel 4 | see table <br> Channel configuration |  | Byte | $\begin{aligned} & \text { Default } \\ & 0 \times 00 \end{aligned}$ | 0 | 128 | 0x0Y1F |


| No. | Name | Value | Internal <br> value | Internal <br> value, <br> type | Default | Min. | Max. | EDS <br> Slot/ <br> Index |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| 33 | Channel <br> moni- <br> toring <br> Output <br> channel <br> 4 | see table <br> Channel monitoring | Byte | Default <br> $0 \times 00$ | 0 | 3 | $0 \times 0 \mathrm{Y} 20$ |  |
| 34 <br> to <br> 39 | Channel <br> configu- <br> ration <br> and <br> channel <br> moni- <br> toring of <br> the <br> output <br> channels <br> 5 to 7 | see tables <br> channel configura- <br> tion and channel <br> monitoring | Byte <br> Byte | Default <br> $0 \times 00$ <br> $0 \times 00$ | 0 <br> 0 | 128 | $0 \times 0 Y 21$ <br> to <br> $0 \times 0 Y 26$ |  |

${ }^{1}$ ) With CS31 and addresses less than 70 and FBP, the value is increased by 1
${ }^{2}$ ) Not with FBP

GSD file:

| Ext_User_Prm_Data_Len $=$ | 40 |
| :--- | :--- |
| Ext_User_Prm_Data_Const $(0)=$ | $0 \times 05,0 \times f 1,0 \times 25,1$ |
|  | $0 \times 01,0 \times 00,0 \times 00,1$ |
|  | $0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00$, |
| $0 \times 00,1$ |  |
|  | $0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00$, |
| $0 \times 00,1$ |  |
|  | $0 \times 00,0 \times 00,0 \times 00,0 \times 00,1$ |
|  | $0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00,1$ |
|  | $0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00$, |
|  | $0 \times 00 ;$ |

Table 154: Input channel ( $8 x$ )

| No. | Name | Internal value, type | Default |
| :--- | :--- | :--- | :--- |
| 1 | Channel configuration <br> see table $\left.{ }^{2}\right)$ | Byte | 0 <br> $0 \times 00$ see table ${ }^{2}$ ) |
| 2 | Channel monitoring $^{\left.\text {see table }{ }^{3}\right)}$ | Byte | 0 <br> $0 \times 00$ see table $\left.{ }^{3}\right)$ |

Table 155: Channel configuration ${ }^{2}$ )

| Internal value | Operating modes of the analog inputs, individually configurable |
| :---: | :---: |
| 0 | Unused (default) |
| 1 | Analog input 0 V ... 10 V |
| 2 | Digital input |
| 3 | Analog input 0 mA ... 20 mA |
| 4 | Analog input $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ |
| 5 | Analog input -10 V ... +10 V |
| 8 | Analog input Pt100, $-50{ }^{\circ} \mathrm{C} \ldots+400{ }^{\circ} \mathrm{C}$ (2-wire) |
| 9 | Analog input Pt100, $-50^{\circ} \mathrm{C} \ldots+400{ }^{\circ} \mathrm{C}$ (3-wire), requires 2 channels *) |
| 10 | Analog input $0 \ldots 10 \mathrm{~V}$ via differential inputs, requires 2 channels *) |
| 11 | Analog input $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ via differential inputs, requires 2 channels *) |
| 14 | Analog input Pt100, $-50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ (2-wire) |
| 15 | Analog input Pt100, $-50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ (3-wire), requires 2 channels *) |
| 16 | Analog input Pt1000, $-50^{\circ} \mathrm{C} \ldots+400{ }^{\circ} \mathrm{C}$ (2-wire) |
| 17 | Analog input Pt1000, $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ (3-wire), requires 2 channels *) |
| 18 | Analog input Ni1000, $-50^{\circ} \mathrm{C} \ldots+150{ }^{\circ} \mathrm{C}$ (2-wire) |
| 19 | Analog input Ni1000, $-50^{\circ} \mathrm{C} \ldots+150{ }^{\circ} \mathrm{C}$ (3-wire), requires 2 channels *) |
|  | *) In the operating modes with 3-wire configuration or with differential inputs, two adjacent analog inputs belong together (e.g. the channels 0 and 1). In these cases, both channels are configured in the desired operating mode. The lower address must be the even address (channel 0). The next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1). |

Table 156: Channel monitoring ${ }^{3}$ )

| Internal value | Monitoring |
| :--- | :--- |
| 0 | Plausibility, open-circuit (broken wire) and short circuit |
| 1 | Open-circuit and short-circuit |
| 2 | Plausibility |
| 3 | No monitoring |

Table 157: Output channel 0 (1 channel)

| No. | Name | Value | Internal value | Internal <br> value, type | Default |
| :--- | :--- | :--- | :--- | :--- | :--- |
| 1 | Channel con- <br> figuration | see table ${ }^{4}$ ) | see table ${ }^{4}$ ) | Byte | see table $\left.{ }^{4}\right)$ |
| 2 | Channel mon- <br> itoring | see table ${ }^{5}$ ) | see table $\left.{ }^{5}\right)$ | Byte | see table $\left.{ }^{5}\right)$ |
| 3 | Substitute <br> value <br> see table $\left.{ }^{6}\right)$ | $0 \ldots 65535$ | $0 \ldots$ |  |  |
| $0 x f f f$ |  |  |  |  |  |$\quad$ Word $\quad 0$.

Table 158: Output channels 1 ... 7 (7x)

| No. | Name | Internal value, type | Default |
| :--- | :--- | :--- | :--- |
| 1 | Channel configura- <br> tion <br> see table $\left.{ }^{4}\right)$ | Byte | see table ${ }^{4}$ ) |
| 2 | Channel monitoring $\left.^{\text {see table }}{ }^{5}\right)$ | Byte | see table ${ }^{5}$ ) |

Table 159: Channel configuration ${ }^{4}$ )

| Internal value | Operating modes of the analog outputs, individually configurable |
| :--- | :--- |
| 0 | Unused (default) |
| 128 | Analog output $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ |
| 129 | Analog output $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ (not with the channels $4 \ldots 7$ ) |
| 130 | Analog output $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ (not with the channels $4 \ldots 7$ ) |

Table 160: Channel monitoring ${ }^{5}$ )

| Internal value | Monitoring |
| :--- | :--- |
| 0 | Plausibility, open circuit (broken wire) and short circuit (default) |
| 1 | Open-circuit (broken wire) and short-circuit |
| 2 | Plausibility |
| 3 | No monitoring |

Table 161: Substitute value ${ }^{6}$ )

| Intended behavior of output <br> channel when the control <br> system stops | Required setting of the <br> module parameter "Behav- <br> iour of outputs in case of a <br> communication error" | Required setting of the <br> channel parameter "Substi- <br> tute value" |
| :--- | :--- | :--- |
| Output OFF | Off | 0 |
| Last value | Last value | 0 |
| Substitute value | Off or last value | $1 \ldots 65535$ |

## Diagnosis

Table 162: Possible diagnosis of I/O channels

| Output range | Condition |  |
| :--- | :--- | :--- |
|  | Output value in the PLC <br> underflow | Output value in the PLC overflow |
| $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | Error identifier $=7$ | Error identifier $=4$ |
| $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ |  |  |
| $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ |  |  |


| Input range |  |  |  |  |
| :--- | :--- | :--- | :--- | :--- |
|  | Sondition | Wire break | Input value under- <br> flow | Input value over- <br> flow |
| $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | no diagnosis possible | no diagnosis possible | no diagnosis possible | Error identifier $=48$ |
| $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | Error identifier $=7$ | Error identifier $=7$ | Error identifier $=7$ | Error identifier $=48$ |
| $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | no diagnosis possible | Error identifier $=48$ | Error identifier $=7$ | Error identifier $=48$ |

Table 163: Content of diagnosis messages


| E1 ... E4 | d1 | d2 | d3 | d4 |  | $\begin{array}{\|l} \hline \text { Identifier } \\ 000 \ldots \\ 063 \end{array}$ | AC500 display | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch |  | Err | PS501 <br> PLC <br> browser |  |  |
| Byte 6 <br> Bit 6 ... 7 | - | Byte 3 | Byte 4 | Byte 5 |  | Byte 6 <br> Bit 0 ... 5 | FBP diagnosis block |  |  |
| Class | Interface | Device | Module | Channel |  | Error Identifier | Error message |  | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) | ${ }^{4}$ ) |  |  |  |  |  |
| 4 | 14 | 1... 10 | 1 | $0 \ldots 3$ | $0 \ldots 7$ | 48 | Analog value overflow or broken wire at an analog input |  | Check input value or terminal |
|  | 11/12 | ADR | $1 . .10$ |  |  |  |  |  |  |  |  |
| 4 | 14 | 1... 10 | 1 | 0... 3 | $0 \ldots 7$ | 7 | Analog value underflow at an analog input |  | Check input value |
|  | 11/12 | ADR | $1 . . .10$ |  |  |  |  |  |  |  |  |
| 4 | 14 | 1... 10 | 1 | $0 \ldots 3$ | $0 \ldots 7$ | 47 | Short circuit at an analog input |  | Check terminal |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |  |  |  |  |
| 4 | 14 | 1... 10 | 3 | $4 \ldots 7$ | $8 \ldots 15$ | 4 | Analog value overflow at an analog output |  | Check output value |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |  |  |  |  |
| 4 | 14 | 1... 10 | 3 | $4 \ldots 7$ | $8 \ldots 15$ | 7 | Analog value underflow at an analog output |  | Check output value |
|  | 11/12 | ADR | $1 . .10$ |  |  |  |  |  |  |  |  |

Remarks:

| $\left.{ }^{1}\right)$ | In AC500, the following interface identifier applies: <br> $14=$ I/O bus, 11 = COM1 (e.g. CS31 bus), 12 = COM2. <br> The FBP diagnosis block does not contain this identifier. |
| :--- | :--- |
| $\left.{ }^{2}\right)$ | With "Device" the following allocation applies: <br> $31=$ module itself, 1 ... 10 = expansion module $1 \ldots 10$, ADR = hardware <br> address (e.g. of the DC551) |
| $\left.{ }^{3}\right)$ | With "Module" the following allocation applies depending on the master: <br> Module error: I/O bus or FBP: 31 = module itself; COM1/COM2: $1 \ldots 10=$ <br> expansion $1 \ldots 10$ <br> Channel error: I/O bus or FBP = module type (1 = AI, 3 = AO); COM1/COM2: <br> $1 \ldots 10=$ expansion 1 ... 10 |
| $\left.{ }^{4}\right)$ | In case of module errors, with channel "31 = Module itself" is output. |

## State LEDs

During the power ON procedure, the module initializes automatically. All LEDs (except the channel LEDs) are ON during this time.


## Measuring ranges

## Input ranges of voltage, current and digital input

The represented resolution corresponds to 16 bits.

| Range | $\begin{aligned} & \hline 0 \mathrm{~V} \ldots 10 \\ & \mathrm{~V} \end{aligned}$ | $\begin{aligned} & -10 \mathrm{~V} . . . \\ & +10 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 0 \mathrm{~mA} \ldots \\ & 20 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 4 \mathrm{~mA} \ldots \\ & 20 \mathrm{~mA} \end{aligned}$ | Digital input | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  | Decimal | Hex. |
| Overflow | >11.7589 | >11.7589 | >23.5178 | >22.8142 |  | 32767 | 7FFF |
| Measured value too high |  | 11.7589 $:$ 10.0004 | 23.5178 $:$ 20.0007 | 22.8142 $:$ 20.0006 |  | 32511 <br> 27649 | $\begin{aligned} & 7 \mathrm{EFF} \\ & \vdots \\ & 6 \mathrm{C} 01 \end{aligned}$ |
| Normal range <br> Normal range or |  |  | 20.0000 $:$ 0.0007 | 20.0000 $:$ 4.0006 | ON | 27648 <br> 1 | $\begin{array}{\|l\|l\|} \hline 6 \mathrm{C} 00 \\ \vdots \\ 0001 \\ \hline \end{array}$ |
| measured | 0.0000 | 0.0000 | 0 | 4 | OFF | 0 | 0000 |
| $\left\lvert\, \begin{aligned} & \text { valu } \\ & \text { low } \end{aligned}\right.$ | $\begin{aligned} & \hline-0.0004 \\ & -1.7593 \end{aligned}$ | $\begin{aligned} & -0.0004 \\ & : \\ & : \\ & : \\ & -10.0000 \end{aligned}$ |  | $\begin{aligned} & 3.9994 \\ & : \\ & 0 \end{aligned}$ |  | -1 <br> -4864 <br> -6912 <br> -27648 | $\begin{array}{\|l} \hline \text { FFFF } \\ \text { ED00 } \\ \text { E500 } \\ : \\ 9400 \end{array}$ |
| Measured value too low |  |  |  |  |  | -27649 <br> -32512 | $\begin{array}{\|l} \hline \text { 93FF } \\ \vdots \\ 8100 \\ \hline \end{array}$ |
| Underflow | <0.0000 | <-11.7589 | <0.0000 | <0.0000 |  | -32768 | 8000 |

Input ranges resistance temperature detector

| Range | $\begin{aligned} & \hline \mathrm{Pt100} / \mathrm{Pt} \\ & 1000 \\ & -50^{\circ} \mathrm{C} \ldots+70 \\ & { }^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \mathrm{Pt100} \mathrm{I} \\ & \mathrm{Pt} 1000 \\ & -50^{\circ} \mathrm{C} \ldots \\ & +400^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \mathrm{Ni} 1000 \\ & -50^{\circ} \mathrm{C} \ldots \\ & +150^{\circ} \mathrm{C} \end{aligned}$ | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  | Decimal | Hex. |
| Overflow | $>+80.0^{\circ} \mathrm{C}$ | $>+450.0{ }^{\circ} \mathrm{C}$ | $>+160.0{ }^{\circ} \mathrm{C}$ | 32767 | 7FFF |
| Measured value too high |  | $\begin{aligned} & +450.0^{\circ} \mathrm{C} \\ & : \\ & +400.1^{\circ} \mathrm{C} \\ & \hline \end{aligned}$ |  | $\begin{array}{\|l} \hline 4500 \\ : \\ 4001 \\ \hline \end{array}$ | $\begin{aligned} & 1194 \\ & : \\ & \text { OFA1 } \\ & \hline \end{aligned}$ |
|  |  |  | $\begin{aligned} & +160.0^{\circ} \mathrm{C} \\ & : \\ & +150.1^{\circ} \mathrm{C} \\ & \hline \end{aligned}$ | $\begin{array}{\|l} \hline 1600 \\ : \\ 1501 \\ \hline \end{array}$ | $\begin{aligned} & 0640 \\ & : \\ & 05 D D \\ & \hline \end{aligned}$ |
|  | $+80.0^{\circ} \mathrm{C}$ <br> $+70.1^{\circ} \mathrm{C}$ |  |  | $\begin{aligned} & 800 \\ & \vdots \\ & 701 \end{aligned}$ | $\begin{aligned} & \text { 0320 } \\ & : \\ & \text { 02BD } \end{aligned}$ |
| Normal range | $\begin{aligned} & +70.0^{\circ} \mathrm{C} \\ & : \\ & +0.1^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & +400.0^{\circ} \mathrm{C} \\ & : \\ & : \\ & \vdots \\ & +0.1^{\circ} \mathrm{C} \end{aligned}$ | $+150.0^{\circ} \mathrm{C}$ $+0.1^{\circ} \mathrm{C}$ | $\begin{aligned} & 4000 \\ & 1500 \\ & 700 \\ & : \\ & 1 \end{aligned}$ | $\begin{aligned} & \hline \text { OFAO } \\ & \text { 05DC } \\ & \text { 02BC } \\ & : \\ & 0001 \end{aligned}$ |
|  | $0.0{ }^{\circ} \mathrm{C}$ | $0.0{ }^{\circ} \mathrm{C}$ | $0.0{ }^{\circ} \mathrm{C}$ | 0 | 0000 |
|  | $-0.1^{\circ} \mathrm{C}$ <br> $-50.0^{\circ} \mathrm{C}$ | $\begin{array}{\|l\|} \hline-0.1^{\circ} \mathrm{C} \\ : \\ -50.0^{\circ} \mathrm{C} \\ \hline \end{array}$ | $-0.1^{\circ} \mathrm{C}$ $-50.0^{\circ} \mathrm{C}$ | $\begin{array}{\|l} \hline-1 \\ : \\ -500 \\ \hline \end{array}$ | $\begin{aligned} & \text { FFFF } \\ & \text { FEOC } \\ & \hline \end{aligned}$ |
| Measured value too low | $\begin{aligned} & -50.1^{\circ} \mathrm{C} \\ & : \\ & -60.0^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & -50.1^{\circ} \mathrm{C} \\ & : \\ & -60.0^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & -50.1^{\circ} \mathrm{C} \\ & : \\ & -60.0^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & -501 \\ & : \\ & -600 \end{aligned}$ | $\begin{aligned} & \text { FEOB } \\ & : \\ & \text { FDA8 } \end{aligned}$ |
| Underflow | $<-60.0{ }^{\circ} \mathrm{C}$ | <-60.0 ${ }^{\circ} \mathrm{C}$ | $<-60.0{ }^{\circ} \mathrm{C}$ | -32768 | 8000 |

## Output ranges voltage and current

The represented resolution corresponds to 16 bits.

| Range | -10 V ... +10 V | $0 \mathrm{~mA} \ldots 20$ | $4 \mathrm{~mA} \ldots 20$ | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  | Decimal | Hex. |
| Overflow | 0 V | 0 mA | 0 mA | > 32511 | > 7EFF |
| Value too high | $\begin{aligned} & 11.7589 \mathrm{~V} \\ & : \\ & 10.0004 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 23.5178 \mathrm{~mA} \\ & : \\ & 20.0007 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 22.8142 \mathrm{~mA} \\ & : \\ & 20.0006 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 32511 \\ & : \\ & 27649 \end{aligned}$ | $\begin{aligned} & \text { 7EFF } \\ & : \\ & 6 \mathrm{CO} \end{aligned}$ |


| Range | -10 V ... +10 V | $\mathrm{O}_{\mathrm{mA}}^{\mathrm{mA} \ldots 20}$ | $4 \mathrm{~mA} \ldots 20$ | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  | Decimal | Hex. |
| Normal range | $\begin{aligned} & \hline 10.0000 \mathrm{~V} \\ & : \\ & 0.0004 \mathrm{~V} \\ & \hline \end{aligned}$ | $\begin{aligned} & 20.0000 \mathrm{~mA} \\ & : \\ & 0.0007 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 20.0000 \mathrm{~mA} \\ & : \\ & 4.0006 \mathrm{~mA} \end{aligned}$ | $27648$ | $\begin{aligned} & \text { 6C00 } \\ & : \\ & 0001 \end{aligned}$ |
|  | 0.0000 V | 0.0000 mA | 4.0000 mA | 0 | 0000 |
|  | $\begin{aligned} & -0.0004 \mathrm{~V} \\ & : \\ & -10.0000 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 0 \mathrm{~mA} \\ & : \\ & 0 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 3.9994 \mathrm{~mA} \\ & 0 \mathrm{~mA} \\ & 0 \mathrm{~mA} \end{aligned}$ | $\begin{array}{\|l} \hline-1 \\ -6912 \\ -27648 \end{array}$ | $\begin{aligned} & \text { FFFF } \\ & \text { E500 } \\ & 9400 \end{aligned}$ |
| Value too low | $\begin{array}{\|l} \hline-10.0004 \mathrm{~V} \\ : \\ -11.7589 \mathrm{~V} \\ \hline \end{array}$ | $\begin{aligned} & 0 \mathrm{~mA} \\ & : \\ & 0 \mathrm{~mA} \end{aligned}$ |  | $\begin{aligned} & -27649 \\ & : \\ & -32512 \\ & \hline \end{aligned}$ | $\begin{aligned} & \text { 93FF } \\ & : \\ & 8100 \end{aligned}$ |
| Underflow | 0 V | 0 mA | 0 mA | <-32512 | < 8100 |

## Technical data

## Technical data of the module

The System Data of AC500 and S500 « Chapter 5.1.2 "System data AC500" on page 166 are applicable to the standard version.

Only additional details are therefore documented below.

| Parameter | Value |
| :---: | :---: |
| Process voltage |  |
| Connections | Terminals 1.8, 2.8, 3.8 and 4.8 for +24 V (UP) as well as 1.9, 2.9, 3.9 and 4.9 for 0 V (ZP) |
| Rated value | 24 VDC |
| Max. ripple | 5 \% |
| Protection against reversed voltage | Yes |
| Rated protection fuse on UP | 10 A fast |
| Galvanic isolation | Yes, per module |
| Current consumption |  |
| From 24 VDC power supply at the terminals UP/L+ and ZP/M of the CPU/bus module | Ca. 2 mA |
| From UP at normal operation | 0.10 A + output loads |
| Inrush current from UP (at power up) | $0.040 \mathrm{~A}^{2} \mathrm{~s}$ |
| Max. length of analog cables, conductor cross section $>0.14 \mathrm{~mm}^{2}$ | 100 m |
| Weight | 300 g |


| Parameter | Value |
| :--- | :--- |
| Mounting position | Horizontal or vertical with derating <br> (output load reduced to $50 \%$ at $+40^{\circ} \mathrm{C}$ <br> per group) |
| Cooling | The natural convection cooling must not <br> be hindered by cable ducts or other parts <br> in the control cabinet. |

- NOTICE!

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and temporary overvoltage up to 30 V DC.

## Technical data of the analog inputs

| Parameter | Value |  |
| :---: | :---: | :---: |
| Number of channels per module | 8 |  |
| Distribution of channels into groups | 1 group of 8 channels |  |
| Connections of the channels C0- ... C7- | Terminals 2.0 ... 2.7 |  |
| Connections of the channels $\mathrm{C} 0+\ldots \mathrm{C} 7+$ | Terminals 3.0 ... 3.7 |  |
| Input type | Bipolar (not with current or Pt100/Pt1000/Ni1000) |  |
| Galvanic isolation | Against internal supply and other modules |  |
| Configurability | $0 \vee \ldots 10 \mathrm{~V},-10 \mathrm{~V} \ldots+10 \mathrm{~V}, 0 \mathrm{~mA} . .20 \mathrm{~mA}$, 4 mA ... 20 mA, Pt100/1000, Ni1000 (each input can be configured individually) |  |
| Channel input resistance | $\begin{aligned} & \text { Voltage: > } 100 \mathrm{k} \Omega \\ & \text { Current: ca. } 330 \Omega \end{aligned}$ |  |
| Time constant of the input filter | Voltage: $100 \mu \mathrm{~s}$Current: $100 \mu \mathrm{~s}$ |  |
| Indication of the input signals | One LED per channel |  |
| Conversion cycle | 2 ms (for 8 inputs +8 outputs), with $\mathrm{Pt} / \mathrm{Ni}$... 1 s |  |
| Resolution | Range 0 V ... 10 V : 12 bits <br> Range $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ : 12 bits including sign <br> Range $0 \mathrm{~mA} . .20 \mathrm{~mA}$ : 12 bits <br> Range $4 \mathrm{~mA} . . .20 \mathrm{~mA}$ : 12 bits |  |
| Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal | Typ. | $\begin{aligned} & \pm 0.5 \% \text { of full scale } \\ & \text { at }+25^{\circ} \mathrm{C} \end{aligned}$ |
|  | Max. | $\pm 1$ \% of full scale (all ranges) <br> at $0^{\circ} \mathrm{C} \ldots+60^{\circ} \mathrm{C}$ or EMC disturbance |
| Relationship between input signal and hex code | See table ${ }^{\mu}$ Chapter 5.2.4.4.2.1.10.1 "Input ranges of voltage, current and digital input" on page 692 |  |
| Unused inputs | Must be configured as "unused". |  |
| Overvoltage protection | Yes |  |

Technical data of the analog inputs, if used as digital inputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | Max. 8 |
| Distribution of channels into groups | 1 group of 8 channels |
| Connections of the channels $\mathrm{C} 0+\ldots \mathrm{C} 7+$ | Terminals $3.0 \ldots 3.7$ |
| Reference potential for the inputs | Terminals $1.9 \ldots 4.9(\mathrm{ZP})$ |
| Input signal delay | Typ. 8 ms, configurable from $0.1 \ldots 32 \mathrm{~ms}$ |
| Indication of the input signals | 1 LED per channel |
| Input signal voltage | 24 VDC |
|  | Signal 0 |
|  | Undefined signal |
|  | Signal 1 |
| Input current per channel | $+5 \mathrm{~V} \ldots+5 \mathrm{~V} . . .13 \mathrm{~V}$ |
|  | Input voltage +24 V |
|  | Input voltage +5 V |
|  | Input voltage +15 V |
|  | Input voltage +30 V |
| Input resistance | Typ. 7 mA |

## Technical data of the analog outputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 8, all channels for voltage, the first 4 channels <br> also for current |
| Distribution of channels into groups | 1 group of 8 channels |
|  | Channels C0- .. C7- |
| Channels C0+ ... C7+ | Terminals $2.0 \ldots 2.7$ |
| Output type | Terminals $3.0 \ldots 3.7$ |
| Galvanic isolation | Against internal supply and other modules |
| Configurability | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}, 0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$, <br> $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ (each output can be config- <br> ured individually), current outputs only chan- <br> nels $0 \ldots 3$ |
| Output resistance (load), as current output | $0 \Omega \ldots 500 \Omega$ |
| Output loadability, as voltage output | Max. $\pm 10 \mathrm{~mA}$ |
| Indication of the output signals | One LED per channel |
| Resolution | 12 bits including sign |
| Settling time for full range change (resistive <br> load, output signal within specified tolerance) | Typ. 5 ms |
| Conversion error of the analog values caused <br> by non-linearity, adjustment error at factory <br> and resolution within the normal range | Typ. |


| Parameter | Value |
| :--- | :--- |
|  | Max. |
| $\pm 1 \%$ of full scale (all ranges) <br> at $0^{\circ} \mathrm{C} \ldots+60^{\circ} \mathrm{C}$ or EMC disturb- <br> ance |  |
| Relationship between output signal and hex <br> code | See table 'AC522 - Analog input/output <br> module' $\leftrightarrows$ Chapter 5.2.4.4.2.1.10.3 "Output <br> ranges voltage and current" on page 693 |
| Unused outputs | Must be configured as "unused". |

## Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm

The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 250 500 R0001 | AC522, analog input/output module, <br> 8 AC, U/I/RTD, 12 bits including sign, <br> 2-wires | Active |
| 1SAP 450 500 R0001 | AC522-XC, analog input/output <br> module, 8 AC, U/I/RTD, <br> 12 bits including sign, 2-wires, XC ver- <br> sion | Active |

${ }^{*}$ ) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

Al523 - Analog input module

## Features

- 16 configurable analog inputs (IO ... I15) in 2 groups (1.0 ... 2.7 and $3.0 \ldots 4.7$ )

Resolution 12 bits including sign

- Module-wise galvanically isolated
- XC version for use in extreme ambient conditions available


1 I/O bus
2 Allocation between terminal number and signal name
316 yellow LEDs to display the signal states at the analog inputs (IO ... I15)
41 green LED to display the state of the process supply voltage UP
$5 \quad 2$ red LEDs to display errors
6 Label
7 Terminal unit
8 DIN rail
Sign for XC version

## Intended purpose

The device can be used as a decentralized I/O extension module for S 500 communication interface modules (e. g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs.

## Functionality

16 analog inputs, individually configurable for

- Unused (default setting)
- 0 V ... 10 V
-     - $10 \mathrm{~V} \ldots+10 \mathrm{~V}$
- $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$
- 4 mA ... 20 mA
- Pt100, $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ (2-wire)
- Pt100, $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ (3-wire), requires 2 channels
- Pt100, $-50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ (2-wire)
- Pt100, $-50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ (3-wire), requires 2 channels
- Pt1000, $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ (2-wire)
- Pt1000, $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ (3-wire), requires 2 channels
- Ni1000, $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ (2-wire)
- Ni1000, $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ (3-wire), requires 2 channels
- $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ with differential inputs, requires 2 channels
- $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ with differential inputs, requires 2 channels
- Digital signals (digital input)

| Parameter | Value |
| :---: | :---: |
| Resolution of the analog channels |  |
| Voltage -10 V ... +10 V | 12 bits including sign |
| Voltage 0 V ... 10 V | 12 bits |
| Current 0 mA ... $20 \mathrm{~mA}, 4 \mathrm{~mA} . . .20 \mathrm{~mA}$ | 12 bits |
| Temperature | $0.1{ }^{\circ} \mathrm{C}$ |
| LED displays | 19 LEDs for signals and error messages |
| Internal power supply | Via the I/O bus interface (I/O bus) |
| External power supply | Via the terminals ZP and UP (process voltage 24 V DC) |
| Required terminal unit | TU515 or TU516 ${ }^{\Perp}$ Chapter 5.2.5.2 "TU515, TU516, TU541 and TU542 for I/O modules" on page 938 |

## Connections

The modules are plugged on an I/O terminal unit $\triangleq$ Chapter 5.2.5.2 "TU515, TU516, TU541 and TU542 for I/O modules" on page 938. Properly position the modules and press until they lock in place. The terminal units are mounted on a DIN rail or with 2 screws plus the additional accessory for wall mounting ${ }^{\wedge}>$ Chapter 5.2.8.2.5 "TA526-Wall mounting accessory" on page 1324.

The connection of the I/O channels is carried out using the 40 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.


The terminals $1.8 \ldots 4.8$ and $1.9 \ldots 4.9$ are electrically interconnected within the I/O terminal units and have always the same assignment, independent of the inserted module:

Terminals 1.8 ... 4.8: process voltage UP $=+24 \mathrm{~V}$ DC
Terminals $1.9 \ldots 4.9$ : process voltage $\mathrm{ZP}=0 \mathrm{~V}$
The assignment of the other terminals:

| Terminals | Signal | Description |
| :--- | :--- | :--- |
| $1.0 \ldots 1.7$ | $10-\ldots$ I7- | Negative poles of the first 8 <br> analog inputs |
| $2.0 \ldots 2.7$ | $10+\ldots$ I7+ | Positive poles of the first 8 <br> analog inputs |
| $3.0 \ldots 3.7$ | $18-\ldots$ I15- | Negative poles of the fol- <br> lowing 8 analog inputs |
| $4.0 \ldots 4.7$ | $18+\ldots$ I15+ | Positive poles of the following <br> 8 analog inputs |

## CAUTION!

The negative poles of the analog inputs are galvanically connected to each other. They form an "Analog Ground" signal for the module. The negative poles of the analog outputs are also galvanically connected to each other to form an "Analog Ground" signal.

## CAUTION!

There is no galvanic isolation between the analog circuitry and ZP/UP. Therefore, the analog sensors must be galvanically isolated in order to avoid loops via the ground potential or the supply voltage.

## CAUTION!

Because of their common reference potential, analog current inputs cannot be circuited in series, neither within the module nor with channels of other modules.

For the open-circuit detection (wire break), each analog input channel is pulled up to "plus" by a high-resistance resistor. If nothing is connected, the maximum voltage will be read in then.

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a communication interface module or a CPU). Thus, the current consumption from 24 V DC power supply at the terminals L+/UP and M/ZP of the CPU/communication interface module increases by 2 mA per Al523.

The external power supply connection is carried out via the UP (+24 V DC) and the ZP (0 V DC) terminals.

## WARNING!

## Removal/Insertion under power

Removal or insertion under power is permissible only if all conditions for hot swapping are fullfilled.

๕ Chapter 4.9.3 "Replace an I/O module with hot swap" on page 153
The devices are not designed for removal or insertion under power when the conditions for hot swap do not apply. Because of unforeseeable consequences, it is not allowed to plug in or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while they are energized in a hazardous location could result in an electric arc, which could create an ignition source resulting in fire or explosion.

Prior to proceeding, make sure that power is been disconnected and that the area has been thoroughly checked to ensure that flammable materials are not present.

The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

## Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

Generally, analog signals must be laid in shielded cables. The cable shields must be grounded at both sides of the cables. In order to avoid unacceptable potential differences between different parts of the installation, low resistance equipotential bonding conductors must be laid.

The following figure shows the connection of the module:


Fig. 90: 16 analog inputs in two groups, individually configurable ${ }^{\Perp}$ Chapter 5.2.4.4.2.2.3 "Functionality" on page 699

## CAUTION!

By installing equipotential bonding conductors between the different parts of the system, it must be ensured that the potential difference between ZP and AGND never can exceed 1 V .

## CAUTION!

The process supply voltage must be included in the grounding concept (e. g. grounding of the negative terminal).

The modules provide several diagnosis functions $\left.{ }^{\circ}\right\rangle$ Chapter 5.2.4.4.2.2.8 "Diagnosis" on page 717.

## Connection of resistance thermometers in 2-wire configuration

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the module Al523 provides a constant current source which is multiplexed over the 8 analog channels.


Fig. 91: Connection example
The following measuring ranges can be configured ${ }^{\wedge}$ Chapter 5.2.4.4.2.2.7 "Parameterization" on page 714.

| Pt100 | $-50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ | 2-wire configuration, one <br> channel used |
| :--- | :--- | :--- |
| $\mathrm{Pt100}$ | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 2-wire configuration, one <br> channel used |
| $\mathrm{Pt1000}$ | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 2-wire configuration, one <br> channel used |
| Ni1000 | $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ | 2-wire configuration, one <br> channel used |

The function of the LEDs is described under Displays $\Leftrightarrow$ Chapter 5.2.4.4.2.2.8 "Diagnosis" on page 717.

The module AI523 performs a linearization of the resistance characteristic.
In order to avoid error messages from unused analog input channels, it is useful to configure them as "unused".

## Connection of resistance thermometers in 3-wire configuration

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the module Al523 provides a constant current source which is multiplexed over the max. 8 (depending on the configuration) analog channels.


Fig. 92: Connection example
If several measuring points are adjacent to each other, the return line is necessary only once. This saves wiring costs.

With 3-wire configuration, two adjacent analog channels belong together (e.g. the channels 0 and 1). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0 ), the next higher address must be the odd address (channel 1).

The constant current of one channel flows through the resistance thermometer. The constant current of the other channel flows through one of the cores. The module calculates the measured value from the two voltage drops and stores it under the input with the higher channel number (e.g. I1).
In order to keep measuring errors as small as possible, it is necessary to have all the involved conductors in the same cable. All the conductors must have the same cross section.
The following measuring ranges can be configured ${ }^{y}$ Chapter 5.2.4.4.2.2.7 "Parameterization" on page 714

| $\operatorname{Pt100}$ | $-50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ | 3-wire configuration, two <br> channels used |
| :--- | :--- | :--- |
| $\mathrm{Pt100}$ | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 3-wire configuration, two <br> channels used |
| $\mathrm{Pt1000}$ | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 3-wire configuration, two <br> channels used |
| Ni1000 | $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ | 3-wire configuration, two <br> channels used |

The function of the LEDs is described under Displays ${ }^{*}$ Chapter 5.2.4.4.2.2.8 "Diagnosis" on page 717.

The module AI523 performs a linearization of the resistance characteristic.
In order to avoid error messages from unused analog input channels, it is useful to configure them as "unused".


Fig. 93: Connection example

By connecting the sensor's negative pole of the output voltage to AGND, the galvanically isolated voltage source of the sensor is referred to ZP.

The following measuring ranges can be configured ${ }^{\mu}$ Chapter 5.2.4.4.2.2.7 "Parameterization" on page $714 \Leftrightarrow$ Chapter 5.2.4.4.2.2.10 "Measuring ranges" on page 719

| Voltage | $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ | 1 channel used |
| :--- | :--- | :--- |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | 1 channel used |

The function of the LEDs is described under Displays $\Rightarrow$ Chapter 5.2.4.4.2.2.8 "Diagnosis" on page 717.

In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as "unused".

## Connection of active-type analog sensors (Current) with galvanically isolated power supply



Fig. 94: Connection example
The following measuring ranges can be configured $\Leftrightarrow$ Chapter 5.2.4.4.2.2.7 "Parameterization" on page $714 \Leftrightarrow$ Chapter 5.2.4.4.2.2.10 "Measuring ranges" on page 719

| Current | $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 1 channel used |
| :--- | :--- | :--- |
| Current | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 1 channel used |

The function of the LEDs is described under Displays $\Leftrightarrow$ Chapter 5.2.4.4.2.2.8 "Diagnosis" on page 717.

Unused input channels can be left open-circuited, because they are of low resistance.


Fig. 95: Connection example

## CAUTION!

The potential difference between AGND and ZP at the module must not be greater than 1 V , not even in case of long lines ${ }^{\#} \Rightarrow$ Chapter 5.2.4.4.2.2 "Al523 Analog input module" on page 698.

If AGND does not get connected to $Z P$, the sensor current flows to $Z P$ via the AGND line. The measuring signal is distorted, as a very low current flows over the voltage line. The total current through the PTC should not exceed 50 $m A$. This measuring method is therefore only suitable for short lines and small sensor currents. If there are bigger distances, the difference measuring method has to be preferred.

The following measuring ranges can be configured $\Leftrightarrow$ Chapter 5.2.4.4.2.2.10 "Measuring ranges" on page 719

| Voltage | $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ | 1 channel used |
| :--- | :--- | :--- |
| Voltage | $\left.-10 \mathrm{~V} \ldots+10 \mathrm{~V}{ }^{*}\right)$ | 1 channel used |
| $\left.{ }^{*}\right)$ if the sensor can provide this signal range |  |  |

The function of the LEDs is described under Displays « Chapter 5.2.4.4.2.2.8 "Diagnosis" on page 717.
In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as "unused".

## Connection of passive-type analog sensors (Current)



Fig. 96: Connection example
The following measuring ranges can be configured $\Longleftrightarrow$ Chapter 5.2.4.4.2.2.7 "Parameterization" on page $714 \Leftrightarrow$ Chapter 5.2.4.4.2.2.10 "Measuring ranges" on page 719

| Current | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 1 channel used |
| :--- | :--- | :--- |

The function of the LEDs is described under Displays $\Leftrightarrow$ Chapter 5.2.4.4.2.2.8 "Diagnosis" on page 717.

## CAUTION!

If, during initialization, an analog current sensor supplies more than 25 mA for more than 1 second into an analog input, this input is switched off by the module (input protection). In such cases, it is recommended to protect the analog input by a 10 -volt Zener diode (in parallel to I+ and ZP). But, in general, it is a better solution to use sensors with fast initialization or without current peaks higher than 25 mA .

Unused input channels can be left open-circuited, because they are of low resistance.

## Connection of active-type analog sensors (Voltage) to differential inputs

Differential inputs are very useful if analog sensors which are remotely non-isolated (e.g. the negative terminal is remotely grounded) are used.
The evaluation using differential inputs helps to considerably increase the measuring accuracy and to avoid ground loops.
With differential input configurations, two adjacent analog channels belong together (e.g. the channels 0 and 1). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0 ), the next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1).

The analog value is calculated by subtraction of the input value with the higher address from the input value of the lower address.

The converted analog value is available at the odd channel (higher address).

## CAUTION!

The ground potential at the sensors must not have too big a potential difference with respect to ZP (max. $\pm 1 \mathrm{~V}$ within the full signal range). Otherwise problems can occur concerning the common-mode input voltages of the involved analog inputs.


Fig. 97: Connection example

The negative pole of the sensor must be grounded next to the sensor.

The following measuring ranges can be configured ${ }^{\mu}$ Chapter 5.2.4.4.2.2.7 "Parameterization" on page $714 \Leftrightarrow$ Chapter 5.2.4.4.2.2.10 "Measuring ranges" on page 719:

| Voltage | $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ | with differential inputs, 2 chan- <br> nels used |
| :--- | :--- | :--- |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | with differential inputs, 2 chan- <br> nels used |

The function of the LEDs is described under Displays $\Leftrightarrow$ Chapter 5.2.4.4.2.2.8 "Diagnosis" on page 717.
In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as "unused".

## Use of analog inputs as digital inputs

Several (or all) analog inputs can be configured as digital inputs. The inputs are not galvanically isolated against the other analog channels.


Fig. 98: Connection example
The following operating mode can be configured ${ }^{\mu}$ Chapter 5.2.4.4.2.2.7 "Parameterization" on page $714 \Leftrightarrow$ Chapter 5.2.4.4.2.2.10 "Measuring ranges" on page 719

| Digital input | 24 V | 1 channel used |
| :--- | :--- | :--- |
| Effect of incorrect input ter- <br> minal connection |  | Wrong or no signal detected, <br> no damage up to 35 V |

The function of the LEDs is described under Displays.

## Internal data exchange

| Digital inputs (bytes) | 0 |
| :--- | :--- |
| Digital outputs (bytes) | 0 |
| Counter input data (words) | 16 |
| Counter output data (words) | 0 |

## I/O configuration

The module does not store configuration data itself. It gets its parameterization data from the master device of the I/O bus (CPU or communication interface module) during power-up of the system.
That means replacing I/O modules is possible without any re-parameterization via software.


If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.

## Parameterization

| Firmware version | Configuration |
| :--- | :--- |
| Firmware version > V2.0.0 | The arrangement of the parameter data is per- <br> formed by Control Builder Plus/ Automation <br> Builder software. |

The parameter data directly influences the functionality of modules.
For non-standard applications, it is necessary to adapt the parameters to your system configuration.

Module: Module slot address: Y = 1 ... 10

| No. | Name | Value | Internal value | Internal <br> value, type | Default | Min. | Max. | EDS <br> Slot/ Index |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 1 | Module ID | Internal | $1515$ <br> ${ }^{1}$ ) | Word | $\begin{aligned} & 1515 \\ & 0 x 05 \mathrm{eb} \end{aligned}$ | 0 | 65535 | 0x0Y01 |
| 2 | Ignore module ${ }^{2}$ ) | No Yes | $\begin{array}{\|l\|} \hline 0 \\ 1 \end{array}$ | Byte | $\begin{array}{\|l\|} \hline \text { No } \\ 0 \times 00 \end{array}$ |  |  | not for FBP |
| 3 | Parameter length in bytes | Internal | 34 | Byte | 34-CPU <br> 34-FBP | 0 | 255 | 0x0Y02 |
| 4 | Check supply | $\begin{aligned} & \text { Off } \\ & \text { On } \end{aligned}$ | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | Byte | $\begin{aligned} & \text { On } \\ & 0 \times 01 \end{aligned}$ | 0 | 1 | 0x0Y03 |
| 5 | Analog data format | Default | 0 | Byte | Default $0 \times 00$ |  |  | 0x0Y04 |


| No. | Name | Value | Internal value | Internal <br> value, type | Default | Min. | Max. | EDS <br> Slot/ Index |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 6 | Channel configu- <br> ration <br> Input channel 0 | See tab configu \& Tabl nel con on pag | 'Channel ion' 64 "Chan uration ${ }^{2}$ )" 76 | Byte | $\begin{aligned} & \hline \text { Default } \\ & 0 \times 00 \end{aligned}$ | 0 | 19 | 0x0Y05 |
| 7 | Channel monitoring Input channel 0 | See table monitor \& Table nel mon on page | 'Channel <br> 65 "Chan ring ${ }^{4}$ )" <br> 17 | Byte | $\begin{aligned} & \hline \text { Default } \\ & 0 \times 00 \end{aligned}$ | 0 | 3 | 0x0Y06 |
| $\begin{array}{\|l\|} \hline 8 \\ \text { to } \\ 35 \end{array}$ | Channel configu- <br> ration <br> and <br> channel <br> moni- <br> toring of <br> the input <br> channels <br> 1 ... 14 | See <br> table 'C figuratio « Table nel confi on page <br> and <br> table 'C monitorin * Table nel moni on page | annel con- <br> 164 "Chan uration ${ }^{2}$ )" 716 <br> annel g 165 "Chan oring ${ }^{4}$ " 717 | Byte <br> Byte | $\begin{array}{\|l\|} \hline \text { Default } \\ 0 \times 00 \\ 0 \times 00 \end{array}$ | $\begin{aligned} & 0 \\ & 0 \end{aligned}$ | $\begin{aligned} & 19 \\ & 3 \end{aligned}$ | 0x0Y07 to 0x0Y22 |
| 36 | Channel configuration Input channel 15 | See <br> table 'Ch figuratio « Table nel confi on page | annel con- <br> 164 "Chan uration ${ }^{2}$ )" 716 | Byte | Default <br> 0x00 | 0 | 19 | 0x0Y23 |
| 37 | Channel monitoring Input channel 15 | See tabl monitor \& Table nel mon on page | $\begin{aligned} & \text { 'Channel } \\ & \text { gs' } \\ & 165 \text { "Chan } \\ & \text { oring 4)" } \\ & 717 \end{aligned}$ | Byte | $\begin{array}{\|l\|} \hline \text { Default } \\ 0 \times 00 \end{array}$ | 0 | 3 | 0x0Y24 |
| ${ }^{1}$ ) With CS31 and addresses less than 70 and FBP, the value is increased by 1 <br> ${ }^{2}$ ) Not with FBP |  |  |  |  |  |  |  |  |

GSD file:

| Ext_User_Prm_Data_Len = | 37 |
| :---: | :---: |
| Ext_User_Prm_Data_Const(0) = | 0x05, 0xec, 0x22, 1 |
|  | 0x01, 0x00, 1 |
|  | $\begin{aligned} & 0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00 \\ & 0 \times 00,1 \end{aligned}$ |
|  | $\begin{aligned} & 0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00, \\ & 0 \times 00,1 \end{aligned}$ |
|  | $\begin{aligned} & 0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00 \\ & 0 \times 00,1 \end{aligned}$ |
|  | $\begin{aligned} & 0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00, \\ & 0 \times 00 ; \end{aligned}$ |

Input channel (16 x with Al523)

| No. | Name | Value | Internal value | Internal <br> value, type | Default |
| :--- | :--- | :--- | :--- | :--- | :--- |
| 1 | Channel con- <br> figuration | see table $^{2}$ ) | see table ${ }^{2}$ ) | Byte | 0 <br> $0 \times 00$ see $\left.{ }^{3}\right)$ |
| 2 | Channel mon- <br> itoring | ${\text { see table }{ }^{4} \text { ) }}^{\left.\text {See table }{ }^{4}\right)}$ | Byte | 0 <br> $0 x 00$ see $\left.{ }^{5}\right)$ |  |

Table 164: Channel configuration ${ }^{2}$ )

| Interna I value | Operating modes of the analog inputs, individually configurable |
| :---: | :---: |
| 0 | Unused (default) ${ }^{3}$ ) |
| 1 | Analog input 0 V ... 10 V |
| 2 | Digital input |
| 3 | Analog input $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ |
| 4 | Analog input 4 mA ... 20 mA |
| 5 | Analog input -10 V ... +10 V |
| 8 | Analog input Pt100, $-50^{\circ} \mathrm{C} \ldots+400{ }^{\circ} \mathrm{C}$ (2-wire) |
| 9 | Analog input Pt100, $-50^{\circ} \mathrm{C} \ldots+40{ }^{\circ} \mathrm{C}$ (3-wire), requires 2 channels *) |
| 10 | Analog input $0 \ldots 10 \mathrm{~V}$ via differential inputs, requires 2 channels *) |
| 11 | Analog input -10 V ... +10 V via differential inputs, requires 2 channels *) |
| 14 | Analog input Pt100, $-50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ (2-wire) |
| 15 | Analog input Pt100, $-50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ (3-wire), requires 2 channels *) |
| 16 | Analog input Pt1000, $-50^{\circ} \mathrm{C} \ldots+400{ }^{\circ} \mathrm{C}$ (2-wire) |
| 17 | Analog input Pt1000, $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ (3-wire), requires 2 channels *) |
| 18 | Analog input Ni1000, $-50^{\circ} \mathrm{C} \ldots+150{ }^{\circ} \mathrm{C}$ (2-wire) |


| Interna <br> I value | Operating modes of the analog inputs, individually configurable |
| :--- | :--- |
| 19 | Analog input Ni1000, $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}(3-$ wire $)$, requires 2 channels *) |
|  | *) In the operating modes with 3-wire configuration or with differential inputs, two <br> adjacent analog inputs belong together (e.g. the channels 0 and 1). In these cases, <br> both channels are configured in the desired operating mode. The lower address must <br> be the even address (channel 0). The next higher address must be the odd address <br> (channel 1). The converted analog value is available at the higher address (channel <br> 1). |

Table 165: Channel monitoring ${ }^{4}$ )

| Intern <br> al <br> value | Monitoring |
| :--- | :--- |
| 0 | Plausibility, open-circuit (broken wire) and short circuit <br> 5 |
| 1 | Open-circuit and short circuit |
| 2 | Plausibility |
| 3 | No monitoring |

## Diagnosis

Table 166: Possible diagnosis of I/O channels

| Input range |  |  |  |  |
| :--- | :--- | :--- | :--- | :--- |
|  | Condition | Short circuit | Input value under- <br> flow | Input value over- <br> flow |
| $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | no diagnosis possible | no diagnosis possible | no diagnosis possible | Error identifier $=48$ |
| $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | Error identifier $=7$ | Error identifier $=7$ | Error identifier $=7$ | Error identifier $=48$ |
| $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | no diagnosis possible | Error identifier $=48$ | Error identifier $=7$ | Error identifier $=48$ |


| E1 ... E4 | d1 | d2 | d3 | d4 | Identifier $000 \text {... } 063$ | AC500 <br> display | <- Display in |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err | PS501 <br> PLC <br> browser |  |
| Byte 6 <br> Bit 6 ... 7 | - | Byte 3 | Byte 4 | Byte 5 | Byte 6 <br> Bit 0 ... 5 | FBP diagnosis block |  |
| Class | Interface | Device | Module | Channel | Error identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) | ${ }^{4}$ ) |  |  |  |
| Module error |  |  |  |  |  |  |  |
| 3 | 14 | $1 . .10$ | 31 | 31 | 19 | Checksum error in the I/O module | Replace I/O module |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |
| 3 | 14 | $1 \ldots 10$ | 31 | 31 | 3 | Timeout in the I/O module | Replace I/O module |
|  | 11/12 | ADR | 1 ... 10 |  |  |  |  |


| E1 ... E4 | d1 | d2 | d3 | d4 | $\begin{aligned} & \text { Identifier } \\ & 000 \ldots 063 \end{aligned}$ | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err |  |  |
| Byte 6 <br> Bit 6 ... 7 | - | Byte 3 | Byte 4 | Byte 5 | Byte 6 <br> Bit 0 ... 5 |  |  |
| Class | Interface | Device | Module | Channel | Error identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | $\left.{ }^{2}\right)$ | $\left.{ }^{3}\right)$ | $\left.{ }^{4}\right)$ |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 40 | Different hard-/firmware versions in the module | Replace I/O module |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 43 | Internal error in the module | Replace I/O module |
|  | $11 / 12$ | ADR | 1... 10 |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 36 | Internal data exchange failure | Replace I/O module |
|  | $11 / 12$ | ADR | 1... 10 |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 9 | Overflow diagnosis buffer | New start |
|  | $11 / 12$ | ADR | 1... 10 |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 26 | Parameter error | Check master |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 11 | Process voltage too low | Check process voltage |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |
| 4 | 14 | 1... 10 | 31 | 31 | 45 | Process voltage is switched off (ON -> OFF) | Process voltage ON |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |
| Channel error |  |  |  |  |  |  |  |
| 4 | 14 | 1... 10 | 1 | $0 \ldots 15$ | 48 | Analog value overflow or broken wire at an analog input | Check input value or terminal |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |
| 4 | 14 | 1... 10 | 1 | $0 \ldots 15$ | 7 | Analog value underflow at an analog input | Check input value |
|  | 11 / 12 | ADR | 1... 10 |  |  |  |  |
| 4 | 14 | 1... 10 | 1 | 0... 15 | 47 | Short circuit at an analog input | Check terminal |
|  | 11/12 | ADR | $1 . .10$ |  |  |  |  |

Remarks:

| $\left.{ }^{1}\right)$ | In AC500, the following interface identifier applies: <br> $14=1 / O$ bus, $11=$ COM1 (e.g. CS31 bus), $12=$ COM2. <br> The FBP diagnosis block does not contain this identifier. |
| :--- | :--- |
| $\left.{ }^{2}\right)$ | With "Device" the following allocation applies: <br> $31=$ module itself, $1 \ldots 10=$ expansion module $1 \ldots 10$, ADR $=$ hardware <br> address (e.g. of the DC551) |


| $\left.{ }^{3}\right)$ | With "Module" the following allocation applies depending on the master: <br> Module error: I/O bus or FBP: 31 = module itself; COM1/COM2: $1 \ldots 10=$ <br> expansion $1 \ldots 10$ <br> Channel error: I/O bus or FBP = module type (1 = AI); COM1/COM2: $1 \ldots 10=$ <br> expansion $1 \ldots 10$ |
| :--- | :--- |
| $\left.{ }^{4}\right)$ | In case of module errors, with channel " $31=$ Module itself" is output. |

## State LEDs

During the power ON procedure, the module initializes automatically. All LEDs (except the channel LEDs) are ON during this time.


## Measuring ranges

Input ranges of voltage, current and digital input
The represented resolution corresponds to 16 bits.

| Range | $0 \mathrm{~V} . .10 \mathrm{~V}$ | $\begin{aligned} & -10 \mathrm{~V} \ldots \\ & +10 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 0 \mathrm{~mA} \ldots \\ & 20 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 4 \mathrm{~mA} \ldots \\ & 20 \mathrm{~mA} \end{aligned}$ | Digital input | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  | Decimal | Hex. |
| Overflow | >11.7589 | >11.7589 | >23.5178 | >22.8142 |  | 32767 | 7FFF |
| Measured value too high | $\begin{aligned} & 11.7589 \\ & \cdot \\ & 10.0004 \end{aligned}$ | $\begin{aligned} & 11.7589 \\ & : \\ & 10.0004 \end{aligned}$ | 23.5178 <br> 20.0007 | 22.8142 $:$ 20.0006 |  | $\begin{aligned} & 32511 \\ & : \\ & 27649 \end{aligned}$ | $\begin{aligned} & \text { 7EFF } \\ & : \\ & 6 \mathrm{C} 01 \end{aligned}$ |
|  | $\begin{aligned} & 10.0000 \\ & : \\ & 0.0004 \end{aligned}$ | $\begin{aligned} & 10.0000 \\ & : \\ & 0.0004 \end{aligned}$ | $\begin{aligned} & 20.0000 \\ & : \\ & 0.0007 \end{aligned}$ | $\begin{aligned} & 20.0000 \\ & : \\ & 4.0006 \end{aligned}$ | ON | $27648$ <br> 1 | $\begin{aligned} & \hline 6 \mathrm{C} 00 \\ & : \\ & 0001 \end{aligned}$ |
|  | 0.0000 | 0.0000 | 0 | 4 | OFF | 0 | 0000 |
|  | $\begin{array}{\|l\|} \hline-0.0004 \\ -1.7593 \end{array}$ | $\begin{aligned} & -0.0004 \\ & : \\ & : \\ & -10.0000 \end{aligned}$ |  | $\begin{aligned} & 3.9994 \\ & 1.1858 \end{aligned}$ |  | -1 <br> -4864 -27648 | $\begin{aligned} & \text { FFFF } \\ & \text { ED00 } \\ & : \\ & 9400 \end{aligned}$ |
|  |  | $\begin{aligned} & -10.0004 \\ & : \\ & -11.7589 \end{aligned}$ |  |  |  | -27649 <br> -32512 | $\begin{aligned} & 93 \mathrm{FF} \\ & : \\ & 8100 \end{aligned}$ |
| Underflow | <-1.7593 | <-11.7589 | <0.0000 | <1.1858 |  | -32768 | 8000 |

Input ranges resistance temperature detector
The resolution corresponds to 16 bits.

| Range | $\begin{array}{\|l\|} \hline \mathrm{Pt} 100 / \mathrm{Pt} \\ 1000 \\ -50^{\circ} \mathrm{C} \ldots+70 \\ { }^{\circ} \mathrm{C} \end{array}$ | $\begin{aligned} & \hline \mathrm{Pt1001} \\ & \mathrm{Pt} 1000 \\ & -50^{\circ} \mathrm{C} \ldots \\ & +400^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \mathrm{Ni} 1000 \\ & -50^{\circ} \mathrm{C} \ldots \\ & +150^{\circ} \mathrm{C} \end{aligned}$ | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  | Decimal | Hex. |
| Overflow | $>+80.0^{\circ} \mathrm{C}$ | $>+450.0{ }^{\circ} \mathrm{C}$ | $>+160.0{ }^{\circ} \mathrm{C}$ | 32767 | 7FFF |
| Measured value too high |  | $\begin{aligned} & +450.0^{\circ} \mathrm{C} \\ & : \\ & +400.1^{\circ} \mathrm{C} \end{aligned}$ |  | $\begin{aligned} & 4500 \\ & : \\ & 4001 \end{aligned}$ | $\begin{aligned} & 1194 \\ & : \\ & \text { OFA1 } \end{aligned}$ |
|  |  |  | $\begin{aligned} & +160.0^{\circ} \mathrm{C} \\ & : \\ & +150.1^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & 1600 \\ & : \\ & 1501 \end{aligned}$ | $\begin{aligned} & 0640 \\ & : \\ & \text { 05DD } \end{aligned}$ |
|  | $+80.0^{\circ} \mathrm{C}$ <br> $+70.1^{\circ} \mathrm{C}$ |  |  | $\begin{aligned} & 800 \\ & : \\ & 701 \\ & \hline \end{aligned}$ | $\begin{array}{\|l} \hline 0320 \\ : \\ \text { 02BD } \\ \hline \end{array}$ |


| Range | $\begin{array}{\|l} \mathrm{Pt} 100 / \mathrm{Pt} \\ 1000 \\ -50^{\circ} \mathrm{C} \ldots+70 \\ { }^{\circ} \mathrm{C} \end{array}$ | $\begin{aligned} & \mathrm{Pt} 100 \mathrm{I} \\ & \mathrm{Pt} 1000 \\ & -50^{\circ} \mathrm{C} \ldots \\ & +400^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \mathrm{Ni} 1000 \\ & -50^{\circ} \mathrm{C} \ldots \\ & +150^{\circ} \mathrm{C} \end{aligned}$ | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  | Decimal | Hex. |
| Normal range | $\begin{aligned} & +70.0^{\circ} \mathrm{C} \\ & : \\ & +0.1^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & +400.0^{\circ} \mathrm{C} \\ & \vdots \\ & : \\ & +0.1^{\circ} \mathrm{C} \end{aligned}$ | $+150.0^{\circ} \mathrm{C}$ $+0.1^{\circ} \mathrm{C}$ | $\begin{aligned} & 4000 \\ & 1500 \\ & 700 \\ & : \\ & 1 \end{aligned}$ | $\begin{aligned} & \hline \text { OFAO } \\ & \text { 05DC } \\ & \text { 02BC } \\ & : \\ & 0001 \end{aligned}$ |
|  | $0.0{ }^{\circ} \mathrm{C}$ | $0.0{ }^{\circ} \mathrm{C}$ | $0.0{ }^{\circ} \mathrm{C}$ | 0 | 0000 |
|  | $-0.1^{\circ} \mathrm{C}$ <br> $-50.0^{\circ} \mathrm{C}$ | $-0.1^{\circ} \mathrm{C}$ $-50.0^{\circ} \mathrm{C}$ | $-0.1^{\circ} \mathrm{C}$ $-50.0^{\circ} \mathrm{C}$ | $\begin{array}{\|l} \hline-1 \\ : \\ -500 \\ \hline \end{array}$ | $\begin{aligned} & \text { FFFF } \\ & \text { : } \\ & \text { FEOC } \end{aligned}$ |
| Measured value too low | $-50.1^{\circ} \mathrm{C}$ <br> $-60.0^{\circ} \mathrm{C}$ | $-50.1^{\circ} \mathrm{C}$ <br> $-60.0^{\circ} \mathrm{C}$ | $-50.1^{\circ} \mathrm{C}$ <br> $-60.0^{\circ} \mathrm{C}$ | $\begin{aligned} & -501 \\ & : \\ & -600 \end{aligned}$ | FEOB $:$ FDA8 |
| Underflow | <-60.0 ${ }^{\circ} \mathrm{C}$ | <-60.0 ${ }^{\circ} \mathrm{C}$ | <-60.0 ${ }^{\circ} \mathrm{C}$ | -32768 | 8000 |

## Technical data

## Technical data of the module

The system data of AC500 and S500 are applicable to the standard version ${ }^{\leftrightarrow}$ Chapter 5.1.2 "System data AC500" on page 166.

The system data of AC500-XC are applicable to the XC version $\stackrel{\Perp}{ }{ }^{\Perp}$ Chapter 5.1.3 "System data AC500-XC" on page 169.

Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter | Value |
| :---: | :---: |
| Process voltage |  |
| Connections | Terminals $1.8,2.8,3.8$ and 4.8 for +24 V (UP) as well as 1.9, 2.9, 3.9 and 4.9 for 0 V (ZP) |
| Rated value | 24 V DC |
| Max. ripple | 5 \% |
| Protection against reversed voltage | Yes |
| Rated protection fuse on UP | 10 A fast |
| Galvanic isolation | Yes, per module |
| Current consumption |  |
| From 24 V DC power supply at the terminals UP/L+ and ZP/M of the CPU/communication interface module | Ca. 2 mA |
| From UP at normal operation / with outputs | 0.15 A + output loads |


| Parameter | Value |
| :--- | :--- |
| Inrush current from UP (at power up) | $0.050 \mathrm{~A}^{2} \mathrm{~s}$ |
| Max. length of analog cables, conductor cross section <br> $>0.14 \mathrm{~mm}^{2}$ | 100 m |
| Weight | 300 g |
| Mounting position | Horizontal or vertical with derating <br> (output load reduced to $50 \%$ at +40 <br> ${ }^{\circ} \mathrm{C}$ per group) |
| Cooling | The natural convection cooling must <br> not be hindered by cable ducts or <br> other parts in the control cabinet. |

## NOTICE!

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and temporary overvoltage up to 30 V DC.

## Technical data of the analog inputs

| Parameter | Value |  |
| :---: | :---: | :---: |
| Number of channels per module | 16 |  |
| Distribution of channels into groups | 2 groups of 8 channels each |  |
| Connections of the channels I0- ... I7Connections of the channels IO+ ... I7+ | Terminals 1.0 ... 1.7 <br> Terminals 2.0 ... 2.7 |  |
| Connections of the channels I8- ... I15- <br> Connections of the channels I8+ ... I15+ | Terminals 3.0 ... 3.7 Terminals 4.0 ... 4.7 |  |
| Input type | Bipolar (not with current or Pt100/ Pt1000/ Ni1000) |  |
| Galvanic isolation | Against internal supply and other modules |  |
| Configurability | $0 \vee \ldots 10 \mathrm{~V},-10 \mathrm{~V} \ldots+10 \mathrm{~V}, 0 / 4 \mathrm{~mA} . . .20 \mathrm{~mA}$, Pt100/1000, Ni1000 (each input can be configured individually) |  |
| Channel input resistance | Voltage: > $100 \mathrm{k} \Omega$ <br> Current: ca. $330 \Omega$ |  |
| Time constant of the input filter | Voltage: $100 \mu \mathrm{~s}$ Current: $100 \mu \mathrm{~s}$ |  |
| Indication of the input signals | 1 LED per channel |  |
| Conversion cycle | 2 ms (for 16 inputs), with Pt/Ni... 1 s |  |
| Resolution | Range $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ : 12 bits |  |
|  | Range -10 V ... +10 V: 12 bits including sign |  |
|  | Range $0 \mathrm{~mA} . . .20 \mathrm{~mA}$ : 12 bits |  |
|  | Range $4 \mathrm{~mA} . . .20 \mathrm{~mA}$ : 12 bits |  |
| Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range | Typ. | $\pm 0.5 \%$ of full scale at $+25^{\circ} \mathrm{C}$ |


| Parameter | Value |
| :--- | :--- |
|  | Max.$\pm 1 \%$ of full scale (all ranges) <br> at $0{ }^{\circ} \mathrm{C} \ldots+60^{\circ} \mathrm{C}$ or EMC disturbance |
| Relationship between input signal and hex <br> code | そ Chapter 5.2.4.4.2.2.10.2 "Input ranges <br> resistance temperature detector" on page 720 |
| Unused voltage inputs | Are configured as "unused" |
| Unused current inputs | Have a low resistance, can be left open- <br> circuited |
| Overvoltage protection | Yes |

## Technical data of the analog inputs, if used as digital inputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | Max. 16 |
| Distribution of channels into groups | 2 groups of 8 channels each |
| Connections of the channels 10+ ... 17+ Connections of the channels I8+ ... I15+ | $\begin{aligned} & \text { Terminals } 2.0 \ldots 2.7 \\ & \text { Terminals } 4.0 \ldots 4.7 \end{aligned}$ |
| Reference potential for the inputs | Terminals 1.9, 2.9, 3.9 and 4.9 (ZP) |
| Input signal delay | Typ. 8 ms , configurable from $0.1 \ldots 32 \mathrm{~ms}$ |
| Indication of the input signals | 1 LED per channel |
| Input signal voltage | 24 V DC |
| Signal 0 | -30 V ... +5V |
| Undefined signal | +5V ... +13 V |
| Signal 1 | +13V ... +30 V |
| Input current per channel |  |
| Input voltage +24 V | Typ. 7 mA |
| Input voltage +5 V | Typ. 1.4 mA |
| Input voltage +15 V | Typ. 4.3 mA |
| Input voltage +30 V | < 9 mA |
| Input resistance | Ca. $3.5 \mathrm{k} \Omega$ |

## Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm

The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 250 300 R0001 | Al523, analog input module, 16 AI, <br> U/I/Pt100, 12 bits including sign, <br> 2-wires | Active |
| 1SAP 450 300 R0001 | Al523-XC, analog input module, 16 AI, <br> U/I/Pt100, 12 bits including sign, <br> 2-wires, XC version | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

## Al531-Analog input module

## Features

- 8 configurable analog inputs ( $10 \ldots 17$ ) in 2 groups ( 1.0 ... 1.7 and $2.0 \ldots 2.7$ as well as 3.0 ... 3.7 and 4.0 ... 4.7)

Resolution 16 bits including sign

- Module-wise galvanically isolated
- XC version for use in extreme ambient conditions available


1 I/O bus
2 Allocation between terminal number and signal names
34 yellow LEDs to display the states at the inputs $10 \ldots$ I3
44 yellow LEDs to display the states at the inputs $14 \ldots$... 7
51 green LED to display the process supply voltage UP
62 red LEDs to display errors (CH-ERR2 and CH-ERR4)
7 Label
8 Terminal unit
9 DIN rail
${ }_{*}^{*}+\underset{*}{*}$ Sign for XC version

## Intended purpose

The device can be used as a decentralized I/O extension module for S500 communication interface modules (e. g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs.

## Functionality

8 analog inputs, individually configurable for

- Unused (default setting)
- 0 V ... $5 \mathrm{~V}, 0 \mathrm{~V}$... 10 V
- $-50 \mathrm{mV} \ldots+50 \mathrm{mV},-500 \mathrm{mV}$... +500 mV
- $-1 \mathrm{~V} \ldots+1 \mathrm{~V},-5 \mathrm{~V} \ldots+5 \mathrm{~V},-10 \mathrm{~V} \ldots+10 \mathrm{~V}$
- $0 \mathrm{~mA} . . .20 \mathrm{~mA}$
- $4 \mathrm{~mA} . .20 \mathrm{~mA}$
- $-20 \mathrm{~mA} \ldots 20 \mathrm{~mA}$
- Pt100, $-50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ or $+400^{\circ} \mathrm{C}$ (2-, 3- and 4 -wire)
- Pt100, $-200^{\circ} \mathrm{C} \ldots+850^{\circ} \mathrm{C}(2-, 3-$ and 4 -wire $)$
- Pt1000, $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ (2-, 3- and 4-wire)
- Ni1000, $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ (2-, 3- and 4-wire)
- $\operatorname{Cu50}$ (1.426): $-50^{\circ} \mathrm{C} \ldots+200^{\circ} \mathrm{C}$ (2-, 3- and 4 -wire)
- Cu50 (1.428): $-200^{\circ} \mathrm{C} \ldots+200^{\circ} \mathrm{C}$ (2-, 3- and 4-wire)
- $0 \Omega \ldots 50 \mathrm{k} \Omega$
- Thermocouples of types J, K, T, N, S
- Resistance measuring bridge
- Digital signals (digital input)

| Parameter |  |
| :--- | :--- |
| Vesolution of the analog channels |  |
|  | Voltage and current |
| Temperature | 16 bits including sign |
| LED displays | $+0.1^{\circ} \mathrm{C}\left(0,01^{\circ} \mathrm{C}\right.$ at $\left.\mathrm{Pt} 100-50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}\right)$ |
| Internal power supply | 11 LEDs for signals and error messages |
| External power supply | through the I/O bus interface (I/O bus) |
| Required terminal unit | via terminals (process voltage UP = 24 V DC$)$ |

## Connections

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

The modules are plugged on an I/O terminal unit $\Leftrightarrow$ Chapter 5.2.5.2 "TU515, TU516, TU541 and TU542 for I/O modules" on page 938. Properly position the modules and press until they lock in place. The terminal units are mounted on a DIN rail or with 2 screws plus the additional accessory for wall mounting ${ }^{\star} \Rightarrow$ Chapter 5.2.8.2.5 "TA526-Wall mounting accessory" on page 1324.
The connection of the I/O channels is carried out using the 40 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

The terminals $1.8,2.8,3.8,4.8,1.9,2.9,3.9$ and 4.9 are electrically interconnected within the I/O terminal units and always have the same assignment, independent of the inserted module:

Terminals 1.8, 2.8, 3.8 and 4.8: process voltage UP $=+24 \mathrm{~V}$ DC
Terminals 1.9, 2.9, 3.9 and 4.9: process voltage $\mathrm{ZP}=0 \mathrm{~V}$
The assignment of the other terminals:

| Terminals | Signal | Description |
| :--- | :--- | :--- |
| $2.0,2.2,2.4,2.6$ | IO+ .. I3+ | Positive poles of the first 4 <br> analog inputs |
| $1.0,1.2,1.4,1.6$ | IO- ... I3- | Negative poles of the first 4 <br> analog inputs |
| $2.1,2.3,2.5,2.7$ | IOA ... I3A | Connections A (supply) of the <br> first 4 analog inputs |
| $1.1,1.3,1.5,1.7$ | IOB ... I3B | Connections B (analog <br> ground) of the first 4 analog <br> inputs |
| $4.0,4.2,4.4,4.6$ | I4+ .. I7+ | Positive poles of the following <br> 4 analog inputs |
| $3.0,3.2,3.4,3.6$ | I4- I7- $\ldots$ I7A | Negative poles of the fol- <br> lowing 4 analog inputs |
| $4.1,4.3,4.5,4.7$ | Connections A (supply) of the <br> following 4 analog inputs |  |
| $3.1,3.3,3.5,3.7$ | Connections B (analog <br> ground) of the following 4 <br> analog inputs |  |



The " $1 x B$ " clamps ( $x=0 \ldots 7$ ) of the analog inputs are galvanically connected to each other. They form an "Analog Ground Signal" (AGND) for the module.

The negative poles of the analog inputs Ix-may accept a potential difference up to $\pm 20$ V DC with regard to the common reference potential IxB (AGND, $Z P$ ). Observing this maximum voltage difference, analog current inputs of one module can be switched in series to each other and also with current inputs of other modules.

For the open-circuit detection (wire break), each positive analog input channel Ix+ is pulled up to "plus" by a high-resistance resistor and each negative analog input channel Ix- is pulled down to "minus" by a resistor. If wire break occurs, a maximum voltage (overflow or underflow) will be read in then.

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a communication interface module or a CPU). Thus, the current consumption from 24 V DC power supply at the terminals L+/UP and M/ZP of the CPU/communication interface module increases by 2 mA per AI531.

The external power supply connection is carried out via the UP (+24 V DC) and the ZP (0 V DC) terminals.

## WARNING!

## Removal/Insertion under power

Removal or insertion under power is permissible only if all conditions for hot swapping are fullfilled.
(4) Chapter 4.9.3 "Replace an I/O module with hot swap" on page 153

The devices are not designed for removal or insertion under power when the conditions for hot swap do not apply. Because of unforeseeable consequences, it is not allowed to plug in or unplug devices with the power being ON.
Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while they are energized in a hazardous location could result in an electric arc, which could create an ignition source resulting in fire or explosion.
Prior to proceeding, make sure that power is been disconnected and that the area has been thoroughly checked to ensure that flammable materials are not present.

The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

## Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

Generally, analog signals must be laid in shielded cables. The cable shields must be grounded at both sides of the cables. In order to avoid unacceptable potential differences between different parts of the installation, low resistance equipotential bonding conductors must be laid.


Fig. 99: 8 analog inputs in two groups, individually configurable ${ }^{\mu}$ Chapter 5.2.4.4.2.3.3 "Functionality" on page 726

## CAUTION!

By installing equipotential bonding conductors between the different parts of the system, it must be ensured that the potential difference between ZP and AGND never can exceed 1 V .

## CAUTION!

The process supply voltage must be included in the grounding concept (e. g. grounding of the negative terminal).

The module provides several diagnosis functions $\#$ Chapter 5.2.4.4.2.3.8 "Diagnosis" on page 753 .

## Connection of active-type analog sensors (Voltage) with galvanically isolated power supply Standard ranges



Fig. 100: Connection example
The measuring ranges can be configured ${ }^{\mu}$, Chapter 5.2.4.4.2.3.7 "Parameterization" on page 749:

| Voltage | $-50 \mathrm{mV} \ldots+50 \mathrm{mV}$ | 1 channel used |
| :--- | :--- | :--- |
| Voltage | $-500 \mathrm{mV} \ldots+500 \mathrm{mV}$ | 1 channel used |
| Voltage | $-1 \mathrm{~V} \ldots+1 \mathrm{~V}$ | 1 channel used |
| Voltage | $-5 \mathrm{~V} \ldots+5 \mathrm{~V}$ | 1 channel used |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | 1 channel used |
| Voltage | $0 \mathrm{~V} \ldots+5 \mathrm{~V}$ | 1 channel used |
| Voltage | $0 \mathrm{~V} \ldots+10 \mathrm{~V}$ | 1 channel used |



Fig. 101: Connection example
The measuring range can be configured ${ }^{\mu}$ Chapter 5.2.4.4.2.3.7 "Parameterization" on page 749:

| Voltage | Common mode voltage | 1 channel used |
| :--- | :--- | :--- |

The function of the LEDs is described under diagnosis and displays/displays $\Rightarrow$ Chapter 5.2.4.4.2.3.8 "Diagnosis" on page 753.

In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as "unused".

Connection of active-type analog sensors (Voltage) with no galvanically isolated power supply
Standard ranges


Fig. 102: Connection example
CAUTION!
If GND is not directly connected to ZP at the sensor, the supply current flows
via the GND line to ZP. Measuring errors can only occur caused by voltage
differences higher than $\pm 20 \mathrm{~V}$ DC between GND and ZP .

The measuring ranges can be configured ${ }^{*}$ Chapter 5.2.4.4.2.3.7 "Parameterization" on page 749 :

| Voltage | $-50 \mathrm{mV} \ldots+50 \mathrm{mV}$ | 1 channel used |
| :--- | :--- | :--- |
| Voltage | $-500 \mathrm{mV} \ldots+500 \mathrm{mV}$ | 1 channel used |
| Voltage | $-1 \mathrm{~V} \ldots+1 \mathrm{~V}$ | 1 channel used |
| Voltage | $-5 \mathrm{~V} \ldots+5 \mathrm{~V}$ | 1 channel used |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | 1 channel used |
| Voltage | $0 \mathrm{~V} \ldots+5 \mathrm{~V}$ | 1 channel used |
| Voltage | $0 \mathrm{~V} \ldots+10 \mathrm{~V}$ | 1 channel used |



Fig. 103: Connection example

## CAUTION!

If GND is not directly connected to ZP at the sensor, the supply current flows via the GND line to ZP . Measuring errors can only occur caused by voltage differences higher than $\pm 20 \mathrm{~V}$ DC between GND and ZP.

The measuring range can be configured ${ }^{\mu}$ Chapter 5.2.4.4.2.3.7 "Parameterization" on page 749 :

| Voltage | Common mode voltage | 1 channel used |
| :--- | :--- | :--- |

The function of the LEDs is described under diagnosis and displays/displays $\Rightarrow$ Chapter 5.2.4.4.2.3.8 "Diagnosis" on page 753.

In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as "unused".

## Connection of active-type analog sensors (Current) with galvanically isolated power supply



Fig. 104: Connection example
The following measuring ranges can be configured ${ }^{\Leftrightarrow}$ Chapter 5.2.4.4.2.3.7 "Parameterization" on page 749:

| Current | $-20 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 1 channel used |
| :--- | :--- | :--- |
| Current | $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 1 channel used |
| Current | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 1 channel used |

The function of the LEDs is described under diagnosis and displays/displays ${ }_{\mu}^{\mu}$ Chapter 5.2.4.4.2.3.8 "Diagnosis" on page 753.

Unused input channels can be left open, because they are of low resistance.

## Connection of active-type analog sensors (Current) with galvanically isolated power supply and seriesconnection of an additional input



Fig. 105: Connection example
1 Analog input of the second device

If series-connection of an additional input is used, the input resistance of the module (ca. $330 \Omega$ ) must be added to the input resistance of the second device. Make sure that the maximum permitted load resistance of the analog sensor is not exceeded (see the data sheet of the analog sensor).

The input of the module is not related to ZP. If the input of the second device is related to $Z P$, the order of sequence in the series-connection must be observed by all means (from the sensor to the module and then to the input of the second device).

The following measuring ranges can be configured ${ }^{y} y$ Chapter 5.2.4.4.2.3.7 "Parameterization" on page 749:

| Current | $-20 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 1 channel used |
| :--- | :--- | :--- |
| Current | $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 1 channel used |
| Current | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 1 channel used |

For a description of the functions of the LEDs, please refer to diagnosis and displays/displays (y) Chapter 5.2.4.4.2.3.8 "Diagnosis" on page 753.

Unused input channels can be left open, because they are of low resistance.

## Connection of passive-type analog sensors (Current)



Fig. 106: Connection example
The following measuring ranges can be configured $\Leftrightarrow$ Chapter 5.2.4.4.2.3.7 "Parameterization" on page 749:

| Current | $-20 \mathrm{~mA} \ldots 20 \mathrm{~mA}{ }^{*}$ ) | 1 channel used |
| :--- | :--- | :--- |
| Current | $0 \mathrm{~mA} \ldots 20 \mathrm{~mA} *$ ) | 1 channel used |
| Current | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 1 channel used |
| ${ }^{*}$ ) This setting is not applicable with passive-type analog sensors (current). |  |  |

The function of the LEDs is described under diagnosis and displays/displays $\Rightarrow$ Chapter 5.2.4.4.2.3.8 "Diagnosis" on page 753.

Unused input channels can be left open, because they are of low resistance.


Fig. 107: Connection example
1 Analog input of the second device

If series-connection of an additional input is used, the input resistance of the module (ca. $330 \Omega$ ) must be added to the input resistance of the second device. Make sure that the maximum permitted load resistance of the analog sensor is not exceeded (see the data sheet of the analog sensor).

The input of the module is not related to ZP. If the input of the second device is related to $Z P$, the order of sequence in the series-connection must be observed by all means (from the sensor to the module and then to the input of the second device).

The following measuring ranges can be configured $\leadsto$ Chapter 5.2.4.4.2.3.7 "Parameterization" on page 749:

| Current | $-20 \mathrm{~mA} \ldots 20 \mathrm{~mA} *)$ | 1 channel used |
| :--- | :--- | :--- |
| Current | $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ *) | 1 channel used |
| Current | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 1 channel used |
| ${ }^{*}$ ) This setting is not applicable with passive-type analog sensors (current). |  |  |

The function of the LEDs is described under diagnosis and displays/displays $\Leftrightarrow$ Chapter 5.2.4.4.2.3.8 "Diagnosis" on page 753.

Unused input channels can be left open, because they are of low resistance.

## Connection of digital signal sources at analog inputs

Several (or all) analog inputs can be configured as digital inputs. The inputs are not galvanically isolated against the other analog channels.


Fig. 108: Connection example
The following operating mode can be configured ${ }^{\wedge}$ Chapter 5.2.4.4.2.3.7 "Parameterization" on page 749 :

| Digital input | 24 V | 1 channel used |
| :--- | :--- | :--- |
| Effect of incorrect input ter- <br> minal connection |  | Wrong or no signal detected, <br> no damage up to 35 V |

For a description of the function of the LEDs, please refer to diagnosis and displays/displays « Chapter 5.2.4.4.2.3.8 "Diagnosis" on page 753.

## Connection of resistance thermometers in 2-wire configuration

When resistance thermometers (Pt100, Pt1000, Ni1000, Cu50) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the module AI531 provides a constant current source which is multiplexed over the 4 analog channels.


Fig. 109: Connection example
The following measuring ranges can be configured ${ }^{\Leftrightarrow}$ Chapter 5.2.4.4.2.3.7 "Parameterization" on page 749:
$\left.\begin{array}{|l|l|l|}\hline \text { Pt100 } & -50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C} /+400^{\circ} \mathrm{C} ; & 1 \text { channel used } \\ -200^{\circ} \mathrm{C} \ldots+850^{\circ} \mathrm{C}\end{array}\right]$

For a description of the function of the LEDs, please refer to Diagnosis and displays / displays Chapter 5.2.4.4.2.3.8 "Diagnosis" on page 753.
The module linearizes the resistance thermometer characteristics.
In order to avoid error messages from unused analog input channels, it is useful to configure them as "unused".

## Connection of resistance thermometers in 3-wire configuration

When resistance thermometers (Pt100, Pt1000, Ni1000, Cu50) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the module AI531 provides a constant current source which is multiplexed over the 4 analog channels.


Fig. 110: Connection example
The following measuring ranges can be configured ${ }^{\Perp}$ Chapter 5.2.4.4.2.3.7 "Parameterization" on page 749 :

| Pt100 | $-50^{\circ} \mathrm{C} \ldots+70{ }^{\circ} \mathrm{C} /+400^{\circ} \mathrm{C} ;$ | 1 channel used |
| :--- | :--- | :--- |
|  | $-200^{\circ} \mathrm{C} \ldots+850{ }^{\circ} \mathrm{C}$ |  |
| Pt1000 | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 1 channel used |
| Ni1000 | $-50^{\circ} \mathrm{C} \ldots+150{ }^{\circ} \mathrm{C}$ | 1 channel used |
| Cu50 | $-50^{\circ} \mathrm{C} \ldots+200^{\circ} \mathrm{C}(1.426) ;$ | 1 channel used |
|  | $-200^{\circ} \mathrm{C} \ldots+200^{\circ} \mathrm{C}(1.428)$ |  |

For a description of the function of the LEDs, please refer to diagnosis and displays/displays «y Chapter 5.2.4.4.2.3.8 "Diagnosis" on page 753.
The module linearizes the resistance thermometer characteristics. In order to keep measuring errors as small as possible, it is necessary by all means to have all the involved conductors in the same cable. All the conductors must have the same cross section.
In order to avoid error messages from unused analog input channels, it is useful to configure them as "unused".

## Connection of resistance thermometers in 4-wire configuration

When resistance thermometers (Pt100, Pt1000, Ni1000, Cu50) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the module AI531 provides a constant current source which is multiplexed over the 4 analog channels.


Fig. 111: Connection example
The following measuring ranges can be configured ${ }^{«}$, Chapter 5.2.4.4.2.3.7 "Parameterization" on page 749:
$\left.\begin{array}{|l|l|l|}\hline \text { Pt100 } & -50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C} /+400^{\circ} \mathrm{C} ; & 1 \text { channel used } \\ -200^{\circ} \mathrm{C} \ldots+850^{\circ} \mathrm{C}\end{array}\right]$

For a description of the function of the LEDs, please refer to diagnosis and displays/displays * Chapter 5.2.4.4.2.3.8 "Diagnosis" on page 753.

The module linearizes the resistance thermometer characteristics. In order to keep measuring errors as small as possible, it is necessary by all means, to have all the involved conductors in the same cable.

In order to avoid error messages from unused analog input channels, it is useful to configure them as "unused".

## Connection of resistors in 2-wire configuration

For evaluating resistors, a constant current must flow through them to build the necessary voltage drop. For this, the module Al531 provides a constant current source which is multiplexed over the 4 analog channels.


Fig. 112: Connection example
The following measuring ranges can be configured $\#$ Chapter 5.2.4.4.2.3.7 "Parameterization" on page 749 :

| Resistor | $50 \mathrm{k} \Omega$ | 1 channel used |
| :--- | :--- | :--- |

For a description of the function of the LEDs, please refer to diagnosis and displays/displays * Chapter 5.2.4.4.2.3.8 "Diagnosis" on page 753.

In order to avoid error messages from unused analog input channels, it is useful to configure them as "unused".

## Connection of a resistance measuring bridge with internal supply

When resistance measuring bridges are connected, the short-circuit-proof voltage output (internal supply) at pin 10 A (or $12 \mathrm{~A}, 14 \mathrm{~A}, 16 \mathrm{~A}$ ) must be used. This supply voltage is activated as soon as "Voltage Measurement" is configured for the relevant channel.


Fig. 113: Connection example
1 Internal supply
All voltage measuring ranges can be configured $\stackrel{y}{ }{ }^{\wedge}$ Chapter 5.2.4.4.2.3.7 "Parameterization" on page 749.

| Voltage | $-50 \mathrm{mV} \ldots+50 \mathrm{mV}$ | 1 channel used |
| :--- | :--- | :--- |
| Voltage | $-500 \mathrm{mV} \ldots+500 \mathrm{mV}$ | 1 channel used |
| Voltage | $-1 \mathrm{~V} \ldots+1 \mathrm{~V}$ | 1 channel used |
| Voltage | $-5 \mathrm{~V} \ldots+5 \mathrm{~V}$ | 1 channel used |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | 1 channel used |
| Voltage | $0 \mathrm{~V} \ldots+5 \mathrm{~V}$ | 1 channel used |
| Voltage | $0 \mathrm{~V} \ldots+10 \mathrm{~V}$ | 1 channel used |

The calculation of the resistor deviation must be performed via the bridge voltage by the PLC user program.

## Connection of a resistance measuring bridge with external supply

With the connection of a resistance measuring bridge with external supply, the supply voltage is provided separately.


Fig. 114: Connection example
1 Bridge to IxB necessary with galvanically isolated supply
All voltage measuring ranges can be configured $\stackrel{y}{ }$, Chapter 5.2.4.4.2.3.7 "Parameterization" on page 749 .

| Voltage | $-50 \mathrm{mV} \ldots+50 \mathrm{mV}$ | 1 channel used |
| :--- | :--- | :--- |
| Voltage | $-500 \mathrm{mV} \ldots+500 \mathrm{mV}$ | 1 channel used |
| Voltage | $-1 \mathrm{~V} \ldots+1 \mathrm{~V}$ | 1 channel used |
| Voltage | $-5 \mathrm{~V} \ldots+5 \mathrm{~V}$ | 1 channel used |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | 1 channel used |
| Voltage | $0 \mathrm{~V} \ldots+5 \mathrm{~V}$ | 1 channel used |
| Voltage | $0 \mathrm{~V} \ldots+10 \mathrm{~V}$ | 1 channel used |

The calculation of the resistor deviation must be performed via the bridge voltage by the PLC user program.

## Connection of thermocouples



Fig. 115: Connection example
The following measuring ranges can be configured ${ }^{\mu}$ Chapter 5.2.4.4.2.3.7 "Parameterization" on page 749 :

| J type | $-210^{\circ} \mathrm{C} \ldots+1200^{\circ} \mathrm{C}$ | Fe-CuNi | 1 channel used |
| :--- | :--- | :--- | :--- |
| K type | $-270^{\circ} \mathrm{C} \ldots+1372^{\circ} \mathrm{C}$ | $\mathrm{Ni}-\mathrm{CrNi}$ | 1 channel used |
| N type | $-270^{\circ} \mathrm{C} \ldots+1300^{\circ} \mathrm{C}$ | $\mathrm{NiCrSi-NiSi}$ | 1 channel used |
| S type | $-50^{\circ} \mathrm{C} \ldots+1768^{\circ} \mathrm{C}$ | $\mathrm{Pt} 10 R \mathrm{Rh}-\mathrm{Pt}$ | 1 channel used |
| T type | $-270^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | $\mathrm{Cu}-\mathrm{CuNi}$ | 1 channel used |

For a description of the function of the LEDs, please refer to diagnosis and displays/displays *) Chapter 5.2.4.4.2.3.8 "Diagnosis" on page 753.
The module linearizes the thermocouple characteristics. It supports the following possibilities of temperature compensation and handling with cold junctions:

## Internal compensation

An internal temperature sensor which is located next to the terminal unit is used to detect the temperature of the cold junction. So the compensating cables must be connected directly to the terminal unit, where the cold junction is located.

The setting "Internal compensation (default)" for the parameter "Compensation channel" should be selected.

To get more precise temperature measurements, the use of an external compensation method is recommended.

## External compensation with temperature input

The temperature for the cold junction can be determinated externally.
A measured or known temperature value (e.g. ambient temperature in the cabinet) is transferred to the module via the output data word to all required channels. The possible temperature range is $-25^{\circ} \mathrm{C} \ldots+60^{\circ} \mathrm{C}$ and is monitored by the AI531.

The setting "External with temperature value" for the parameter "Compensation channel" should be selected.

## External compensation with compensation box

A compensation box balances the temperature difference between the cold junction and the reference temperature by generating a bridge voltage. The reference temperature is transferred via the output data word.
The compensation box must fit to the type of thermocouple and is located at the end of the compensating cables, where the cold junction is located. The cabling to the AI531 can be carried out with normal cables. The operating manual of the compensation box also has to be considered.

The setting "External with temperature value" for the parameter "Compensation channel" should be selected.

## External compensation with flanking channel

A flanking channel of the same input group can be used for compensation, e. g. for channel 3 , the channels 0,1 and 2 can be selected as reference channels. The type of sensor for the reference channel can be selected in the parameters for the flanking channel. For example, a RTD sensor which is located next to the thermocouple terminal can be used as reference point for other channels.
The setting "Channel $x$ " for the parameter "Compensation channel" should be selected. Refer to 'Channel configuration' for possible settings $\Rightarrow$ Chapter 5.2.4.4.2.3.7 "Parameterization" on page 749.
In order to avoid error messages from unused analog input channels, it is useful to configure them as "unused".

## Internal data exchange

| Digital inputs (bytes) | 0 |
| :--- | :--- |
| Digital outputs (bytes) | 0 |
| Analog inputs (words) | 8 |
| Analog outputs (words) | 1 |

## I/O configuration

The module does not store configuration data itself. It gets its parameterization data from the master device of the I/O bus (CPU or communication interface module) during power-up of the system.
This means that replacing I/O modules is possible without any re-parameterization via software.

$$
\begin{aligned}
& \text { If the external power supply voltage via UP/ZP terminals fails, the I/O module } \\
& \text { loses its configuration data. The whole station has to be switched off and on } \\
& \text { again to re-configure the module. }
\end{aligned}
$$

## Parameterization

| Firmware version | Configuration |
| :--- | :--- |
| Firmware version > V2.0.0 | The arrangement of the parameter data is per- <br> formed by Control Builder Plus/ Automation <br> Builder software. |

The parameter data directly influences the functionality of modules.
For non-standard applications, it is necessary to adapt the parameters to your system configuration.

Module: Module slot address: Y = 1 ... 10

| Name | Value | Internal value | Internal value, Type | Default | Min. | Max. | EDS Slot/ Index |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Module ID | Internal | $1535$ <br> ${ }^{1}$ ) | Word | $\begin{aligned} & 1535 \\ & 0 \times 05 f f \end{aligned}$ | 0 | 65535 | 0x0Y01 |
| Ignore module ${ }^{2}$ ) | $\begin{aligned} & \text { No } \\ & \text { Yes } \end{aligned}$ | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | Byte | $\begin{array}{\|l\|} \hline \text { No } \\ 0 \times 00 \end{array}$ |  |  | Not for FBP |
| Parameter length in bytes | Internal | 36 | Byte | 36 | 0 | 255 | 0x0Y02 |
| Check supply | $\begin{aligned} & \text { Off } \\ & \text { On } \end{aligned}$ | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | Byte | $\begin{aligned} & \text { On } \\ & 0 \times 01 \end{aligned}$ |  |  | 0x0Y03 |
| Analog data format | Default | 0 | Byte | Default $0 \times 00$ |  |  | 0x0Y04 |

[^13]${ }^{2}$ ) Not with FBP
GSD file:

| Ext_User_Prm_Data_Len = | 39 |
| :---: | :---: |
| Ext_User_Prm_Data_Const(0) = | 0x05, 0xff, 0x24, 1 |
|  | 0x01, 0x00, 0x00, 0x00 \ |
|  | $\begin{aligned} & 0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00 \text {, } \\ & 0 \times 00,1 \end{aligned}$ |
|  | $\begin{aligned} & 0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00 \\ & 0 \times 00,1 \end{aligned}$ |
|  | $\begin{aligned} & 0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00 \text {, } \\ & 0 \times 00,1 \end{aligned}$ |
|  | $\begin{aligned} & 0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00 \\ & 0 \times 00 \end{aligned}$ |

Input channel (8x)

| No. | Name | Value | Internal value | Internal value, Type | Default | EDS Slot Index |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 1 | Channel configuration | see table 'Channel configuration' (4) Table 167 " Channel configuration" on page 751 | see <br> 'Channel configuration' (4) Table 1 67 "Chan nel configuration" on page 75 | Byte | $\begin{array}{\|l} \hline 0 \\ 0 \times 00 \end{array}$ | 0x0Y07 |
| 2 | Channel monitoring | see table 'Channel monitoring' (4) Table 168 Channel monitoring" on page 752 | see 'Channel monitoring' cs) Table 1 68 "Chan nel monitoring" on page 75 | Byte | $\begin{array}{\|l} \hline 0 \\ 0 \times 03 \end{array}$ |  |
| 3 | Line frequency suppression | see table 'Line frequency suppression' «) Further information on page 752 | see 'Line frequency suppression' \& Further information on page 752 | Byte <br> 2 | $\begin{aligned} & 0 \\ & 0 \times 00 \end{aligned}$ |  |
| 4 | Compensation channel | see table 'Compensation channel' «) Further information on page 752 | see table 'Compensation channel' ③) Further information on page 75 | Byte <br> 2 | $\left\lvert\, \begin{aligned} & 0 \\ & 0 \times 00 \end{aligned}\right.$ |  |

Table 167: Channel configuration

| Internal value | Operating modes for the analog inputs, individually configurable |
| :---: | :---: |
| 0 | Unused (default) |
| 2 | Digital input |
| 34 | Analog input -50 mV ... +50 mV |
| 35 | Analog input -500 mV ... +500 mV |
| 36 | Analog input -1 V ... +1 V |
| 7 | Analog input -5 $\mathrm{V} \ldots .+5 \mathrm{~V}$ |
| 5 | Analog input -10 V ... +10 V |
| 6 | Analog input $0 \mathrm{~V} . . .+5 \mathrm{~V}$ |
| 1 | Analog input $0 \mathrm{~V} . . .+10 \mathrm{~V}$ |
| 37 | Analog input -20 mA ... +20 mA |
| 3 | Analog input $0 \mathrm{~mA} . . .20 \mathrm{~mA}$ |
| 4 | Analog input $4 \mathrm{~mA} . . .20 \mathrm{~mA}$ |
| 14 | Analog input Pt100 (2-wire), $-50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ |
| 15 | Analog input Pt100 (3-wire), $-50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ |
| 48 | Analog input Pt100 (4-wire), $-50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ |
| 57 | Analog input Pt100 (2-wire), $-50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ (resolution: $0,01 \mathrm{~K}$ ) |
| 58 | Analog input Pt100 (3-wire), $-50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ (resolution: $0,01 \mathrm{~K}$ ) |
| 59 | Analog input Pt100 (4-wire), $-50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ (resolution: $0,01 \mathrm{~K}$ ) |
| 8 | Analog input Pt100 (2-wire), $-50^{\circ} \mathrm{C} . . .+400^{\circ} \mathrm{C}$ |
| 9 | Analog input Pt100 (3-wire), $-50^{\circ} \mathrm{C}$... $+400^{\circ} \mathrm{C}$ |
| 49 | Analog input Pt100 (4-wire), $50^{\circ} \mathrm{C}$... $+400^{\circ} \mathrm{C}$ |
| 45 | Analog input Pt100 (2-wire), $-200^{\circ} \mathrm{C} . . .+850^{\circ} \mathrm{C}$ |
| 46 | Analog input Pt100 (3-wire), $-200^{\circ} \mathrm{C} \ldots+850^{\circ} \mathrm{C}$ |
| 47 | Analog input Pt100 (4-wire), $-200^{\circ} \mathrm{C} \ldots+850^{\circ} \mathrm{C}$ |
| 16 | Analog input Pt1000 (2-wire), $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ |
| 17 | Analog input Pt1000 (3-wire), $-50^{\circ} \mathrm{C} \ldots+400{ }^{\circ} \mathrm{C}$ |
| 50 | Analog input Pt1000 (4-wire), $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ |
| 18 | Analog input Ni1000 (2-wire), $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ |
| 19 | Analog input Ni1000 (3-wire), $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ |
| 51 | Analog input Ni1000 (4-wire), $-50^{\circ} \mathrm{C} \ldots+150{ }^{\circ} \mathrm{C}$ |
| 39 | Analog input Cu50 1.426 (2-wire) -50 ${ }^{\circ} \mathrm{C}$... $+200^{\circ} \mathrm{C}$ |
| 40 | Analog input Cu50 1.426 (3-wire) $-50^{\circ} \mathrm{C}$... $+200^{\circ} \mathrm{C}$ |
| 41 | Analog input Cu50 1.426 (4-wire) -50 ${ }^{\circ} \mathrm{C}$... $+200^{\circ} \mathrm{C}$ |
| 42 | Analog input Cu50 1.428 (2-wire) - $200^{\circ} \mathrm{C} . . .+200^{\circ} \mathrm{C}$ |
| 43 | Analog input Cu50 1.428 (3-wire) - $200^{\circ} \mathrm{C} \ldots+200^{\circ} \mathrm{C}$ |
| 44 | Analog input Cu50 1.428 (4-wire) -200 ${ }^{\circ} \mathrm{C} . . .+200^{\circ} \mathrm{C}$ |
| 24 | Analog input J-type thermocouple $-210^{\circ} \mathrm{C} \ldots+1200^{\circ} \mathrm{C}$ |
| 25 | Analog input K-type thermocouple $-270^{\circ} \mathrm{C} . . .+1372{ }^{\circ} \mathrm{C}$ |
| 30 | Analog input N -type thermocouple - $270{ }^{\circ} \mathrm{C} \ldots+1300^{\circ} \mathrm{C}$ |


| Internal <br> value | Operating modes for the analog inputs, individually configurable |
| :--- | :--- |
| 27 | Analog input S-type thermocouple $-50^{\circ} \mathrm{C} \ldots+1768^{\circ} \mathrm{C}$ |
| 28 | Analog input T-type thermocouple $-270^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ |
| 38 | Analog input resistor $50 \mathrm{k} \Omega$ |
| 52 | Temperature-internal reference point |
| 53 | Common mode voltage |

Table 168: Channel monitoring

| Internal <br> value | Monitoring |
| :--- | :--- |
| 0 | Plausibility, open-circuit (wire break) and short circuit (default) |
| 3 | No monitoring |

Table 169: Line frequency suppression

| Internal <br> value | Line frequency suppression |
| :--- | :--- |
| 0 | 50 Hz |
| 1 | 60 Hz |
| 2 | No line frequency suppression |

Table 170: Compensation channel

| Internal <br> value | Compensation channel |
| :--- | :--- |
| 0 | Internal compensation (default) |
| 1 | Channel 0 (possible with channels 1, 2, 3) |
| 2 | Channel 1 (possible with channels $0,2,3$ ) |
| 3 | Channel 2 (possible with channels $0,1,3$ ) |
| 4 | Channel 3 (possible with channels $0,1,2$ ) |
| 5 | Channel 4 (possible with channels 5, 6, 7) |
| 6 | Channel 5 (possible with channels 4, 6, 7) |
| 7 | Channel 6 (possible with channels 4, 5, 7) |
| 8 | Channel 7 (possible with channels 4, 5, 6) |
| 9 | External with temperature value |

Diagnosis

| E1 ... E4 | d1 | d2 | d3 | d4 | $\begin{array}{\|l\|} \hline \text { Identifier } \\ 000 . . .063 \end{array}$ | AC500 <br> display | <- Display in |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err | PS501 PLC browser |  |
| Byte 6 <br> Bit 6 ... 7 | - | Byte 3 | Byte 4 | Byte 5 | Byte 6 $\text { Bit } 0 \text {... } 5$ | FBP diagnosis block |  |
| Class | Interface | Device | Module | Channel | Error identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) | $\left.{ }^{4}\right)$ |  |  |  |

Module error

|  | 14 | 1 ... 10 | 31 | 31 | 19 | Checksum error in the | Replace I/O module |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 3 | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |
| 3 | 14 | 1 ... 10 | 31 | 31 | 3 | Timeout in the I/O module | Replace I/O module |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |
| 3 | 14 | $1 . .10$ | 31 | 31 | 40 | Different hard-/firmware versions in the module | Replace I/O module |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |
| 3 | 14 | 1 ... 10 | 31 | 31 | 43 | Internal error in the module, e.g. internal analog voltage is not correct | Replace I/O module |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |
| 3 | 14 | 1 ... 10 | 31 | 31 | 36 | Internal data exchange failure | Replace I/O module |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |
| 3 | 14 | 1 ... 10 | 31 | 31 | 9 | Overflow diagnosis buffer | Restart |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |
| 3 | 14 | 1 ... 10 | 31 | 31 | 26 | Parameter error | Check master |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |
| 3 | 14 | 1 ... 10 | 31 | 31 | 11 | Process voltage too low | Check process voltage |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |
| 4 | 14 | $1 . .10$ | 31 | 31 | 45 | Process voltage is switched OFF (ON -> OFF) | Process voltage ON |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |

Channel error

|  | 14 | 1 ... 10 | 1 | $0 \ldots 7$ | 48 | Analog value overflow or broken wire at an analog input | Check input value or terminal |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 4 | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |
| 4 | 14 | 1 ... 10 | 1 | $0 . . .7$ | 7 | Analog value underflow at an analog input | Check input value |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |
| 4 | 14 | 1 ... 10 | 1 | $0 . . .7$ | 47 | Short circuit at an analog input | Check terminal |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |
| 4 | 14 | 1 ... 10 | 1 | $0 . . .7$ | 1 | Possibly wrong measured value caused by inadmissible temperature of the compensation channel | Check the temperature compensation channel |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |


| E1 ... E4 | d1 | d2 | d3 | d4 | Identifier $000 \ldots 063$ | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err |  |  |
| Byte 6 <br> Bit 6 ... 7 | - | Byte 3 | Byte 4 | Byte 5 | Byte 6 <br> Bit 0 ... 5 |  |  |
| Class | Interface | Device | Module | Channel | Error identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | $\left.{ }^{3}\right)$ | ${ }^{4}$ ) |  |  |  |
| 4 | 14 | $1 \ldots 10$ | 1 | $0 \ldots 7$ | 2 | Invalid measured value of the channel caused by overly high voltage difference | Check voltage difference; install equalizing conductors if necessary |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |
| 4 | 14 | $1 \ldots 10$ | 1 | $0 \ldots 7$ | 11 | Output voltage 10 V faulty | Check output load |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |

Remarks:

| $\left.{ }^{1}\right)$ | In AC500, the following interface identifier applies: 14 = I/O bus, 11 = COM1 (e.g. CS31 bus), $12=$ COM2. <br> The FBP diagnosis block does not contain this identifier. |
| :---: | :---: |
| ${ }^{2}$ ) | With "Device" the following allocation applies: <br> 31 = module itself, $1 \ldots 10$ expansion module $1 \ldots 10$, ADR = hardware address (e.g. of the DC551) |
| ${ }^{3}$ ) | With "Module" the following allocation applies dependent of the master: <br> Module error: I/O bus or FBP: 31 = module itself; COM1/COM2: 1 ... 10 = expansion 1 ... 10 <br> Channel error: I/O bus or FBP = module type ( 1 = AI); COM1/COM2: <br> 1 ... 10 = expansion 1 ... 10 |
| ${ }^{4}$ ) | In case of module errors, with channel "31 = Module itself" is output. |

## State LEDs

During the power ON procedure, the module initializes automatically. All LEDs (except the channel LEDs) are ON during this time.
States of the LEDs (see also section diagnosis LEDs in the S500 system data):


## Measuring ranges

## Voltage input ranges

Bipolar voltage input range, measuring bridge
The represented resolution corresponds to 16 bits.

| Range | -50 <br> mV ... <br> $+50 \mathrm{mV}$ | $\begin{aligned} & -500 \\ & \mathrm{mV} \ldots \\ & +500 \\ & \mathrm{mV} \end{aligned}$ | $\begin{aligned} & -1 \mathrm{~V} \ldots \\ & +1 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & -5 \mathrm{~V} . . \\ & +5 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & -10 \mathrm{~V} . . \\ & +10 \mathrm{~V} \end{aligned}$ | Commo n Mode Voltage | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  |  | Decimal | Hex. |
| Overflow | $58.7945$ | $\begin{aligned} & 5 \\ & 587.944 \\ & 9 \end{aligned}$ | $\begin{aligned} & \hline> \\ & 1.17589 \end{aligned}$ | > 5.8794 | $11.7589$ | $20.0000$ | 32767 | 7FFF |
| Measured value too high | $\begin{aligned} & 58.7945 \\ & : \\ & 50.0018 \end{aligned}$ | $\begin{aligned} & 587.944 \\ & 9 \\ & : \\ & 500.018 \\ & 1 \end{aligned}$ | $\begin{aligned} & 1.17589 \\ & : \\ & 1.00004 \end{aligned}$ | $\begin{aligned} & 5.8794 \\ & : \\ & 5.0002 \end{aligned}$ | $\begin{aligned} & 11.7589 \\ & : \\ & 10.0004 \end{aligned}$ |  | $\begin{aligned} & 32511 \\ & : \\ & 27649 \end{aligned}$ | $\begin{aligned} & \text { 7EFF } \\ & : \\ & 6 \mathrm{C} 01 \end{aligned}$ |
| Normal range | $\begin{aligned} & 50.0000 \\ & : \\ & 0.0018 \end{aligned}$ | $\begin{aligned} & 500.000 \\ & 0 \\ & : \\ & 0.0181 \end{aligned}$ | $\begin{aligned} & 1.00000 \\ & : \\ & 0.00004 \end{aligned}$ | $\begin{aligned} & 5.0000 \\ & : \\ & 0.0002 \end{aligned}$ | $\begin{aligned} & 10.0000 \\ & : \\ & 0.0004 \end{aligned}$ | $\begin{aligned} & 20.0000 \\ & : \\ & 0.0008 \end{aligned}$ | $\begin{aligned} & 27648 \\ & : \\ & 1 \end{aligned}$ | $\begin{aligned} & \text { 6C00 } \\ & : \\ & 0001 \end{aligned}$ |
|  | 0.0000 | 0.0000 | 0.0000 | 0.00000 | 0.0000 | 0.0000 | 0 | 0000 |


| Range | $\begin{array}{\|l} -50 \\ \mathrm{mV} \ldots \\ +50 \mathrm{mV} \end{array}$ | $\begin{aligned} & \hline-500 \\ & \mathrm{mV} \ldots \\ & +500 \\ & \mathrm{mV} \end{aligned}$ | $\begin{aligned} & \hline-1 \mathrm{~V} \ldots \\ & +1 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & -5 \mathrm{~V} . . . \\ & +5 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & -10 \mathrm{~V} . . . \\ & +10 \mathrm{~V} \end{aligned}$ | Commo <br> n Mode Voltage | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  |  | Decimal | Hex. |
| Normal range or Measured value too low | \|-0.0018 $-50.0000$ | $\begin{aligned} & -0.0181 \\ & : \\ & -500.000 \\ & 0 \end{aligned}$ | $\begin{aligned} & -0.00004 \\ & : \\ & -1.00000 \end{aligned}$ | $\begin{aligned} & -0.0002 \\ & : \\ & -5.0000 \end{aligned}$ | -0.004 $-10.0000$ | $\begin{aligned} & -0.0008 \\ & : \\ & -20.0000 \end{aligned}$ | $\begin{array}{\|l} -1 \\ : \\ -27648 \end{array}$ | $\begin{aligned} & \text { FFFF } \\ & : \\ & 9400 \end{aligned}$ |
| Measured value too low | $-50.0018$ $-58.7945$ | $\begin{aligned} & \hline-500.018 \\ & 1 \\ & : \\ & -587.944 \\ & 9 \end{aligned}$ | $\begin{aligned} & -1.00004 \\ & : \\ & -1.17589 \end{aligned}$ | $\begin{aligned} & -5.0002 \\ & : \\ & -5.8794 \end{aligned}$ | $\begin{aligned} & -10.0004 \\ & : \\ & -11.7589 \end{aligned}$ |  | $\begin{aligned} & -27649 \\ & : \\ & -32512 \end{aligned}$ | $\begin{aligned} & \text { 93FF } \\ & : \\ & 8100 \end{aligned}$ |
| Underflow | < | $\begin{aligned} & \hline< \\ & -587.944 \\ & 9 \end{aligned}$ | $\begin{aligned} & \hline< \\ & -1.17589 \end{aligned}$ | $\begin{aligned} & \mid< \\ & -5.8794 \end{aligned}$ | $\begin{aligned} & \hline< \\ & -11.7589 \end{aligned}$ | $\begin{aligned} & < \\ & -20.0000 \end{aligned}$ | -32768 | 8000 |

Unipolar voltage input range, measuring bridge, digital input

| Range | 0V... +5 V | $\begin{aligned} & \mathrm{O} V \ldots+10 \\ & \mathrm{~V} \end{aligned}$ | Digital input | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  | Decimal | Hex. |
| Measured value too high | $\begin{aligned} & 5.8794 \\ & : \\ & 5.0002 \end{aligned}$ | $\begin{aligned} & 11.7589 \\ & : \\ & 10.0004 \end{aligned}$ |  | $\begin{aligned} & 32511 \\ & : \\ & 27649 \end{aligned}$ | $\begin{aligned} & \text { 7EFF } \\ & : \\ & 6 \mathrm{C} 01 \end{aligned}$ |
| Normal range | $\begin{aligned} & 5.0000 \\ & : \\ & 0.0002 \end{aligned}$ | $\begin{aligned} & 10.0000 \\ & : \\ & 0.0004 \end{aligned}$ | ON | $27648$ $1$ | $\begin{aligned} & 6 \mathrm{C} 00 \\ & : \\ & 0001 \end{aligned}$ |
|  | 0.0000 | 0.0000 | OFF | 0 | 0000 |
| Measured value too low | $\begin{aligned} & -0.0002 \\ & : \\ & -0.8794 \end{aligned}$ | $\begin{aligned} & -0.0004 \\ & : \\ & -1.1759 \end{aligned}$ |  | $\begin{aligned} & -1 \\ & : \\ & -4864 \end{aligned}$ | $\begin{aligned} & \text { FFFF } \\ & : \\ & \text { ED00 } \end{aligned}$ |
| Underflow | <-0.8794 | <-1.1759 |  | -32768 | 8000 |

## Current input ranges

| Range | $\mathbf{- 2 0 ~ m A} \ldots$ <br> $\mathbf{+ 2 0} \mathbf{m A}$ | $\mathbf{0} \mathbf{m A} \ldots+20$ <br> mA | $\mathbf{4} \mathbf{m A} \ldots 20$ <br> mA | Digital value |  |
| :--- | :--- | :--- | :--- | :--- | :--- |
|  |  |  | Decimal | Hex. |  |
| Overflow | $>23.5178$ | $>23.5178$ | $>22.8142$ | 32767 | 7FFF |
| Measured <br> value too <br> high | 23.5178 | 23.5178 | 22.8142 | 32511 | 7EFF |
|  | $:$ | $:$ | $:$ | $:$ | $:$ |


| Range | $\begin{aligned} & -20 \mathrm{~mA} \ldots \\ & +20 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 0 \mathrm{~mA} \ldots+20 \\ & \mathrm{~mA} \end{aligned}$ | ${ }_{\mathrm{mA}}^{4 \mathrm{~mA}} . . .20$ | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  | Decimal | Hex. |
| Normal range | 20.0000 $:$ 0.0007 | 20.0000 <br> 0.0007 | 20.0000 $:$ 4.0006 | 27648 |  |
|  | 0.0000 | 0.0000 | 4.0000 | 0 | 0000 |
|  | $\begin{array}{\|l} \hline-0.0007 \\ : \\ -20.0000 \end{array}$ |  |  | -1 <br> -27648 | FFFF $:$ 9400 |
| Measured value too low |  | -0.0007 <br> -3.5178 | $\begin{aligned} & 3.9994 \\ & : \\ & 1.1852 \end{aligned}$ | $\begin{aligned} & \hline-1 \\ & : \\ & -4864 \end{aligned}$ | $\begin{aligned} & \text { FFFF } \\ & \text { : } \\ & \text { ED00 } \end{aligned}$ |
|  | -20.0007 $:$ -23.5178 |  |  | $\begin{aligned} & -27649 \\ & : \\ & -32512 \end{aligned}$ | $\begin{aligned} & \hline 93 F F \\ & : \\ & 8100 \\ & \hline \end{aligned}$ |
| Underflow | <-23.5178 | <-3.5178 | < 1.1852 | -32768 | 8000 |

## Resistance thermometer input ranges

The represented resolution corresponds to 16 bits.

| Range | $\begin{aligned} & \text { Pt100 } \\ & -50^{\circ} \mathrm{C} \ldots \\ & \left.+70^{\circ} \mathrm{C}{ }^{\mathrm{i}}\right) \end{aligned}$ | $\begin{aligned} & \hline \text { Pt100 I } \\ & \text { Pt1000 } \\ & -50^{\circ} \mathrm{C} \ldots \\ & +400^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \hline \mathrm{Pt} 100 \\ & -200 \\ & { }^{\circ} \mathrm{C} \ldots \\ & +850{ }^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \mathrm{Ni} 1000 \\ & -50^{\circ} \mathrm{C} \ldots \\ & +150^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \hline \text { Cu50 } \\ & -200 \\ & { }^{\circ} \mathrm{C} \ldots \\ & +200^{\circ} \mathrm{C} \end{aligned}$ | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  | Decimal | Hex. |
| Overflow | $\begin{aligned} & >+80.0 \\ & { }^{\circ} \mathrm{C} \text { C } \end{aligned}$ | $\begin{aligned} & >+450.0 \\ & { }^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & >+850 \\ & { }^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & >+160.0 \\ & { }^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & >+200 \\ & { }^{\circ} \mathrm{C} \mathrm{C} \end{aligned}$ | 32767 | 7FFF |
| Measured value too high |  | $\begin{aligned} & +450.0 \\ & { }^{\circ} \mathrm{C} \\ & \vdots \\ & +400.1 \\ & { }^{\circ} \mathrm{C} \mathrm{C} \end{aligned}$ |  |  |  | $\begin{aligned} & 4500 \\ & : \\ & 4001 \end{aligned}$ | $\begin{aligned} & 1194 \\ & : \\ & \text { 0FA1 } \end{aligned}$ |
|  |  |  |  | $\begin{aligned} & +160.0 \\ & { }^{\circ} \mathrm{C} \\ & \vdots \\ & +150.1 \\ & { }^{\circ} \mathrm{C} \mathrm{C} \end{aligned}$ |  | $\begin{aligned} & 1600 \\ & : \\ & 1501 \end{aligned}$ | $\begin{aligned} & 0640 \\ & : \\ & 05 D D \end{aligned}$ |
|  | $+80.0^{\circ} \mathrm{C}$ <br> $+70.1^{\circ} \mathrm{C}$ |  |  |  |  | $\begin{aligned} & \hline 800 \\ & : \\ & 701 \\ & \hline \end{aligned}$ | $\begin{aligned} & 0320 \\ & : \\ & 02 B D \end{aligned}$ |


| Range | Pt 100$-50^{\circ} \mathrm{C} \ldots$$\left.+70^{\circ} \mathrm{C}{ }^{i}\right)$ | $\begin{aligned} & \hline \mathrm{Pt} 100 \mathrm{I} \\ & \mathrm{Pt} 1000 \\ & -50^{\circ} \mathrm{C} \ldots \\ & +400^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \hline \mathrm{Pt100} \\ & -200 \\ & { }^{\circ} \mathrm{C} \ldots \\ & +850^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \hline \mathrm{Ni} 1000 \\ & -50^{\circ} \mathrm{C} \ldots \\ & +150^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \hline \mathrm{Cu} 50 \\ & -200 \\ & { }^{\circ} \mathrm{C} \ldots \\ & +200^{\circ} \mathrm{C} \end{aligned}$ | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  | Decimal | Hex. |
| Normal range | $\begin{aligned} & : \\ & : \\ & : \\ & +70.0^{\circ} \mathrm{C} \\ & : \\ & +0.1^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & +400.0 \\ & { }^{\circ} \mathrm{C} \\ & : \\ & : \\ & : \\ & : \\ & +0.1^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \hline+850.0 \\ & { }^{\circ} \mathrm{C} \\ & \vdots \\ & \vdots \\ & \vdots \\ & \vdots \\ & +0.1^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & +150.0 \\ & { }^{\circ} \mathrm{C} \\ & : \\ & \vdots \\ & +0.1^{\circ} \mathrm{C} \end{aligned}$ | +200.0 <br> ${ }^{\circ} \mathrm{C}$ $\begin{aligned} & : \\ & : \\ & : \\ & +0.1^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \hline 8500 \\ & 4000 \\ & 2000 \\ & 1500 \\ & 700 \\ & : \\ & 1 \end{aligned}$ | $\begin{aligned} & 2134 \\ & \text { 0FAO } \\ & \text { 07DO } \\ & \text { 05DC } \\ & \text { 02BC } \\ & : \\ & 1 \end{aligned}$ |
|  | $0.0{ }^{\circ} \mathrm{C}$ | $0.0{ }^{\circ} \mathrm{C}$ | $0.0{ }^{\circ} \mathrm{C}$ | $0.0{ }^{\circ} \mathrm{C}$ |  | 0 | 0000 |
|  | $\begin{aligned} & -0.1^{\circ} \mathrm{C} \\ & : \\ & -50.0^{\circ} \mathrm{C} \end{aligned}$ | $-0.1^{\circ} \mathrm{C}$ <br> $-50.0^{\circ} \mathrm{C}$ | $\begin{aligned} & -0.1^{\circ} \mathrm{C} \\ & : \\ & : \\ & -200^{\circ} \mathrm{C} \end{aligned}$ | $-0.1^{\circ} \mathrm{C}$ <br> $-50.0^{\circ} \mathrm{C}$ | $\begin{array}{\|l\|} \hline-0.1^{\circ} \mathrm{C} \\ : \\ -50.0^{\circ} \mathrm{C} \\ \left.{ }^{2}\right) \\ -200.0^{\circ} \mathrm{C} \\ \left.{ }^{2}\right) \end{array}$ | $\begin{aligned} & \hline-1 \\ & : \\ & -500 \\ & -2000 \end{aligned}$ | $\begin{aligned} & \text { FFFF } \\ & \text { : } \\ & \text { FE0C } \\ & \text { F830 } \end{aligned}$ |
| Measured value too low | $\begin{aligned} & -50.1^{\circ} \mathrm{C} \\ & : \\ & -60.0^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & -50.1^{\circ} \mathrm{C} \\ & = \\ & -60.0^{\circ} \mathrm{C} \end{aligned}$ |  | $\begin{aligned} & -50.1^{\circ} \mathrm{C} \\ & : \\ & -60.0^{\circ} \mathrm{C} \end{aligned}$ |  | $\begin{aligned} & -501 \\ & : \\ & -600 \end{aligned}$ | $\begin{aligned} & \text { FE0B } \\ & : \\ & \text { FDA8 } \end{aligned}$ |
| Underflow | $\begin{aligned} & <-60.0 \\ & { }^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \text { <-60.0 } \\ & { }^{\circ} \mathrm{C} \end{aligned}$ | $<-200^{\circ} \mathrm{C}$ | $\begin{aligned} & \text { <-60.0 } \\ & { }^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \mid<-200^{\circ} \mathrm{C} \\ & \left.{ }^{2}\right) \end{aligned}$ | -32768 | 8000 |

${ }^{1}$ ) also possible with resolution 0.01 K
${ }^{2}$ ) if Cu 50 with $1.426,-50^{\circ} \mathrm{C}$ is valid; if Cu 50 with $1.428,-200.0^{\circ} \mathrm{C}$ is valid

## Resistor input range

The represented resolution corresponds to 16 bits.

| Range | Resistor $[\Omega]$ | Digital value |  |
| :--- | :--- | :--- | :--- |
|  |  | Decimal | Hex. |
| Overflow | $>55000$ | 32767 | 7 FFF |
| Measured value <br> too high | 55000 | 30413 | 76 CD |
|  | $:$ | $:$ | $:$ |
|  | 50001 | 27649 | 6 C 01 |
| Normal range | 50000 | 27648 | 6 C 00 |
|  | $:$ | $:$ | $:$ |
|  | 2 | 1 | 0001 |
|  | 1 | 1 | 0001 |
|  | 0 | 0 | 0000 |

## Thermocouple input ranges

The represented resolution corresponds to 16 bits.

| Range | $\begin{aligned} & \text { Typ J } \\ & -210 \\ & { }^{\circ} \mathrm{C} \ldots \\ & +1200^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \hline \text { Typ K } \\ & -270 \\ & { }^{\circ} \mathrm{C} \ldots \\ & +1372^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \hline \text { Typ N } \\ & -270 \\ & { }^{\circ} \mathrm{C} \ldots \\ & +1300^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \text { Typ S } \\ & -50^{\circ} \mathrm{C} \ldots \\ & +1768{ }^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \hline \text { Typ T } \\ & -270 \\ & { }^{\circ} \mathrm{C} \ldots \\ & +400^{\circ} \mathrm{C} \end{aligned}$ | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  | Decimal | Hex. |
| Overflow | $\begin{aligned} & \hline> \\ & +1200.0 \\ & { }^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \hline> \\ & +1372.0 \\ & { }^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \hline> \\ & +1300.0 \\ & { }^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \hline> \\ & +1768.0 \\ & { }^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & >+400.0 \\ & { }^{\circ} \mathrm{C} \end{aligned}$ | 32767 | 7FFF |
| Normal range |  |  |  | $\begin{aligned} & +1768.0 \\ & { }^{\circ} \mathrm{C} \end{aligned}$ |  | 17680 | 4510 |
|  |  | $\begin{aligned} & +1372.0 \\ & { }^{\circ} \mathrm{C} \end{aligned}$ |  | : |  | 13720 | 3598 |
|  |  | : | $\begin{aligned} & +1300.0 \\ & { }^{\circ} \mathrm{C} \end{aligned}$ | : |  | 13000 | 32C8 |
|  | $\begin{aligned} & +1200.0 \\ & { }^{\circ} \mathrm{C} \end{aligned}$ | : | : | : |  | 12000 | 2EE0 |
|  | : | : | : | : | $\begin{aligned} & +400.0 \\ & { }^{\circ} \mathrm{C} \end{aligned}$ | 4000 | OFAO |
|  | : | : | : | : | : | : | : |
|  | $+0.1{ }^{\circ} \mathrm{C}$ | $+0.1{ }^{\circ} \mathrm{C}$ | $+0.1{ }^{\circ} \mathrm{C}$ | $+0.1^{\circ} \mathrm{C}$ | $+0.1{ }^{\circ} \mathrm{C}$ | 1 | 1 |
|  | $0.0{ }^{\circ} \mathrm{C}$ | $0.0{ }^{\circ} \mathrm{C}$ | $0.0{ }^{\circ} \mathrm{C}$ | $0.0{ }^{\circ} \mathrm{C}$ |  | 0 | 0000 |
|  | $-0.1^{\circ} \mathrm{C}$ | $-0.1{ }^{\circ} \mathrm{C}$ | $-0.1{ }^{\circ} \mathrm{C}$ | $-0.1^{\circ} \mathrm{C}$ | $-0.1{ }^{\circ} \mathrm{C}$ | -1 | FFFF |
|  | : | : | : | : | : | : | : |
|  | : | : | : | $-50.0{ }^{\circ} \mathrm{C}$ | : | -500 | FEOC |
|  | $-210.0{ }^{\circ} \mathrm{C}$ | : | : | : | : | -2100 | F7CC |
|  |  | $-270.0{ }^{\circ} \mathrm{C}$ | $-270.0{ }^{\circ} \mathrm{C}$ |  | $-270.0{ }^{\circ} \mathrm{C}$ | -2700 | F574 |
| Underflow | $\begin{aligned} & <-210.0 \\ & { }^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & <-270.0 \\ & { }^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & <-270.0 \\ & { }^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & <-50.0 \\ & { }^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & <-270.0 \\ & { }^{\circ} \mathrm{C} \end{aligned}$ | -32768 | 8000 |

## Temperature-internal reference point ranges

| Range | Value | Digital value |  |
| :--- | :--- | :--- | :--- |
|  |  | Decimal | Hex. |
| Overflow | $>+85^{\circ} \mathrm{C}$ | 32767 | $7 F F F$ |
|  | $+85^{\circ} \mathrm{C}$ | 850 | 0352 |
|  | $0^{\circ} \mathrm{C}$ | 0 | 0000 |
|  | $-40^{\circ} \mathrm{C}$ | -400 | FE70 |
| Underflow | $<-40^{\circ} \mathrm{C}$ | -32768 | 8000 |

## Technical data

## Technical data of the module

The system data of AC500 and S500 are applicable to the standard version ${ }^{*} \Rightarrow$ Chapter 5.1.2 "System data AC500" on page 166.

The system data of AC500-XC are applicable to the XC version $\stackrel{\text { Chapter 5.1.3 "System data }}{ }$ AC500-XC" on page 169.

Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter | Value |
| :---: | :---: |
| Process voltage |  |
| Connections | Terminals $1.8,2.8,3.8$ and 4.8 for +24 V (UP) as well as $1.9,2.9,3.9$ and 4.9 for $0 \mathrm{~V}(\mathrm{ZP})$ |
| Rated value | 24 V DC |
| Max. ripple | 5 \% |
| Protection against reversed voltage | Yes |
| Rated protection fuse on UP | 10 A fast |
| Galvanic isolation | Yes, per module |
| Current consumption |  |
| From 24 V DC power supply at the terminals UP/L+ and ZP/M of the CPU/communication interface module | Ca. 2 mA |
| Current consumption from UP in normal operation | 130 mA |
| Inrush current from UP (at power up) | $0.056 \mathrm{~A}^{2} \mathrm{~s}$ |
| Max. length of analog cables, conductor cross section $>0.14 \mathrm{~mm}^{2}$ ) | 100 m |
| Weight | 130 g |
| Mounting position | Horizontal or vertical with derating (max. temperature $40^{\circ} \mathrm{C}$ ) |
| Cooling | The natural convection cooling must not be hindered by cable ducts or other parts in the control cabinet. |
| *) Please note that an additional current of approx. $3 \mu \mathrm{~A}$ flows out of the input for the wire break detection. Depending on the internal resistance of the signal source and the wire, this can lead to a higher measured value due to the voltage drop. |  |

## NOTICE!

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and temporary overvoltage up to 30 V DC.

## Technical data of the analog inputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 8 |
| Distribution of channels into groups | 2 groups of 4 channels each |
| Connections of the channels I0 ... I3 | Terminals 1.0 ... 1.7 and terminals 2.0 ... 2.7 |
| Connections of the channels 14 ... 17 | Terminals 3.0 ... 3.7 and terminals 4.0 ... 4.7 |
| Input type | Bipolar (not with current or Pt100/ Pt1000/ Ni1000/ Cu50/ resistor) |
| Galvanic isolation | Against internal supply and other modules |
| Common mode input range | $\pm 20 \mathrm{~V}$ DC plus signal voltage |
| Configurability | Digital input, $-50 \mathrm{mV} \ldots+50 \mathrm{mV}$, -500 mV ... $+500 \mathrm{mV},-1 \mathrm{~V} \ldots+1 \mathrm{~V}$, $-5 \vee \ldots+5 \vee,-10 \vee \ldots+10 \vee$, $0 \mathrm{~V} . .+5 \mathrm{~V}, 0 \mathrm{~V} \ldots+10 \mathrm{~V}$, -20 mA ... $+20 \mathrm{~mA}, 0 \mathrm{~mA} . .20 \mathrm{~mA}$, 4 mA ... $20 \mathrm{~mA}, \mathrm{Pt} 100$, Pt1000, Ni1000, Cu50, resistor, thermocouple types J, K, $\mathrm{N}, \mathrm{S}, \mathrm{T}$ (each input can be configured individually) |
| Channel input resistance | Voltage: > $100 \mathrm{k} \Omega$, current: ca. $330 \Omega$ |
| Time constant of the input filter | Line-frequency suppression $50 \mathrm{~Hz}, 60$ Hz , none |
| Indication of the input signals | 1 yellow LED per channel, the brightness depends on the value of the analog signal |
| Conversion time | 1 ms (none), <br> $100 \mathrm{~ms}(50 \mathrm{~Hz} / 60 \mathrm{~Hz})$ per channel |
| Resolution | 16 bits including sign |
| Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range | $\begin{array}{\|l\|l} \hline \text { Typ. } & \pm 0.1 \% \text { (voltage) } \\ & \pm 0.3 \% \text { (current, resistor) } \\ & \text { at } 25^{\circ} \mathrm{C} \\ \hline \end{array}$ |
|  | $\begin{array}{\|l\|l} \hline \text { Max } & \pm 0.7 \% \text { (voltage) } \\ & \pm 0.9 \% \text { (current, resistor) } \\ & \pm 0.5 \% \text { (thermocouple type J, N, } \\ & \text { S, T; thermocouple type K > }-220 \\ & { }^{\circ} \mathrm{C} \text { ) } \\ & 1.0 \mathrm{~K} \text { (resistance temperature } \\ \text { detectors) } \\ & \begin{array}{l} \text { at } 0^{\circ} \mathrm{C} \\ \text { ance } \end{array} . .0^{\circ} \mathrm{C} \text { or EMC disturb- } \end{array}$ |
| Maximum permanent allowed overload (no damage) |  |


| Parameter |  |
| :--- | :--- |
| Current input | Value |
|  | When the input current exceeds the <br> overflow value of the measurement <br> range, the input impedance is switthed <br> to high impedance for protection. The <br> maximum allowed overload is then 30 <br> v. The digital value corresponds to the <br> overflow value. Periodically, the input <br> impedance is switched to the normal <br> value and the input current is measured. <br> If the input current is within the meas- <br> urement range, the input impedance <br> remains at the normal level and the dig- <br> ital value corresponds to the measured <br> current. |
| Voltage input | 30 V |
| Relationship between input signal and hex code | § Table 168 "Channel monitoring" <br> on page 752 |
| Unused voltage inputs | Are configured as "unused" |
| Unused current inputs | Have a low resistance, can be left open- <br> circuited |
| Overvoltage protection | Yes |

## Technical data of the analog inputs if used as digital inputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | Max. 8 |
| Distribution of channels into groups | 2 groups of 4 channels each |
| Connections of the channels I0+ to I3+ <br> Connections of the channels I4+ to $17+$ | Terminals 2.0, 2.2, 2.4, 2.6 <br> Terminals 4.0, 4.2, 4.4, 4.6 |
| Reference potential for the inputs | Terminals 1.9, 2.9, 3.9 and 4.9 (ZP) |
| Input delay | Typ. 2 ms |
| Indication of the input signals | 1 LED per channel |
| Input signal voltage | 24 V DC |
|  | Signal 0 |
|  | Undefined signal |
|  | Signal 1 |
| Input current per channel | $+50 \mathrm{~V} \ldots+5 \mathrm{~V}$ |
|  | Input voltage +24 V |
|  | Input voltage +5 V |
|  | Input voltage +15 V |
|  | Input voltage +30 V |
| Input resistance | Typ. 50 mA |

## Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm

The dimensions are in mm and in brackets in inch.

Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 250 600 R0001 | Al531, analog input module, 8 AI, <br> U/I/Pt100, TC, 16 bits including sign, <br> 4-wires | Active |
| 1SAP 450 600 R0001 | AI531-XC, analog input module, 8 AI, <br> U/I/Pt100, TC, 16 bits including sign, <br> 4-wires, XC version | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

## AO523-Analog output module

## Features

- Module-wise galvanically isolated
- XC version for use in extreme ambient conditions available


1 I/O bus
2 Allocation between terminal number and signal name
316 yellow LEDs to display the signal states at the analog outputs (O0 ... O15)
41 green LED to display the state of the process supply voltage UP
52 red LEDs to display errors
6 Label
7 Terminal unit
8 DIN rail
${ }_{*}^{*}+{ }_{*}^{*}$ Sign for XC version

Intended purpose
The device can be used as a decentralized I/O extension module for S500 communication interface modules (e. g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs.

- 16 analog outputs in two groups:
- 8 channels configurable for voltage or currrent output (O0...O3 / O8...O11)
- 8 channels for voltage output (O4...O7 / O12...O15)

Resolution 12 bits including sign

| Parameter | Value |
| :---: | :---: |
| Resolution of the analog channels |  |
| Voltage -10 V ... +10 V | 12 bits including sign |
| Current 0 mA ... $20 \mathrm{~mA}, 4 \mathrm{~mA} . . .20 \mathrm{~mA}$ | 12 bits |
| LED displays | 19 LEDs for signals and error messages |
| Internal power supply | Through the I/O bus interface (I/O bus) |
| External power supply | Via the terminals ZP and UP (process voltage 24 V DC) |
| Required terminal unit | TU515 or TU516 $\Leftrightarrow$ Chapter 5.2.5.2 "TU515, TU516, TU541 and TU542 for I/O modules" on page 938 |

## Connections

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

The modules are plugged on an I/O terminal unit $\&$ Chapter 5.2.5.2 "TU515, TU516, TU541 and TU542 for I/O modules" on page 938. Properly position the modules and press until they lock in place. The terminal units are mounted on a DIN rail or with 2 screws plus the additional accessory for wall mounting ${ }^{\star} \Rightarrow$ Chapter 5.2.8.2.5 "TA526-Wall mounting accessory" on page 1324.
The connection of the I/O channels is carried out using the 40 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

The terminals $1.8 \ldots 4.8$ and $1.9 \ldots 4.9$ are electrically interconnected within the I/O terminal units and have always the same assignment, independent of the inserted module:
Terminals $1.8 \ldots 4.8$ : process voltage UP $=+24 \mathrm{~V}$ DC
Terminals 1.9 ... 4.9: process voltage $\mathrm{ZP}=0 \mathrm{~V}$ DC
The assignment of the other terminals:

| Terminals | Signal | Description |
| :--- | :--- | :--- |
| $1.0 \ldots 1.7$ | $\mathrm{O} 0-\ldots \mathrm{O} 7-$ | Negative poles of the first 8 <br> analog outputs |
| $2.0 \ldots 2.7$ | $\mathrm{O} 0+\ldots \mathrm{O}++$ | Positive poles of the first 8 <br> analog outputs |


| Terminals | Signal | Description |
| :--- | :--- | :--- |
| $3.0 \ldots 3.7$ | O8- .. O15- | Negative poles of the fol- <br> lowing 8 analog outputs |
| $4.0 \ldots 4.7$ | $\mathrm{O8}+\ldots \mathrm{O} 15+$ | Positive poles of the following <br> 8 analog outputs |

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a communication interface module or a CPU). Thus, the current consumption from 24 V DC power supply at the terminals L+/UP and M/ZP of the CPU/communication interface module increases by 2 mA per AO523.
The external power supply connection is carried out via the UP (+24 V DC) and the ZP (0 V DC) terminals.

## WARNING!

## Removal/Insertion under power

Removal or insertion under power is permissible only if all conditions for hot swapping are fullfilled.

๕ Chapter 4.9.3 "Replace an I/O module with hot swap" on page 153
The devices are not designed for removal or insertion under power when the conditions for hot swap do not apply. Because of unforeseeable consequences, it is not allowed to plug in or unplug devices with the power being ON.
Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while they are energized in a hazardous location could result in an electric arc, which could create an ignition source resulting in fire or explosion.

Prior to proceeding, make sure that power is been disconnected and that the area has been thoroughly checked to ensure that flammable materials are not present.

The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

## Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

Generally, analog signals must be laid in shielded cables. The cable shields must be grounded at both sides of the cables. In order to avoid unacceptable potential differences between different parts of the installation, low resistance equipotential bonding conductors must be laid.


Fig. 116: Connection of the module: 16 analog outputs in two groups ${ }^{\wedge}$ Chapter 5.2.4.4.2.4.3 "Functionality" on page 765

## CAUTION!

By installing equipotential bonding conductors between the different parts of the system, it must be ensured that the potential difference between ZP and AGND never can exceed 1 V .

## CAUTION!

The process supply voltage must be included in the grounding concept (e. g. grounding of the negative terminal).

The modules provide several diagnosis functions ${ }^{\mu}$ Chapter 5.2.4.4.2.4.8 "Diagnosis" on page 774 .

## Connection of analog output loads (Voltage, current)



Fig. 117: Connection example
The following measuring ranges can be configured $\Longleftrightarrow$ Chapter 5.2.4.4.2.4.7 "Parameterization" on page 769:

| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | Load max. $\pm 10 \mathrm{~mA}$ | 1 channel used |
| :--- | :--- | :--- | :--- |
| Current | $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | Load $0 \Omega \ldots 500 \Omega$ | 1 channel used |
| Current | $4 \ldots 20 \mathrm{~mA}$ | Load $0 \Omega \ldots 500 \Omega$ | 1 channel used |

Only the channels $0 \ldots 3$ and $8 \ldots 11$ can be configured as current output ( $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ or $4 \mathrm{~mA} . . .20 \mathrm{~mA}$ ).
The function of the LEDs is described under Displays.
Unused analog outputs can be left open-circuited.

## Internal data exchange

| Digital inputs (bytes) | 0 |
| :--- | :--- |
| Digital outputs (bytes) | 0 |
| Analog inputs (words) | 0 |
| Analog outputs (words) | 16 |

## I/O configuration

The module does not store configuration data itself. It gets its parameterization data from the master device of the I/O bus (CPU or communication interface module) during power-up of the system.
That means replacing I/O modules is possible without any re-parameterization via software.

If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.

## Parameterization

| Firmware version | Configuration |
| :--- | :--- |
| Firmware version > V2.0.0 | The arrangement of the parameter data is per- <br> formed by Control Builder Plus/ Automation <br> Builder software. |

The parameter data directly influences the functionality of modules.
For non-standard applications, it is necessary to adapt the parameters to your system configuration.

Module: Module slot address: $\mathrm{Y}=1$... 10

| No. | Name | Value | Internal value | Internal value, type | Default | Min. | Max. | EDS <br> Slot/ <br> Index |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 1 | Module ID | Internal | $\begin{aligned} & 1510 \\ & 15 \end{aligned}$ | Word | $\begin{aligned} & \hline 1510 \\ & 0 \times 05 \mathrm{e} 6 \end{aligned}$ | 0 | 65535 | 0x0Y01 |
| 2 | Ignore module ${ }^{2}$ ) | $\begin{aligned} & \text { No } \\ & \text { Yes } \end{aligned}$ | $0$ | Byte | $\begin{array}{\|l\|} \hline \text { No } \\ 0 \times 00 \end{array}$ |  |  | Not for FBP |
| 3 | Parameter length in bytes | Internal | 39 | Byte | $\begin{aligned} & 39-\mathrm{CPU} \\ & 39-\mathrm{FBP} \end{aligned}$ | 0 | 255 | 0x0Y02 |
| 4 | Check supply | $\begin{aligned} & \text { Off } \\ & \text { On } \end{aligned}$ | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | Byte | $\begin{array}{\|l\|} \hline \text { On } \\ \text { 0x01 } \end{array}$ | 0 | 1 | 0x0Y03 |
| 5 | Analog data format | Default | 0 | Byte | $\begin{array}{\|l} \hline \text { Default } \\ 0 \times 00 \end{array}$ |  |  | 0x0Y04 |
| 6 | Behaviour of outputs at com-munication errors | Off <br> Last value <br> Substitute value | $\begin{aligned} & \hline 0 \\ & 1+\left(n^{*} 5\right) \\ & 2+\left(n^{*} 5\right), \\ & n \leq 2 \end{aligned}$ | Byte | $\begin{aligned} & \text { Off } \\ & 0 \times 00 \end{aligned}$ | 0 | 2 | 0x0Y05 |
| 7 | Channel con- <br> figura tion <br> Output channel 0 | See table configura $\Leftrightarrow$ Table nel config on page | 'Channel tion' <br> 71 "Chan uration ${ }^{3}$ )" 73 | Byte | Default <br> 0x00 | 0 | 130 | 0x0Y06 |
| 8 | Channel monitoring Output channel 0 | See table monitorin * Table nel monit on page | 'Channel <br> 72 "Chan oring ${ }^{4}$ " 73 | Byte | Default <br> 0x00 | 0 | 3 | 0x0Y07 |
| 9 | Substitute value Output channel 0 | Output channel 0 ! | 0... 0xfff | Word | Default <br> 0x0000 | 0 | 65535 | 0x0Y08 |


| No. | Name | Value | Internal value | Internal value, type | Default | Min. | Max. | EDS Slot/ Index |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 10 ... 15 | Channel configu- <br> ration <br> and <br> channel <br> moni- <br> toring of <br> the <br> output <br> channels <br> 1 to 3 | See <br> table 'Ch figuration © Table nel config on page <br> and <br> table 'Ch monitorin Table nel monit on page | nnel con- <br> 71 "Chan uration ${ }^{3}$ )" 73 <br> annel 172 oring ${ }^{4}$ " 73 | $\begin{array}{\|l\|} \hline \text { Byte } \\ \text { Byte } \end{array}$ | $\begin{array}{\|l\|} \hline \text { Default } \\ 0 \times 00 \\ 0 \times 00 \end{array}$ | $\left\lvert\, \begin{aligned} & 0 \\ & 0 \end{aligned}\right.$ | $\begin{aligned} & 130 \\ & 3 \end{aligned}$ | 0x0Y09 to <br> OxOYOE |
| $16 . . .23$ | Channel configu- <br> ration and channel monitoring of the output channels 4 to 7 | See <br> table 'Ch figuration © Table nel config on page and table 'Ch monitorin *) Table nel monit on page | annel con- <br> 71 "Chan uration ${ }^{3}$ )" 773 <br> nnel <br> ${ }^{\prime}$ <br> 72 "Chan oring ${ }^{4}$ )" <br> 73 | Byte <br> Byte | Default $0 \times 00$ $0 \times 00$ | $0$ | $\begin{aligned} & 128 \\ & 3 \end{aligned}$ |  |
| 24 | Channel configura tion Output channel 8 | See table configura * Table nel config on page | 'Channel tion' <br> 71 "Chan <br> uration ${ }^{3}$ )" <br> 73 | Byte | Default <br> 0x00 | 0 | 130 | 0x0Y17 |
| 25 | Channel monitoring Output channel 8 | See table monitorin * Table nel monit on page | 'Channel <br> ' <br> 72 "Chan <br> oring ${ }^{4}$ )" <br> 73 | Byte | $\begin{array}{\|l\|l\|} \hline \text { Default } \\ 0 \times 00 \end{array}$ | 0 | 3 | 0x0Y18 |
| 26 | Substitute value Output channel 8 | Output channel 8! | 0... 0xffff | Word | Default 0x0000 | 0 | 65535 | 0x0Y19 |


| No. | Name | Value | Internal value | Internal value, type | Default | Min. | Max. | EDS Slot/ Index |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $\begin{aligned} & 27 \\ & \ldots \\ & 32 \end{aligned}$ | Channel configuration and channel monitoring of the output channels 9 to 11 | See table 'C figuratio H Table nel config on page and table 'C toring' ↔ Table nel mon on page | nnel con- <br> 71 "Chan uration ${ }^{3}$ )" 73 <br> nnel moi- <br> 72 "Chan ring ${ }^{4}$ )" 73 | Byte <br> Byte | $\begin{aligned} & \hline \text { Default } \\ & 0 \times 00 \\ & 0 \times 00 \end{aligned}$ | $\begin{aligned} & 0 \\ & 0 \end{aligned}$ | $\begin{aligned} & 130 \\ & 3 \end{aligned}$ | $0 \mathrm{x0Y1A}$ <br> to 0x0Y1F |
| $\begin{array}{\|l\|} \hline 33 \\ \ldots \\ 40 \end{array}$ | Channel configuration and channel monitoring of the output channels 12 to 15 | See table 'C figuratio ↔ Table nel config on page and table 'C monitor』 Table nel mon on page | nnel con- <br> 71 "Chan uration ${ }^{3}$ )" 73 <br> nnel 72 "Chan ring ${ }^{4}$ )" 73 | Byte <br> Byte | $\begin{aligned} & \hline \text { Default } \\ & 0 \times 00 \\ & 0 \times 00 \end{aligned}$ | $\begin{aligned} & 0 \\ & 0 \end{aligned}$ | $\begin{aligned} & 128 \\ & 3 \end{aligned}$ | $0 \times 0 Y 20$ <br> to 0x0Y27 |
| ${ }^{1}$ ) With CS31 and addresses less than 70 and FBP, the value is increased by 1 <br> ${ }^{2}$ ) Not with FBP |  |  |  |  |  |  |  |  |

GSD file:

| Ext_User_Prm_Data_Len = <br> Ext_User_Prm_Data_Const(0) = | $\begin{aligned} & 42 \\ & 0 \times 05,0 \times e 7,0 \times 27,1 \\ & 0 \times 01,0 \times 00,0 \times 00,1 \\ & 0 \times 00,0 \times 00,0 \times 00,0 \times 00,1 \\ & 0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00, ~ I \\ & 0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00 \\ & 0 \times 00, ~ \ \\ & 0 \times 00,0 \times 00,0 \times 00,0 \times 00, ~ I \\ & 0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00, ~ I \\ & 0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00 \\ & 0 x 00 ; \end{aligned}$ |
| :---: | :---: |


| Output channels 0 and 8 (2 channels, AO523) | No. | Name | Value | Internal value | Internal value, type | Default |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | 1 | Channel configuration | see below table 'Channel configuration' Table 171 " Channel configuration ${ }^{3}$ )" on page 773 | see below table 'Channel configuration' Table 171 " Channel configuration ${ }^{3}$ )" on page 773 | Byte | see below table 'Channel configuration' Table 171" Channel configuration ${ }^{3}$ " on page 773 |
|  | 2 | Channel monitoring | see below table 'Channel monitoring' « Table 172" Channel monitoring ${ }^{4}$ )" on page 773 | see below table 'Channel monitoring' « Table 172" Channel monitoring ${ }^{4}$ )" on page 773 *8) | Byte | see below table 'Channel monitoring' « Table 172 " Channel monitoring ${ }^{4}$ )" on page 773 |
|  | 3 | Substitute value ③ Table 173 " Substitute value" on page 774 | 0 ... 65535 | 0 ... <br> 0xffff | Word | 0 |


| Output channels <br> 1 <br> (14 and $9 \ldots 15$ <br> (14 channels, <br> AO523) | 1 | No. | Name |
| :--- | :--- | :--- | :--- |

Table 171: Channel configuration ${ }^{3}$ )

| Internal value | Operating modes of the analog outputs, <br> individually configurable |
| :--- | :--- |
| 0 | Unused (default) |
| 128 | Analog output $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ |
| 129 | Analog output $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ (not with the <br> channels $4 \ldots 7$ and $12 \ldots 15$ ) |
| 130 | Analog output $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ (not with the <br> channels $4 \ldots 7$ and $12 \ldots 15)$ |

Table 172: Channel monitoring ${ }^{4}$ )

| Internal value | Monitoring |
| :--- | :--- |
| 0 | Plausibility, open-circuit (broken wire) and <br> short circuit (default) |
| 1 | Open-circuit (broken wire) and short circuit |
| 2 | Plausibility |
| 3 | No monitoring |

Table 173: Substitute value

| Intended behavior of <br> channel 0 when the control <br> system stops | Required setting of the <br> module parameter "Behav- <br> iour of outputs in case of a <br> communication error" | Required setting of the <br> channel parameter "Substi- <br> tute value" |
| :--- | :--- | :--- |
| Output OFF | OFF | 0 |
| Last value | Last value | 0 |
| Substitute value | OFF or Last value | $1 \ldots 65535$ |

## Diagnosis

Table 174: Possible diagnosis of I/O channels

| Output range | Condition |  |
| :--- | :--- | :--- |
|  | Output value in the PLC <br> underflow | Output value in the PLC overflow |
| $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | Error identifier $=7$ | Error identifier $=4$ |
| $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ |  |  |
| $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ |  |  |


| E1 ... E4 | d1 | d2 | d3 | d4 | $\begin{array}{\|l\|} \hline \text { Identifier } \\ 000 \ldots 063 \end{array}$ | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err |  |  |
| Byte 6 <br> Bit 6 ... 7 | - | Byte 3 | Byte 4 | Byte 5 | Byte 6 <br> Bit 0 ... 5 |  |  |
| Class | Interface | Device | Module | Channel | Error Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | $\left.{ }^{3}\right)$ | ${ }^{4}$ ) |  |  |  |
| Module error |  |  |  |  |  |  |  |
| 3 | 14 | $1 . .10$ | 31 | 31 | 19 | Checksum error in the I/O module | Replace I/O module |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |
| 3 | 14 | $1 \ldots 10$ | 31 | 31 | 3 | Timeout in the I/O module | Replace I/O module |
|  | 11 / 12 | ADR | 1... 10 |  |  |  |  |
| 3 | 14 | $1 \ldots 10$ | 31 | 31 | 40 | Different hard-/firmware versions in the module | Replace I/O module |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |
| 3 | 14 | $1 \ldots 10$ | 31 | 31 | 43 | Internal error in the module | Replace I/O module |
|  | 11/12 | ADR | 1 ... 10 |  |  |  |  |
| 3 | 14 | $1 . .10$ | 31 | 31 | 36 | Internal data exchange failure | Replace I/O module |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |
| 3 | 14 | $1 \ldots 10$ | 31 | 31 | 9 | Overflow diagnosis buffer | New start |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |
| 3 | 14 | $1 \ldots 10$ | 31 | 31 | 26 | Parameter error | Check master |


| E1 ... E4 | d1 | d2 | d3 | d4 | $\begin{aligned} & \hline \text { Identifier } \\ & 000 \ldots 063 \end{aligned}$ | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err |  |  |
| Byte 6 <br> Bit 6 ... 7 | - | Byte 3 | Byte 4 | Byte 5 | Byte 6 <br> Bit 0 ... 5 |  |  |
| Class | Interface | Device | Module | Channel | Error Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) | ${ }^{4}$ ) |  |  |  |
|  | $11 / 12$ | ADR | 1... 10 |  |  |  |  |
| 3 | 14 | $1 \ldots 10$ | 31 | 31 | 11 | Process voltage too low | Check process voltage |
|  | 11/12 | ADR | $1 . .10$ |  |  |  |  |
| 4 | 14 | $1 \ldots 10$ | 31 | 31 | 45 | Process voltage is switched off (ON -> OFF) | Process voltage ON |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |
| Channel error |  |  |  |  |  |  |  |
| 4 | 14 | $1 . .10$ | 3 | $0 \ldots 15$ | 48 | Analog value overflow at an analog output | Check output value |
|  | 11 / 12 | ADR | 1... 10 |  |  |  |  |
| 4 | 14 | $1 \ldots 10$ | 3 | $0 \ldots 15$ | 7 | Analog value underflow at an analog output | Check output value |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |

Remarks:

| ${ }^{1}$ ) | In AC500, the following interface identifier applies: $14=$ I/O bus, $11=$ COM1 (e.g. CS31 bus), $12=$ COM2. <br> The FBP diagnosis block does not contain this identifier. |
| :---: | :---: |
| ${ }^{2}$ ) | With "Device" the following allocation applies: <br> 31 = module itself, $1 \ldots 10$ = expansion module $1 \ldots 10$, ADR = hardware address (e.g. of the DC551) |
| ${ }^{3}$ ) | With "Module" the following allocation applies dependent of the master: <br> Module error: I/O bus or FBP: 31 = module itself; COM1/COM2: 1 ... $10=$ expansion 1 ... 10 <br> Channel error: I/O bus or FBP = module type (3 = AO); COM1/COM2: 1 ... 10 = expansion $1 . . .10$ |
| 4) | In case of module errors, with channel "31 = Module itself" is output. |

## State LEDs

During the power ON procedure, the module initializes automatically. All LEDs (except the channel LEDs) are ON during this time.


## Output ranges

## Output ranges voltage and current

The represented resolution corresponds to 16 bits.

| Range | $\begin{aligned} & \hline-10 \text { V } \ldots+10 \\ & \mathrm{~V} \end{aligned}$ | $\begin{aligned} & \mathrm{OmA} \\ & \mathrm{~mA} \end{aligned}$ | $4 \mathrm{~mA} \ldots 20$ | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  | Decimal | Hex. |
| Overflow | > 11.7589 V | $\begin{aligned} & \hline>23.5178 \\ & \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & >22.8142 \\ & \mathrm{~mA} \end{aligned}$ | > 32511 | > 7EFF |
| Value too high | $\begin{aligned} & 11.7589 \mathrm{~V} \\ & : \\ & 10.0004 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 23.5178 \mathrm{~mA} \\ & : \\ & 20.0007 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 22.8142 \mathrm{~mA} \\ & : \\ & 20.0006 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 32511 \\ & : \\ & 27649 \end{aligned}$ | $\begin{aligned} & \text { 7EFF } \\ & : \\ & 6 \mathrm{C} 01 \end{aligned}$ |
| Normal range | $\begin{aligned} & 10.0000 \mathrm{~V} \\ & : \\ & 0.0004 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 20.0000 \mathrm{~mA} \\ & : \\ & 0.0007 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 20.0000 \mathrm{~mA} \\ & : \\ & 4.0006 \mathrm{~mA} \end{aligned}$ | $27648$ | $\begin{aligned} & \hline 6 \mathrm{C} 00 \\ & : \\ & 0001 \end{aligned}$ |
|  | 0.0000 V | 0.0000 mA | 4.0000 mA | 0 | 0000 |
|  | $\begin{aligned} & -0.0004 \mathrm{~V} \\ & : \\ & -10.0000 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 0 \mathrm{~mA} \\ & : \\ & 0 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 3.9994 \mathrm{~mA} \\ & 0 \mathrm{~mA} \\ & 0 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & \hline-1 \\ & -6912 \\ & -27648 \end{aligned}$ | $\begin{aligned} & \text { FFFF } \\ & \text { E500 } \\ & 9400 \end{aligned}$ |


| Range | $\begin{aligned} & -10 \text { V } \ldots+10 \\ & V \end{aligned}$ | $\begin{aligned} & 0 \mathrm{~mA} \ldots 20 \\ & \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & \hline 4 \mathrm{~mA} \ldots 20 \\ & \mathrm{~mA} \end{aligned}$ | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  | Decimal | Hex. |
| Value too low | $-10.0004 \mathrm{~V}$ -11.7589 V | $\begin{aligned} & 0 \mathrm{~mA} \\ & : \\ & 0 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 0 \mathrm{~mA} \\ & : \\ & 0 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & -27649 \\ & : \\ & -32512 \end{aligned}$ | $\begin{aligned} & \text { 93FF } \\ & : \\ & 8100 \end{aligned}$ |
| Underflow | 0 V | 0 mA | 0 mA | <-32512 | < 8100 |

## Technical data

## Technical data of the module

The system data of AC500 and S500 are applicable to the standard version ${ }^{\circledR}$ Chapter 5.1.2 "System data AC500" on page 166.

The system data of AC500-XC are applicable to the XC version ${ }^{\Perp}$ Chapter 5.1.3 "System data AC500-XC" on page 169.
Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter | Value |
| :---: | :---: |
| Process voltage |  |
| Connections | Terminals $1.8,2.8,3.8$ and 4.8 for +24 V (UP) as well as 1.9, 2.9, 3.9 and 4.9 for 0 V (ZP) |
| Rated value | 24 V DC |
| Max. ripple | 5 \% |
| Protection against reversed voltage | Yes |
| Rated protection fuse on UP | 10 A fast |
| Galvanic isolation | Yes, per module |
| Current consumption |  |
| From 24 V DC power supply at the terminals UP/L+ and ZP/M of the CPU/communication interface module | Ca. 2 mA |
| Current consumption from UP at normal operation | 0.15 A + output loads |
| Inrush current from UP (at power up) | 0.040 A ${ }^{2} \mathrm{~s}$ |
| Max. length of analog cables, conductor cross section $>0.14 \mathrm{~mm}^{2}$ | 100 m |
| Weight | 300 g |
| Mounting position | Horizontal or vertical with derating (output load reduced to $50 \%$ at $+40{ }^{\circ} \mathrm{C}$ per group) |
| Cooling | The natural convection cooling must not be hindered by cable ducts or other parts in the control cabinet. |

## NOTICE!

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and temporary overvoltage up to 30 V DC.

## Technical data of the analog outputs

| Parameter | Value |  |
| :---: | :---: | :---: |
| Number of channels per module | 16, of for vo 012 | ich channnels O 0 ... O 3 and $\mathrm{O} 8 \ldots \mathrm{O} 11$ e and current, and channels O 4 ... 7 and only for voltage |
| Distribution of channels into groups | 1 group |  |
| Channels O0- ... O7- <br> Channels O0+ ... O7+ | Terminals 1.0 ... 1.7 <br> Terminals 2.0 ... 2.7 |  |
| Channels O8- ... O15- <br> Channels O8+ ... O15+ | Terminals 3.0 ... 3.7 <br> Terminals 4.0 ... 4.7 |  |
| Output type | Bipolar with voltage, unipolar with current |  |
| Galvanic isolation | Against internal supply and other modules |  |
| Configurability | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}, 0 \mathrm{~mA} . .20 \mathrm{~mA}, 4 \mathrm{~mA} . .20 \mathrm{~mA}$ (each output can be configured individually), current outputs only channels $0 . . .3$ and $8 \ldots 11$ |  |
| Output resistance (load), as current output | $0 \Omega \ldots 500 \Omega$ |  |
| Output loadability, as voltage output | Max. $\pm 10 \mathrm{~mA}$ |  |
| Indication of the output signals | One LED per channel |  |
| Resolution | 12 bits including sign |  |
| Settling time for full range change (resistive load, output signal within specified tolerance) | Typ. 5 ms |  |
| Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range | Typ. | $\pm 0.5 \%$ of full scale at $+25^{\circ} \mathrm{C}$ |
|  | Max. | $\pm 1$ \% of full scale (all ranges) at $0^{\circ} \mathrm{C} \ldots+60^{\circ} \mathrm{C}$ or EMC disturbance |
| Relationship between output signal and hex code | y Chapter 5.2.4.4.2.4.10 "Output ranges" on page 776 |  |
| Unused outputs | Can be left open-circuited |  |

## Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm

The dimensions are in mm and in brackets in inch.

Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 250 200 R0001 | AO523, analog output module, 16 AO, <br> U/I, 12 bits including sign, 2-wires | Active |
| 1SAP 450 200 R0001 | AO523-XC, analog output module, <br> 16 AO, U/I, 12 bits including sign, <br> 2-wires, XC version | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

## AX521-Analog input/output module

## Features

- 4 configurable analog inputs (IO ... I3) in 1 group (1.0 ... 2.3)

Resolution 12 bits including sign

- 4 configurable analog outputs ( O 0 ... O 3 ) in 1 group ( 3.0 ... 4.3)

Resolution 12 bits including sign

- Module-wise galvanically isolated
- XC version for use in extreme ambient conditions available


1 I/O bus
2 Allocation between terminal number and signal name
34 yellow LEDs to display the signal states at the analog inputs (IO ... I3)
44 yellow LEDs to display the signal states at the analog outputs ( $\mathrm{O} 0 \ldots \mathrm{O} 3$ )
51 green LED to display the state of the process supply voltage UP
62 red LEDs to display errors
7 Label
8 Terminal unit
9 DIN rail
Sign for XC version

## Intended purpose

The device can be used as a decentralized I/O extension module for S 500 communication interface modules (e. g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs.

## Functionality

AX521 4 analog inputs (channel 0... channel 3), individually configurable for

- Unused (default setting)
- 0 V ... 10 V
- $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$
- 0 mA ... 20 mA
- 4 mA ... 20 mA
- Pt100, $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ (2-wire)
- Pt100, $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ (3-wire), requires 2 channels
- Pt100, $-50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ (2-wire)
- Pt100, $-50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ (3-wire), requires 2 channels
- Pt1000, $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ (2-wire)
- Pt1000, $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ (3-wire), requires 2 channels
- Ni1000, $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ (2-wire)
- Ni1000, $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ (3-wire), requires 2 channels
- $0 \vee \ldots 10 \mathrm{~V}$ with differential inputs, requires 2 channels
- $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ with differential inputs, requires 2 channels
- Digital signals (digital input)

4 analog outputs (channel $0 \ldots$ channel 3 ), individually configurable for

- Unused (default setting)
- -10 V ... +10 V
- $0 \mathrm{~mA} . .20 \mathrm{~mA}$
- $4 \mathrm{~mA} . .20 \mathrm{~mA}$

| Parameter | Value |
| :---: | :---: |
| Resolution of the analog channels |  |
| Voltage -10 V ... +10 V | 12 bits including sign |
| Voltage $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ | 12 bits |
| Current 0 mA ... $20 \mathrm{~mA}, 4 \mathrm{~mA}$... 20 mA | 12 bits |
| Temperature | $+0.1^{\circ} \mathrm{C}$ |
| LED displays | 11 LEDs for signals and error messages |
| Internal power supply | Via the I/O bus interface (I/O bus) |
| External power supply | Via the terminals ZP and UP (process voltage 24 V DC) |
| Required terminal unit | TU515 or TU516 $\stackrel{y y}{*}$ Chapter 5.2.5.2 "TU515, TU516, TU541 and TU542 for I/O modules" on page 938 |

## Connections

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

The modules are plugged on an I/O terminal unit $\Leftrightarrow$ Chapter 5.2.5.2 "TU515, TU516, TU541 and TU542 for I/O modules" on page 938. Properly position the modules and press until they lock in place. The terminal units are mounted on a DIN rail or with 2 screws plus the additional accessory for wall mounting ${ }^{\text {}} \gg$ Chapter 5.2.8.2.5 "TA526-Wall mounting accessory" on page 1324.
The connection of the I/O channels is carried out using the 40 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

The terminals $1.8,2.8,3.8$ and 4.8 as well as $1.9,2.9,3.9$ and 4.9 are electrically interconnected within the I/O terminal units and have always the same assignment, irrespective of the inserted module:

Terminals 1.8, 2.8, 3.8 and 4.8: process voltage UP $=+24 \mathrm{~V}$ DC
Terminals 1.9, 2.9, 3.9 and 4.9: process voltage $\mathrm{ZP}=0 \mathrm{~V}$ DC

Table 175: Assignment of the other terminals

| Terminals | Signal | Description |
| :--- | :--- | :--- |
| $1.0 \ldots 1.3$ | $\mathrm{IO} \ldots \mathrm{I} 3-$ | Negative poles of the 4 analog <br> inputs |
| $2.0 \ldots 2.3$ | $\mathrm{IO}+\ldots \mathrm{I} 3+$ | Positive poles of the 4 analog <br> inputs |
| $3.0 \ldots 3.3$ | $\mathrm{O} 0-\ldots \mathrm{O} 3-$ | Negative poles of the 4 analog <br> outputs |
| $4.0 \ldots 4.3$ | $\mathrm{O}+\ldots \mathrm{O} 3+$ | Positive poles of the 4 analog <br> outputs |

The negative poles of the analog inputs are connected to each other to form an "Analog Ground" signal for the module.

The negative poles of the analog outputs are connected to each other to form an "Analog Ground" signal for the module.

There is no galvanic isolation between the analog circuitry and ZP/UP. Therefore, the analog sensors must be galvanically isolated in order to avoid loops via the ground potential or the supply voltage.

Because of their common reference potential, analog current inputs cannot be circuited in series, neither within the module nor with channels of other modules.

For the open-circuit detection (wire break), each analog input channel is pulled up to "plus" by a high-resistance resistor. If nothing is connected, the maximum voltage will be read in then.

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a communication interface module or a CPU). Thus, the current consumption from 24 V DC power supply at the terminals L+/UP and M/ZP of the CPU/communication interface module increases by 2 mA per I/O module.
The external power supply connection is carried out via the UP (+24 V DC) and the ZP (0 V DC) terminals.

## WARNING!

## Removal/Insertion under power

Removal or insertion under power is permissible only if all conditions for hot swapping are fullfilled.
(7) Chapter 4.9.3 "Replace an I/O module with hot swap" on page 153

The devices are not designed for removal or insertion under power when the conditions for hot swap do not apply. Because of unforeseeable consequences, it is not allowed to plug in or unplug devices with the power being ON.
Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while they are energized in a hazardous location could result in an electric arc, which could create an ignition source resulting in fire or explosion.
Prior to proceeding, make sure that power is been disconnected and that the area has been thoroughly checked to ensure that flammable materials are not present.
The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

## Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

Generally, analog signals must be laid in shielded cables. The cable shields must be grounded at both sides of the cables. In order to avoid unacceptable potential differences between different parts of the installation, low resistance equipotential bonding conductors must be laid.


Fig. 118: Connection of the I/O module: 4 analog inputs and 4 analog outputs, individually configurable ${ }^{\text {® }}$ Chapter 5.2.4.4.2.5.3 "Functionality" on page 781

## CAUTION!

By installing equipotential bonding conductors between the different parts of the system, it must be ensured that the potential difference between ZP and AGND never can exceed 1 V .

## CAUTION!

The process supply voltage must be included in the grounding concept (e. g. grounding of the negative terminal).

## Connection of resistance thermometers in 2-wire configuration

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the I/O module provides a constant current source which is multiplexed over the 8 analog channels.


Fig. 119: Connection example

| Pt100 | $-50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ | 2-wire configuration, one <br> channel used |
| :--- | :--- | :--- |
| Pt100 | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 2-wire configuration, one <br> channel used |
| Pt1000 | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 2-wire configuration, one <br> channel used |
| Ni1000 | $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ | 2-wire configuration, one <br> channel used |

The I/O module performs a linearization of the resistance characteristic.
In order to avoid error messages from unused analog input channels, it is useful to configure them as "unused".

## Connection of resistance thermometers in 3-wire configuration

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the I/O module provides a constant current source which is multiplexed over the max. 8 (depending on the configuration) analog channels.


Fig. 120: Connection example

If several measuring points are adjacent to each other, only one return line is necessary. This saves wiring costs.

With the 3-wire configuration, two adjacent analog channels belong together (e.g. the channels 0 and 1 ). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0 ), the next higher address must be the odd address (channel 1).
The constant current of one channel flows through the resistance thermometer. The constant current of the other channel flows through one of the cores. The module calculates the measured value from the two voltage drops and stores it under the input with the higher channel number (e.g. I1).
In order to keep measuring errors as small as possible, it is necessary to have all the involved conductors in the same cable. All the conductors must have the same cross section.

| $\operatorname{Pt100}$ | $-50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ | 3-wire configuration, two <br> channels used |
| :--- | :--- | :--- |
| $\mathrm{Pt100}$ | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 3-wire configuration, two <br> channels used |
| $\mathrm{Pt1000}$ | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 3-wire configuration, two <br> channels used |
| Ni1000 | $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ | 3-wire configuration, two <br> channels used |

The I/O module performs a linearization of the resistance characteristic.
In order to avoid error messages from unused analog input channels, it is useful to configure them as "unused".


Fig. 121: Connection example

By connecting the sensor's negative pole of the output voltage to AGND, the galvanically isolated voltage source of the sensor is referred to ZP.

The following measuring ranges can be configured for AX521 ${ }^{〔}$, Chapter 5.2.4.4.2.5.7 "Parameterization" on page 796 and for AX522 ${ }^{〔}$ Chapter 5.2.4.4.2.6.7 "Parameterization" on page 826:

| Voltage | $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ | 1 channel used |
| :--- | :--- | :--- |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | 1 channel used |

In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as "unused".

## Connection of active-type analog sensors (Current) with galvanically isolated power supply



Fig. 122: Connection example

| Current | $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 1 channel used |
| :--- | :--- | :--- |
| Current | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 1 channel used |

Unused input channels can be left open-circuited, because they are of low resistance.


Fig. 123: Connection example

## CAUTION!

The potential difference between AGND and ZP at the module must not be greater than 1 V , not even in case of long lines (see figure Terminal Assignment).

If AGND does not get connected to ZP, the sensor current flows to $Z P$ via the AGND line. The measuring signal is distorted, as a very small current flows through the voltage line. The total current through the PTC should not exceed 50 mA . This measuring method is therefore only suitable for short lines and small sensor currents. If there are bigger distances, the difference measuring method should be applied.

| Voltage | $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ | 1 channel used |
| :--- | :--- | :--- |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V} *)$ | 1 channel used |

*) if the sensor can provide this signal range
In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as "unused".

## Connection of passive-type analog sensors (Current)



Fig. 124: Connection example

| Current | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 1 channel used |
| :--- | :--- | :--- |

## CAUTION!

If, during initialization, an analog current sensor supplies more than 25 mA for more than 1 second to an analog input, this input is switched off by the module (input protection). In such cases, it is recommended to protect the analog input by a 10 -volt Zener diode (in parallel to I+ and ZP). But, in general, sensors with fast initialization or without current peaks higher than 25 mA are preferrable.

Unused input channels can be left open-circuited because they are of low resistance.

## Connection of active-type analog sensors (Voltage) to differential inputs

Differential inputs are very useful if analog sensors are used which are remotely non-isolated (e.g. the minus terminal is remotely grounded).

The use of differential inputs helps to considerably increase the measuring accuracy and to avoid ground loops.
With differential input configurations, two adjacent analog channels belong together (e.g. the channels 0 and 1 ). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0 ), the next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1).

The analog value is calculated by subtraction of the input value with the higher address from the input value of the lower address.

The converted analog value is available at the odd channel (higher address).

## CAUTION!

The ground potential at the sensors must not have too large a potential difference with respect to ZP (max. $\pm 1 \mathrm{~V}$ within the full signal range). Otherwise, problems may occur concerning the common-mode input voltages of the involved analog inputs.


Fig. 125: Connection example

> The negative pole of the sensor must be grounded next to the sensor.

| Voltage | $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ | with differential inputs, 2 chan- <br> nels used |
| :--- | :--- | :--- |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | with differential inputs, 2 chan- <br> nels used |

In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as "unused".

## Use of analog inputs as digital inputs

Several (or all) analog inputs can be configured as digital inputs. The inputs are not galvanically isolated against the other analog channels.


Fig. 126: Connection example

| Digital input | 24 V | 1 channel used |
| :--- | :--- | :--- |
| Effect of incorrect input ter- <br> minal connection |  | Wrong or no signal detected, <br> no damage up to 35 V |



Fig. 127: Connection example

| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | Load max. $\pm 10 \mathrm{~mA}$ | 1 channel used |
| :--- | :--- | :--- | :--- |
| Current | $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | Load $0 \Omega \ldots 500 \Omega$ | 1 channel used |
| Current | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | Load $0 \Omega \ldots 500 \Omega$ | 1 channel used |

Only the channels $0 \ldots 3$ can be configured as current output ( $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ or $4 \mathrm{~mA} . .20 \mathrm{~mA}$ ).
Unused analog outputs can be left open-circuited.

## Internal data exchange

| Digital inputs (bytes) | 0 |
| :--- | :--- |
| Digital outputs (bytes) | 0 |
| Counter input data (words) | 4 |
| Counter output data (words) | 4 |

## I/O configuration

The module does not store configuration data itself. It gets its parameterization data from the master device of the I/O bus (CPU or communication interface module) during power-up of the system.

Hence, replacing I/O modules is possible without any re-parameterization via software.

## Parameterization

| Firmware version | Configuration |
| :--- | :--- |
| Firmware version > V2.0.0 | The arrangement of the parameter data is per- <br> formed by Control Builder Plus/ Automation <br> Builder software. |

The parameter data directly influences the functionality of modules.
For non-standard applications, it is necessary to adapt the parameters to your system configuration.

Module: Module slot address: $\mathrm{Y}=1$... 10

| No. | Name | Value | Internal <br> value | Internal <br> value, <br> type | Default | Min. | Max. | EDS <br> Slot/ <br> Index |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| 1 | Module <br> ID | Internal | 1505 <br> 1 | Word | 1505 <br> $0 \times 05 \mathrm{E} 1$ | 0 | 65535 | $0 \times 0 \mathrm{Y01}$ |
| 2 | Ignore <br> module <br> $\left.{ }_{2}\right)$ | No <br> Yes | 0 <br> 1 | Byte | No <br> $0 \times 00$ |  |  | Not for <br> FBP |
| 3 | Param- <br> eter <br> length in <br> bytes | Internal | 21 | Byte | $21-\mathrm{CPU}$ <br> 21-FBP | 0 | 255 | $0 \times 0 \mathrm{Y02}$ |


| No. | Name | Value | Internal value | Internal value, type | Default | Min. | Max. | EDS Slot/ Index |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 4 | Check supply | $\begin{aligned} & \text { Off } \\ & \text { On } \end{aligned}$ | $1$ | Byte | $\begin{aligned} & \text { On } \\ & 0 \times 01 \end{aligned}$ | 0 | 1 | 0x0Y03 |
| 5 | Analog data format | Default | 0 | Byte | Default <br> 0x00 |  |  | 0x0Y04 |
| 6 | Behaviour of outputs at com-munication errors | Off <br> Last value <br> Substitute value | $\begin{aligned} & \hline 0 \\ & 1+\left(n^{*} 5\right) \\ & 2+\left(n^{*} 5\right), \\ & n \leq 2 \end{aligned}$ | Byte | $\begin{aligned} & \text { Off } \\ & 0 \times 00 \end{aligned}$ | 0 | 2 | 0x0Y05 |
| 7 | Channel configuration Input channel 0 | See tabl configur Table nel config on page | 'Channel tion' <br> 177 "Chan guration ${ }^{2}$ )" 799 | Byte | Default <br> 0x00 | 0 | 19 | 0x0Y06 |
| 8 | Channel monitoring Input channel 0 | See tabl monitori * Table nel mon on page | 'Channel g' <br> 178 "Chan oring ${ }^{3}$ )" 799 | Byte | $\begin{array}{\|l\|} \hline \text { Default } \\ 0 \times 00 \end{array}$ | 0 | 3 | 0x0Y07 |
| $\begin{aligned} & 9 \\ & \text { to } \\ & 14 \end{aligned}$ | Channel configu- <br> ration <br> and <br> channel <br> moni- <br> toring of <br> the input <br> channels <br> 1 to 3 | See tabl 'Channe tion' <br> 4 Table nel config on page 'Channe ing' (4) Table nel mon on page | configura- <br> 177 "Chan guration ²)" 799 and monitor- <br> 178 "Chan oring ${ }^{3}$ )" 799 | $\begin{array}{\|l\|l\|} \hline \text { Byte } \\ \text { Byte } \end{array}$ | $\begin{aligned} & \text { Default } \\ & 0 \times 00 \\ & 0 \times 00 \end{aligned}$ | $0$ | $\begin{aligned} & 19 \\ & 3 \end{aligned}$ | 0x0Y08 to <br> OxOYOD |
| 15 | Channel configuration <br> Output channel 0 | See tabl configur * Table nel config on page | 'Channel tion' <br> 177 "Chan guration ${ }^{2}$ )" 799 | Byte | $\begin{aligned} & \text { Default } \\ & \text { ox00 } \end{aligned}$ | 0 | 130 | 0xOYOE |
| 16 | Channel monitoring Output channel 0 | See tabl monitori « Table nel mon on page | 'Channel ' <br> 178 "Chan oring ${ }^{3}$ )" 799 | Byte | $\begin{array}{\|l\|} \hline \text { Default } \\ 0 \times 00 \end{array}$ | 0 | 3 | 0xOYOF |


| No. | Name | Value | Internal value | Internal value, type | Default | Min. | Max. | $\begin{array}{\|l\|} \hline \text { EDS } \\ \text { Slot/ } \\ \text { Index } \end{array}$ |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 17 | Substi- <br> tute <br> value <br> Output channel 0 | only valid for output channel 0 | $0 . . .0 x$ ffff | Word | Default <br> 0x0000 | 0 | 65535 | 0x0Y10 |
| 18 to 21 | Channel configu- <br> ration <br> and <br> channel <br> moni- <br> toring of <br> the <br> output <br> channels <br> 1 to 2 | See table 'Channel tion' <br> ③ Table nel config on page 'Channel ing' $\stackrel{4}{4}$ Table nel moni on page | configura- <br> 77 "Chan uration ${ }^{2}$ " 79 and monitor- <br> 78 "Chan oring ${ }^{3}$ )" 79 | $\begin{array}{\|l\|l\|} \text { Byte } \\ \text { Byte } \end{array}$ | $\begin{aligned} & \text { Default } \\ & 0 \times 00 \\ & 0 \times 00 \end{aligned}$ | $0$ | $\begin{array}{\|l\|} \hline 130 \\ 3 \end{array}$ | 0x0Y11 to 0x0Y14 |
| 22 | Channel configuration Output channel 3 | See table monitorin $\Leftrightarrow$ Table nel config on page | 'Channel <br> 77 "Chan <br> uration ${ }^{2}$ )" <br> 99 | Byte | Default <br> 0x00 | 0 | 130 | 0x0Y15 |
| 23 | Channel monitoring Output channel 3 | See table monitorin \& Table nel monit on page | 'Channel <br> 78 "Chan ring ${ }^{3}$ )" <br> 99 | Byte | Default <br> 0x00 | 0 | 3 | 0x0Y16 |
| ${ }^{1}$ ) With CS31 and addresses less than 70 and FBP, the value is increased by 1 ${ }^{2}$ ) Not with FBP |  |  |  |  |  |  |  |  |

GSD file:

| Ext_User_Prm_Data_Len $=$ |  |
| :--- | :--- |
| Ext_User_Prm_Data_Const $(0)=$ | 24 |
|  | $0 \times 05,0 \times e 2,0 \times 15,1$ |
|  | $0 \times 01,0 \times 00,0 \times 00 \backslash$ |
|  | $0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00$, |
| $0 \times 00,1$ |  |
|  | $0 \times 00,0 \times 00,0 \times 00,0 \times 00,1$ |
|  | $0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00 ;$ |

Table 176: Input channel (4x)

| No. | Name | Internal value, type | Default |
| :--- | :--- | :--- | :--- |
| 1 | Channel configuration $_{\text {see table }{ }^{2} \text { ) }}$ | Byte | 0 <br> $0 \times 00$ see table $\left.{ }^{2}\right)$ |
| 2 | Channel monitoring $\left._{\text {see table }}{ }^{3}\right)$ | Byte | 0 |
| $0 \times 00$ see table $\left.{ }^{3}\right)$ |  |  |  |

Table 177: Channel configuration ${ }^{2}$ )

| Internal value | Operating modes of the analog inputs, individually configurable |
| :---: | :---: |
| 0 | Unused (default) |
| 1 | Analog input 0 V ... 10 V |
| 2 | Digital input |
| 3 | Analog input 0 mA ... 20 mA |
| 4 | Analog input 4 mA ... 20 mA |
| 5 | Analog input -10 V ... +10 V |
| 8 | Analog input Pt100, $-50{ }^{\circ} \mathrm{C} \ldots+400{ }^{\circ} \mathrm{C}$ (2-wire) |
| 9 | Analog input Pt100, $-50^{\circ} \mathrm{C} \ldots+40{ }^{\circ} \mathrm{C}$ (3-wire), requires 2 channels *) |
| 10 | Analog input $0 \ldots 10 \mathrm{~V}$ via differential inputs, requires 2 channels *) |
| 11 | Analog input $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ via differential inputs, requires 2 channels *) |
| 14 | Analog input Pt100, $-50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ (2-wire) |
| 15 | Analog input Pt100, $-50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ (3-wire), requires 2 channels *) |
| 16 | Analog input Pt1000, $-50^{\circ} \mathrm{C} \ldots+400{ }^{\circ} \mathrm{C}$ (2-wire) |
| 17 | Analog input Pt1000, $-50^{\circ} \mathrm{C} \ldots+400{ }^{\circ} \mathrm{C}$ (3-wire), requires 2 channels *) |
| 18 | Analog input Ni1000, $-50^{\circ} \mathrm{C} \ldots+150{ }^{\circ} \mathrm{C}$ (2-wire) |
| 19 | Analog input Ni1000, $-50^{\circ} \mathrm{C} \ldots+150{ }^{\circ} \mathrm{C}$ (3-wire), requires 2 channels *) |
|  | *) In the operating modes with 3-wire configuration or with differential inputs, two adjacent analog inputs belong together (e.g. the channels 0 and 1). In these cases, both channels are configured in the desired operating mode. The lower address must be the even address (channel 0). The next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1). |

Table 178: Channel monitoring ${ }^{3}$ )

| Internal value | Monitoring |
| :--- | :--- |
| 0 | Plausibility, open-circuit (broken wire) and short circuit |
| 3 | No monitoring |

Table 179: Output channel 0 (1 channel)

| No. | Name | Value | Internal value | Internal <br> value, type | Default |
| :--- | :--- | :--- | :--- | :--- | :--- |
| 1 | Channel con- <br> figuration | see table $^{4}$ ) | see table ${ }^{4}$ ) | Byte | see table ${ }^{4}$ ) |
| 2 | Channel mon- $_{\text {itoring }}$ | see table $^{5}$ ) | see table ${ }^{5}$ ) | Byte | see table $\left.{ }^{5}\right)$ |
| 3 | Substitute <br> value <br> see table $\left.{ }^{6}\right)$ | $0 \ldots 65535$ | $0 \ldots$ <br> $0 x f f f$ | Word | 0 |

Table 180: Output channels 1 ... 3 (3x)

| No. | Name | Internal value, type |
| :--- | :--- | :--- |
| 1 | Channel configuration $^{\text {see table }}$ ) | Byte |
| 2 | Channel monitoring $\left.^{\text {see table }}{ }^{6}\right)$ | Byte |

Table 181: Channel configuration ${ }^{4}$ )

| Internal value | Operating modes of the analog outputs, individually configurable |
| :--- | :--- |
| 0 | Unused (default) |
| 128 | Analog output $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ |
| 129 | Analog output $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ (not with the channels $4 \ldots 7$ and $12 \ldots$ 15) |
| 130 | Analog output $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ (not with the channels $4 \ldots 7$ and $12 \ldots$ 15) |

Table 182: Channel monitoring ${ }^{5}$ )

| Internal value | Monitoring |
| :--- | :--- |
| 0 | Plausibility, open circuit (broken wire) and short circuit (default) |
| 3 | No monitoring |

Table 183: Substitute value ${ }^{6}$ )

| Intended behaviour of <br> output channel when the <br> control system stops | Required setting of the <br> module parameter "Behav- <br> iour of outputs in case of a <br> communication error" | Required setting of the <br> channel parameter "Substi- <br> tute value" |
| :--- | :--- | :--- |
| Output OFF | Off | 0 |
| Last value infinite | Last value | 0 |
| Last value for 5 s and then <br> turn off | Last value 5 sec | 0 |
| Last value for 10 s and then <br> turn off | Last value 10 sec | 0 |
| Substitute value infinite | Substitute value | Depending on configuration |


| Intended behaviour of <br> output channel when the <br> control system stops | Required setting of the <br> module parameter "Behav- <br> iour of outputs in case of a <br> communication error" | Required setting of the <br> channel parameter "Substi- <br> tute value" |
| :--- | :--- | :--- |
| Substitute value for 5 s and <br> then turn off | Substitute value 5 sec | Depending on configuration |
| Substitute value for 10 s and <br> then turn off | Substitute value 10 sec | Depending on configuration |

## Diagnosis

Table 184: Possible diagnosis of I/O channels

| Output range | Condition |  |
| :--- | :--- | :--- |
|  | Output value in the PLC <br> underflow | Output value in the PLC overflow |
| $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | Error identifier $=7$ | Error identifier $=4$ |
| $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ |  |  |
| $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ |  |  |


| Input range | Condition |  |  |  |
| :--- | :--- | :--- | :--- | :--- |
|  | Short circuit | Wire break | Input value under- <br> flow | Input value over- <br> flow |
| $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | no diagnosis possible | no diagnosis possible | no diagnosis possible | Error identifier $=48$ |
| $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | Error identifier $=7$ | Error identifier $=7$ | Error identifier $=7$ | Error identifier $=48$ |
| $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | no diagnosis possible | Error identifier $=48$ | Error identifier $=7$ | Error identifier $=48$ |

Table 185: Content of diagnosis messages

| E1 ... E4 | d1 | d2 | d3 | d4 | Identifier <br> $\mathbf{0 0 0} \ldots$ <br> 063 | AC500 <br> display | <- Display in |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Class | Comp | Dev | Mod | Ch | Err | PS501 <br> PLC <br> browser |  |


| E1 ... E4 | d1 | d2 | d3 | d4 |  | $\begin{aligned} & \text { Identifier } \\ & 000 \ldots \\ & 063 \end{aligned}$ | AC500 display | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch |  | Err | $\begin{array}{\|l} \text { PS501 } \\ \text { PLC } \\ \text { browser } \end{array}$ |  |  |
| Byte 6 <br> Bit 6 ... 7 | - | Byte 3 | Byte 4 | Byte 5 |  | Byte 6 <br> Bit 0 ... 5 | FBP diagnosis block |  |  |
| Class | Interface | Device | Module | Channel |  | Error Identifier | Error message |  | Remedy |
|  | $\left.{ }^{1}\right)$ | ${ }^{2}$ ) | $\left.{ }^{3}\right)$ | ${ }^{4}$ ) |  |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 |  | 40 | Different hard-/firmware versions in the module |  | Replace I/O module |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 |  | 43 | Internal error in the module |  | Replace I/O module |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 |  | 36 | Internal data exchange failure |  | Replace I/O module |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 |  | 9 | Overflow diagnosis buffer |  | New start |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 |  | 26 | Parameter error |  | Check master |
|  | 11/12 | ADR | $1 . .10$ |  |  |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 |  | 11 | Process voltage too low |  | Check process voltage |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |  |  |
| 4 | 14 | 1... 10 | 31 | 31 |  | 45 | Process voltage is switched off (ON -> OFF) |  | Process voltage ON |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |  |  |
| Channel error |  |  |  |  |  |  |  |  |  |
| - |  |  |  | AX521 | AX522 |  |  |  |  |
| 4 | 14 | 1... 10 | 1 | 0... 3 | 0... 7 | 48 | Analog value overflow or broken wire at an analog input |  | Check input value or terminal |
|  | $11 / 12$ | ADR | $1 . .10$ |  |  |  |  |  |  |
| 4 | 14 | 1... 10 | 1 | 0... 3 | $0 \ldots 7$ | 7 | Analog value underflow at an analog input |  | Check input value |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |  |  |
| 4 | 14 | 1... 10 | 1 | $0 \ldots 3$ | $0 . . .7$ | 47 | Short circuit at an analog input |  | Check terminal |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |  |  |
| 4 | 14 | 1... 10 | 3 | $4 \ldots 7$ | $8 \ldots 15$ | 4 | Analog value overflow at an analog output |  | Check output value |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |  |  |
| 4 | 14 | 1... 10 | 3 | $4 \ldots 7$ | 8... 15 | 7 | Analog value underflow at an analog output |  | Check output value |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |  |  |

Remarks:

| ${ }^{1}$ ) | In AC500, the following interface identifier applies: 14 = I/O bus, 11 = COM1 (e.g. CS31 bus), $12=$ COM2. <br> The FBP diagnosis block does not contain this identifier. |
| :---: | :---: |
| ${ }^{2}$ ) | With "Device" the following allocation applies: <br> 31 = module itself, $1 \ldots 10=$ expansion module $1 \ldots 10$, ADR = hardware address (e.g. of the DC551) |
| $\left.{ }^{3}\right)$ | With "Module" the following allocation applies depending on the master: <br> Module error: I/O bus or FBP: 31 = module itself; COM1/COM2: 1 ... $10=$ expansion 1 ... 10 <br> Channel error: I/O bus or FBP = module type ( 1 = AI, 3 = AO); COM1/COM2: <br> 1 ... 10 = expansion $1 . . .10$ |
| $\left.{ }^{4}\right)$ | In case of module errors, with channel "31 = Module itself" is output. |

State LEDs
During the power ON procedure, the module initializes automatically. All LEDs (except the channel LEDs) are ON during this time.


## Measuring ranges

Input ranges of voltage, current and digital input
The represented resolution corresponds to 16 bits.

| Range | $0 \mathrm{~V} . .10 \mathrm{~V}$ | $\begin{aligned} & -10 \mathrm{~V} \ldots \\ & +10 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 0 \mathrm{~mA} \ldots \\ & 20 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 4 \mathrm{~mA} \ldots \\ & 20 \mathrm{~mA} \end{aligned}$ | Digital input | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  | Decimal | Hex. |
| Overflow | >11.7589 | >11.7589 | >23.5178 | >22.8142 |  | 32767 | 7FFF |
| Meas ured value too high | $\begin{aligned} & 11.7589 \\ & \cdot \\ & 10.0004 \end{aligned}$ | $\begin{aligned} & 11.7589 \\ & : \\ & 10.0004 \end{aligned}$ | $\begin{aligned} & 23.5178 \\ & : \\ & 20.0007 \end{aligned}$ | $\begin{aligned} & 22.8142 \\ & : \\ & 20.0006 \end{aligned}$ |  | $\begin{array}{\|l} 32511 \\ : \\ 27649 \end{array}$ | $\begin{aligned} & \text { 7EFF } \\ & : \\ & 6 \mathrm{C} 01 \end{aligned}$ |
|  | $\begin{aligned} & 10.0000 \\ & : \\ & 0.0004 \end{aligned}$ | $\begin{aligned} & 10.0000 \\ & : \\ & 0.0004 \end{aligned}$ | $\begin{aligned} & 20.0000 \\ & : \\ & 0.0007 \end{aligned}$ | $\begin{aligned} & 20.0000 \\ & : \\ & 4.0006 \end{aligned}$ | ON | $27648$ <br> 1 | $\begin{aligned} & \hline 6 \mathrm{C} 00 \\ & : \\ & 0001 \end{aligned}$ |
|  | 0.0000 | 0.0000 | 0 | 4 | OFF | 0 | 0000 |
|  | $\begin{array}{\|l\|} \hline-0.0004 \\ -1.7593 \\ \hline \end{array}$ | $\begin{aligned} & -0.0004 \\ & : \\ & : \\ & -10.0000 \end{aligned}$ |  | $\begin{aligned} & \hline 3.9994 \\ & 1.1858 \end{aligned}$ |  | -1 <br> -4864 \|-27648 | $\begin{aligned} & \hline \text { FFFF } \\ & \text { ED00 } \\ & : \\ & 9400 \end{aligned}$ |
|  |  | -10.0004 <br> -11.7589 |  |  |  | -27649 <br> -32512 | $\begin{aligned} & 93 \text { FF } \\ & : \\ & 8100 \end{aligned}$ |
| Underflow | <-1.7593 | <-11.7589 | <0.0000 | <1.1858 |  | -32768 | 8000 |

Input ranges resistance temperature detector

| Range | $\begin{array}{\|l\|} \hline \mathrm{Pt} 100 / \mathrm{Pt} \\ 1000 \\ -50^{\circ} \mathrm{C} \ldots+70 \end{array}$ | $\begin{array}{\|l\|} \hline \mathrm{Pt} 100 \mathrm{I} \\ \mathrm{Pt} 1000 \\ { }^{\circ} \cdot 50^{\circ} \mathrm{C} \ldots+400 \end{array}$ | $\begin{aligned} & \mathrm{Ni} 1000 \\ & -50{ }^{\circ} \mathrm{C} \ldots+150 \\ & 0^{\circ} \mathrm{C} \end{aligned}$ | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  | Decimal | Hex. |
| Overflow | $>+80.0^{\circ} \mathrm{C}$ | $>+450.0^{\circ} \mathrm{C}$ | $>+160.0{ }^{\circ} \mathrm{C}$ | 32767 | 7FFF |
| Measured value too high |  | $\begin{aligned} & +450.0^{\circ} \mathrm{C} \\ & : \\ & +400.1^{\circ} \mathrm{C} \end{aligned}$ |  | $\begin{aligned} & 4500 \\ & : \\ & 4001 \end{aligned}$ | $\begin{aligned} & 1194 \\ & : \\ & \text { 0FA1 } \end{aligned}$ |
|  |  |  | $\begin{aligned} & +160.0^{\circ} \mathrm{C} \\ & : \\ & +150.1^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & 1600 \\ & : \\ & 1501 \end{aligned}$ | $\begin{aligned} & 0640 \\ & : \\ & \text { 05DD } \end{aligned}$ |
|  | $\begin{aligned} & +80.0^{\circ} \mathrm{C} \\ & : \\ & +70.1^{\circ} \mathrm{C} \end{aligned}$ |  |  | $\begin{aligned} & 800 \\ & : \\ & 701 \end{aligned}$ | $\begin{aligned} & 0320 \\ & : \\ & \text { 02BD } \end{aligned}$ |


| Range | $\begin{aligned} & \hline \mathrm{Pt100} / \mathrm{Pt} \\ & 1000 \\ & -50^{\circ} \mathrm{C} \ldots+70 \end{aligned}$ | $\begin{array}{\|l\|} \hline \mathrm{Pt} 100 \mathrm{I} \\ \mathrm{Pt} 1000 \\ \mathrm{P}^{\circ} \cdot 0^{\circ} \mathrm{C} \ldots+400 \end{array}$ | $\begin{aligned} & \text { Ni1000 } \\ & -50^{\circ} \mathrm{C} \ldots+150 \\ & 0^{\circ} \mathrm{C} \end{aligned}$ | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  | D6cimal | Hex. |
| Normal range |  | $+400.0{ }^{\circ} \mathrm{C}$ |  | 4000 | OFAO |
|  |  |  | $+150.0{ }^{\circ} \mathrm{C}$ | 1500 | 05DC |
|  | $+70.0{ }^{\circ} \mathrm{C}$ |  |  | 700 | 02BC |
|  |  |  |  |  |  |
|  | $+0.1{ }^{\circ} \mathrm{C}$ | $+0.1{ }^{\circ} \mathrm{C}$ | $+0.1{ }^{\circ} \mathrm{C}$ | 1 | 0001 |
|  | $0.0{ }^{\circ} \mathrm{C}$ | $0.0{ }^{\circ} \mathrm{C}$ | $0.0{ }^{\circ} \mathrm{C}$ | 0 | 0000 |
|  | $-0.1{ }^{\circ} \mathrm{C}$ | $-0.1{ }^{\circ} \mathrm{C}$ | $-0.1{ }^{\circ} \mathrm{C}$ | -1 | FFFF |
|  |  |  |  |  |  |
|  | $-50.0{ }^{\circ} \mathrm{C}$ | $-50.0{ }^{\circ} \mathrm{C}$ | $-50.0{ }^{\circ} \mathrm{C}$ | -500 | FEOC |
| Measured value too low | $-50.1{ }^{\circ} \mathrm{C}$ | $-50.1{ }^{\circ} \mathrm{C}$ | $-50.1{ }^{\circ} \mathrm{C}$ | -501 | FEOB |
|  |  |  |  |  |  |
|  | $-60.0{ }^{\circ} \mathrm{C}$ | $-60.0{ }^{\circ} \mathrm{C}$ | $-60.0{ }^{\circ} \mathrm{C}$ | -600 | FDA8 |
| Underflow | $<-60.0{ }^{\circ} \mathrm{C}$ | $<-60.0{ }^{\circ} \mathrm{C}$ | <-60.0 ${ }^{\circ} \mathrm{C}$ | -32768 | 8000 |

## Output ranges voltage and current

The represented resolution corresponds to 16 bits.

| Range | -10 V ...+10 V | $0 \mathrm{~mA} . . .20 \mathrm{mA4} \mathrm{~mA} . . .20 \mathrm{~mA}$ |  | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  | Decimal | Hex. |
| Overflow | 0 V | 0 mA | 0 mA | > 32511 | > 7EFF |
| Value too high | $\begin{aligned} & 11.7589 \mathrm{~V} \\ & : \\ & 10.0004 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 23.5178 \mathrm{~mA} \\ & : \\ & 20.0007 \mathrm{~mA} \\ & \hline \end{aligned}$ | $\begin{aligned} & 22.8142 \mathrm{~mA} \\ & : \\ & 20.0006 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 32511 \\ & : \\ & 27649 \end{aligned}$ | $\begin{aligned} & \text { 7EFF } \\ & : \\ & 6 \mathrm{C} 01 \end{aligned}$ |
| Normal range | $\begin{aligned} & 10.0000 \mathrm{~V} \\ & : \\ & 0.0004 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 20.0000 \mathrm{~mA} \\ & : \\ & 0.0007 \mathrm{~mA} \end{aligned}$ | $20.0000 \mathrm{~mA}$ $4.0006 \mathrm{~mA}$ | $27648$ | $\begin{aligned} & \text { 6C00 } \\ & : \\ & 0001 \end{aligned}$ |
|  | 0.0000 V | 0.0000 mA | 4.0000 mA | 0 | 0000 |
|  | $\begin{aligned} & -0.0004 \mathrm{~V} \\ & : \\ & -10.0000 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 0 \mathrm{~mA} \\ & : \\ & 0 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 3.9994 \mathrm{~mA} \\ & 0 \mathrm{~mA} \\ & 0 \mathrm{~mA} \end{aligned}$ | $\begin{array}{\|l\|} \hline-1 \\ -6912 \\ -27648 \\ \hline \end{array}$ | $\begin{aligned} & \text { FFFF } \\ & \text { E500 } \\ & 9400 \end{aligned}$ |
| Value too Iow | $\begin{aligned} & -10.0004 \mathrm{~V} \\ & : \\ & -11.7589 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 0 \mathrm{~mA} \\ & : \\ & 0 \mathrm{~mA} \end{aligned}$ | 0 mA <br> 0 mA | $\begin{aligned} & -27649 \\ & : \\ & -32512 \end{aligned}$ | $\begin{aligned} & 93 F F \\ & : \\ & 8100 \end{aligned}$ |
| Underflow | 0 V | 0 mA | 0 mA | <-32512 | <8100 |

## Technical data

## Technical data of the module

The system data of AC500 and S500 are applicable to the standard version ${ }^{*} \Rightarrow$ Chapter 5.1.2 "System data AC500" on page 166.

The system data of AC500-XC are applicable to the XC version $\stackrel{y}{ }{ }^{\circ}$ Chapter 5.1.3 "System data AC500-XC" on page 169.

Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter | Value |
| :---: | :---: |
| Process voltage |  |
| Connections | Terminals 1.8, 2.8, 3.8 and 4.8 for +24 V (UP) as well as 1.9, 2.9, 3.9 and 4.9 for 0 V (ZP) |
| Rated value | 24 V DC |
| Max. ripple | 5 \% |
| Protection against reversed voltage | Yes |
| Rated protection fuse on UP | 10 A fast |
| Galvanic isolation | Yes, per module |
| Current consumption |  |
| From 24 V DC power supply at the terminals UP/L+ and ZP/M of the CPU/communication interface module | Ca. 2 mA |
| From UP at normal operation | 0.15 A + output loads |
| Inrush current from UP (at power up) | $0.020 \mathrm{~A}^{2} \mathrm{~s}$ |
| Max. length of analog cables, conductor cross section > $0.14 \mathrm{~mm}^{2}$ | 100 m |
| Weight | 300 g |
| Mounting position | Horizontal or vertical with derating (output load reduced to $50 \%$ at $40^{\circ} \mathrm{C}$ per group) |
| Cooling | The natural convection cooling must not be hindered by cable ducts or other parts in the control cabinet. |

NOTICE!
All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and temporary overvoltage up to 30 V DC.

## Technical data of the analog inputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 4 |
| Distribution of channels into groups | 1 group of 4 channels |
| Connections of the channels I0- ... I3- | Terminals $1.0 \ldots 1.3$ |


| Parameter | Value |  |
| :---: | :---: | :---: |
| Connections of the channels 10+ ... 13+ | Terminals 2.0 ... 2.3 |  |
| Input type | Bipolar (not with current or Pt100/Pt1000/Ni1000) |  |
| Galvanic isolation | Against internal supply and other modules |  |
| Configurability | $0 \mathrm{~V} \ldots 10 \mathrm{~V},-10 \mathrm{~V} \ldots+10 \mathrm{~V}, 0 \mathrm{~mA} \ldots 20 \mathrm{~mA},$ <br> $4 \mathrm{~mA} . . .20 \mathrm{~mA}, \mathrm{Pt} 100 / 1000$, Ni1000 (each input can be configured individually) |  |
| Channel input resistance | Voltage: > 100 k $\Omega$ Current: ca. $330 \Omega$ |  |
| Time constant of the input filter | Voltage: $100 \mu \mathrm{~s}$ Current: $100 \mu \mathrm{~s}$ |  |
| Indication of the input signals | One LED per channel |  |
| Conversion cycle | 2 ms (for 8 inputs +8 outputs), with $\mathrm{Pt} / \mathrm{Ni}$... 1 s |  |
| Resolution | Range 0 V ... 10 V : 12 bits <br> Range -10 V ... +10 V : 12 bits including sign <br> Range $0 \mathrm{~mA} . . .20 \mathrm{~mA}$ : 12 bits <br> Range $4 \mathrm{~mA} . . .20 \mathrm{~mA}$ : 12 bits |  |
| Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within | Typ. | $\pm 0.5 \%$ of full scale at $25^{\circ} \mathrm{C}$ |
| the normal range | Max. | $\pm 1 \%$ of full scale (all ranges) at $0^{\circ} \mathrm{C} \ldots 60^{\circ} \mathrm{C}$ or EMC disturbance |
| Relationship between input signal and hex code | Chapter 5.2.4.4.2.5.10.2 "Input ranges resistance temperature detector" on page 804 |  |
| Unused voltage inputs | Are configured as "unused" |  |
| Unused current inputs | Have a low resistance, can be left open-circuited |  |
| Overvoltage protection | Yes |  |

## Technical data of the analog inputs, if used as digital inputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | Max. 4 |
| Distribution of channels into groups | 1 group of 4 channels |
| Connections of the channels I0+ to I3+ | Terminals 2.0 to 2.3 |
| Reference potential for the inputs | Terminals $1.9,2.9,3.9$ and 4.9 (ZP) |
| Input signal delay | Typ. 8 ms, configurable from 0.1 to 32 ms |
| Indication of the input signals | 1 LED per channel |
| Input signal voltage | 24 V DC |
|  | Signal 0 |
|  | Undefined signal |
|  | Signal 1 |
| Input current per channel | $+5 \mathrm{~V} \ldots+13 \mathrm{~V} \mathrm{~V}$ |
|  | Input voltage +24 V |


| Parameter |  | Value |
| :--- | :--- | :--- |
|  | Input voltage +5 V | Typ. 1.4 mA |
|  | Input voltage +15 V | Typ. 4.3 mA |
|  | Input voltage +30 V | $<9 \mathrm{~mA}$ |
| Input resistance | ca. $3.5 \mathrm{k} \Omega$ |  |

## Technical data of the analog outputs

| Parameter | Value |  |
| :---: | :---: | :---: |
| Number of channels per module | 4, all channels for voltage and current |  |
| Distribution of channels into groups | 1 group of 4 channels |  |
| Channels O0- ... O3- | Terminals 3.0 ... 3.3 |  |
| Channels O0+ ... O3+ | Terminals 4.0 ... 4.3 |  |
| Output type | Bipolar with voltage, unipolar with current |  |
| Galvanic isolation | Against internal supply and other modules |  |
| Configurability | -10 V ... +10 V, 0 mA ... $20 \mathrm{~mA}, 4 \mathrm{~mA} . . .20 \mathrm{~mA}$ (each output can be configured individually), current outputs only channels $0 \ldots 3$ |  |
| Output resistance (load), as current output | $0 \Omega \ldots 500 \Omega$ |  |
| Output loadability, as voltage output | Max. $\pm 10 \mathrm{~mA}$ |  |
| Indication of the output signals | One LED per channel |  |
| Resolution | 12 bits including sign |  |
| Settling time for full range change (resistive load, output signal within specified tolerance) | Typ. 5 ms |  |
| Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range | Typ. | $\pm 0.5 \%$ of full scale at $25^{\circ} \mathrm{C}$ |
|  | Max. | $\pm 1 \%$ of full scale (all ranges) <br> at $0^{\circ} \mathrm{C} . . .60^{\circ} \mathrm{C}$ or EMC disturbance |
| Relationship between output signal and hex code |  |  |
| Unused outputs | Can be left open-circuited |  |

## Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm

The dimensions are in mm and in brackets in inch.

Ordering Data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 250 100 R0001 | AX521, analog input/output module, <br> $4 \mathrm{AI}, 4 \mathrm{AO}, \mathrm{U} / / / \mathrm{Pt} 100$, <br> 12 bits including sign, 2-wires | Active |
| 1SAP 450 100 R0001 | AX521-XC, analog input/output <br> module, 4 AI, 4 AO, U/I/Pt100, <br> 12 bits including sign, 2-wires, <br> XC version | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

## AX522 - Analog input/output module

## Features

- 8 configurable analog inputs ( 10 ... 17) in 1 group (1.0 ... 2.7)

Resolution 12 bits including sign

- 8 configurable analog outputs ( $\mathrm{O} 0 \ldots \mathrm{O}$ ) in 1 group ( 3.0 ... 4.7)

Resolution 12 bits including sign

- Module-wise galvanically isolated
- XC version for use in extreme ambient conditions available


1 I/O bus
2 Allocation between terminal number and signal name
38 yellow LEDs to display the signal states at the analog inputs (I0 ... I7)
48 yellow LEDs to display the signal states at the analog outputs ( $\mathrm{O} 0 \ldots \mathrm{O}$ )
51 green LED to display the state of the process supply voltage UP
62 red LEDs to display errors
7 Label
8 Terminal unit
9 DIN rail
Sign for XC version

## Intended purpose

The device can be used as a decentralized I/O extension module for S 500 communication interface modules (e. g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs.

## Functionality

8 analog inputs (channel $0 \ldots$ channel 7 ), individually configurable

- Unused (default setting)
- 0 V ... 10 V
- $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$
- $0 \mathrm{~mA} . .20 \mathrm{~mA}$
- $4 \mathrm{~mA} . .20 \mathrm{~mA}$
- Pt100, $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ (2-wire)
- Pt100, $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ (3-wire), requires 2 channels
- Pt100, $-50^{\circ} \mathrm{C} . . .+70^{\circ} \mathrm{C}$ (2-wire)
- Pt100, $-50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ (3-wire), requires 2 channels
- Pt1000, $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ (2-wire)
- Pt1000, $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ (3-wire), requires 2 channels
- Ni1000, $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ (2-wire)
- Ni1000, $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ (3-wire), requires 2 channels
- 0 V ... 10 V with differential inputs, requires 2 channels
- $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ with differential inputs, requires 2 channels
- Digital signals (digital input)

| Parameter | Value |
| :--- | :--- |
| Resolution of the analog channels |  |
|  | Voltage $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ |
|  | Voltage $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ |
|  | Current $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$, <br> $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ |
| Temperature | 12 bits including sign |
| LED displays | 12 bits |
| Internal power supply | $0.1^{\circ} \mathrm{C}$ |
| External power supply | 19 LEDs for signals and error messages |
| Required terminal unit | Via the I/O bus interface (I/O bus) <br> 24 <br> Via V DC) |

## Connections

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

The modules are plugged on an I/O terminal unit ${ }^{\wedge}$ h Chapter 5.2.5.2 "TU515, TU516, TU541 and TU542 for I/O modules" on page 938. Properly position the modules and press until they lock in place. The terminal units are mounted on a DIN rail or with 2 screws plus the additional accessory for wall mounting $\stackrel{y y}{c}$ Chapter 5.2.8.2.5 "TA526-Wall mounting accessory" on page 1324.
The connection of the I/O channels is carried out using the 40 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

The terminals $1.8,2.8,3.8$ and 4.8 as well as $1.9,2.9,3.9$ and 4.9 are electrically interconnected within the I/O terminal units and always have the same assignment, independent of the inserted module:
Terminals 1.8, 2.8, 3.8 and 4.8: process voltage UP $=+24 \mathrm{~V} D$
Terminals 1.9, 2.9, 3.9 and 4.9: process voltage $\mathrm{ZP}=0 \mathrm{~V}$ DC
The assignment of the other terminals:

| Terminals | Signal | Description |
| :--- | :--- | :--- |
| $1.0 \ldots 1.7$ | $10-\ldots \mathrm{I} 7-$ | Negative poles of the 8 analog <br> inputs |
| $2.0 \ldots 2.7$ | $\mathrm{IO+} \ldots \mathrm{I7}+$ | Positive poles of the 8 analog <br> inputs |
| $3.0 \ldots 3.7$ | $\mathrm{O} 0-\ldots \mathrm{O}-$ | Negative poles of the 8 analog <br> outputs |
| $4.0 \ldots 4.7$ | $\mathrm{O} 0+\ldots \mathrm{O}++$ | Positive poles of the 8 analog <br> outputs |

The negative poles of the analog inputs are connected to each other to form an "Analog Ground" signal for the module.

The negative poles of the analog outputs are connected to each other to form an "Analog Ground" signal for the module.

There is no galvanic isolation between the analog circuitry and ZP/UP. Therefore, the analog sensors must be galvanically isolated in order to avoid loops via the ground potential or the supply voltage.

Because of their common reference potential, analog current inputs cannot be circuited in series, neither within the module nor with channels of other modules.

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a communication interface module or a CPU). Thus, the current consumption from 24 V DC power supply at the terminals L+/UP and M/ZP of the CPU/communication interface module increases by 2 mA per I/O module.

The external power supply connection is carried out via the UP (+24 V DC) and the ZP (0 V DC) terminals.

## WARNING!

## Removal/Insertion under power

Removal or insertion under power is permissible only if all conditions for hot swapping are fullfilled.

② Chapter 4.9.3 "Replace an I/O module with hot swap" on page 153
The devices are not designed for removal or insertion under power when the conditions for hot swap do not apply. Because of unforeseeable consequences, it is not allowed to plug in or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while they are energized in a hazardous location could result in an electric arc, which could create an ignition source resulting in fire or explosion.

Prior to proceeding, make sure that power is been disconnected and that the area has been thoroughly checked to ensure that flammable materials are not present.
The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

## Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

Generally, analog signals must be laid in shielded cables. The cable shields must be grounded at both sides of the cables. In order to avoid unacceptable potential differences between different parts of the installation, low resistance equipotential bonding conductors must be laid.


Fig. 128: Connection of the I/O module: 8 analog inputs and 8 analog outputs, individually configurable ${ }^{\mu}$ chapter 5.2.4.4.2.6.3 "Functionality" on page 811

## CAUTION!

By installing equipotential bonding conductors between the different parts of the system, it must be ensured that the potential difference between ZP and AGND never can exceed 1 V .

## CAUTION!

The process supply voltage must be included in the grounding concept (e. g. grounding of the negative terminal).

## Connection of resistance thermometers in 2-wire configuration

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the I/O module provides a constant current source which is multiplexed over the 8 analog channels.


Fig. 129: Connection example

| Pt100 | $-50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ | 2-wire configuration, one <br> channel used |
| :--- | :--- | :--- |
| $\operatorname{Pt100}$ | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 2-wire configuration, one <br> channel used |
| $\operatorname{Pt1000}$ | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 2-wire configuration, one <br> channel used |
| Ni1000 | $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ | 2-wire configuration, one <br> channel used |

The I/O module performs a linearization of the resistance characteristic.
In order to avoid error messages from unused analog input channels, it is useful to configure them as "unused".

## Connection of resistance thermometers in 3-wire configuration

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the I/O module provides a constant current source which is multiplexed over the max. 8 (depending on the configuration) analog channels.


Fig. 130: Connection example

If several measuring points are adjacent to each other, only one return line is necessary. This saves wiring costs.

With the 3-wire configuration, two adjacent analog channels belong together (e.g. the channels 0 and 1 ). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0 ), the next higher address must be the odd address (channel 1).
The constant current of one channel flows through the resistance thermometer. The constant current of the other channel flows through one of the cores. The module calculates the measured value from the two voltage drops and stores it under the input with the higher channel number (e.g. I1).
In order to keep measuring errors as small as possible, it is necessary to have all the involved conductors in the same cable. All the conductors must have the same cross section.

| $\operatorname{Pt100}$ | $-50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ | 3-wire configuration, two <br> channels used |
| :--- | :--- | :--- |
| $\mathrm{Pt100}$ | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 3-wire configuration, two <br> channels used |
| $\mathrm{Pt1000}$ | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 3-wire configuration, two <br> channels used |
| Ni1000 | $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ | 3-wire configuration, two <br> channels used |

The I/O module performs a linearization of the resistance characteristic.
In order to avoid error messages from unused analog input channels, it is useful to configure them as "unused".


Fig. 131: Connection example

By connecting the sensor's negative pole of the output voltage to AGND, the galvanically isolated voltage source of the sensor is referred to ZP.

The following measuring ranges can be configured for AX521 ${ }^{〔}$, Chapter 5.2.4.4.2.5.7 "Parameterization" on page 796 and for AX522 ${ }^{〔}$ Chapter 5.2.4.4.2.6.7 "Parameterization" on page 826:

| Voltage | $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ | 1 channel used |
| :--- | :--- | :--- |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | 1 channel used |

In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as "unused".

## Connection of active-type analog sensors (Current) with galvanically isolated power supply



Fig. 132: Connection example

| Current | $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 1 channel used |
| :--- | :--- | :--- |
| Current | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 1 channel used |

Unused input channels can be left open-circuited, because they are of low resistance.


Fig. 133: Connection example

## CAUTION!

The potential difference between AGND and ZP at the module must not be greater than 1 V , not even in case of long lines (see figure Terminal Assignment).

If AGND does not get connected to ZP, the sensor current flows to $Z P$ via the AGND line. The measuring signal is distorted, as a very small current flows through the voltage line. The total current through the PTC should not exceed 50 mA . This measuring method is therefore only suitable for short lines and small sensor currents. If there are bigger distances, the difference measuring method should be applied.

| Voltage | $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ | 1 channel used |
| :--- | :--- | :--- |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V} *)$ | 1 channel used |

*) if the sensor can provide this signal range
In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as "unused".

## Connection of passive-type analog sensors (Current)



Fig. 134: Connection example

| Current | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 1 channel used |
| :--- | :--- | :--- |

## CAUTION!

If, during initialization, an analog current sensor supplies more than 25 mA for more than 1 second to an analog input, this input is switched off by the module (input protection). In such cases, it is recommended to protect the analog input by a 10 -volt Zener diode (in parallel to I+ and ZP). But, in general, sensors with fast initialization or without current peaks higher than 25 mA are preferrable.

Unused input channels can be left open-circuited because they are of low resistance.

## Connection of active-type analog sensors (Voltage) to differential inputs

Differential inputs are very useful if analog sensors are used which are remotely non-isolated (e.g. the minus terminal is remotely grounded).

The use of differential inputs helps to considerably increase the measuring accuracy and to avoid ground loops.
With differential input configurations, two adjacent analog channels belong together (e.g. the channels 0 and 1 ). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0 ), the next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1).

The analog value is calculated by subtraction of the input value with the higher address from the input value of the lower address.

The converted analog value is available at the odd channel (higher address).

## CAUTION!

The ground potential at the sensors must not have too large a potential difference with respect to ZP (max. $\pm 1 \mathrm{~V}$ within the full signal range). Otherwise, problems may occur concerning the common-mode input voltages of the involved analog inputs.


Fig. 135: Connection example

> The negative pole of the sensor must be grounded next to the sensor.

| Voltage | $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ | with differential inputs, 2 chan- <br> nels used |
| :--- | :--- | :--- |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | with differential inputs, 2 chan- <br> nels used |

In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as "unused".

## Use of analog inputs as digital inputs

Several (or all) analog inputs can be configured as digital inputs. The inputs are not galvanically isolated against the other analog channels.


Fig. 136: Connection example

| Digital input | 24 V | 1 channel used |
| :--- | :--- | :--- |
| Effect of incorrect input ter- <br> minal connection |  | Wrong or no signal detected, <br> no damage up to 35 V |



Fig. 137: Connection example

| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | Load max. $\pm 10 \mathrm{~mA}$ | 1 channel used |
| :--- | :--- | :--- | :--- |
| Current | $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | Load $0 \Omega \ldots 500 \Omega$ | 1 channel used |
| Current | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | Load $0 \Omega \ldots 500 \Omega$ | 1 channel used |

Only the channels $0 \ldots 3$ can be configured as current output ( $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ or $4 \mathrm{~mA} . .20 \mathrm{~mA}$ ).
Unused analog outputs can be left open-circuited.

## Internal data exchange

| Digital inputs (bytes) | 0 |
| :--- | :--- |
| Digital outputs (bytes) | 0 |
| Counter input data (words) | 8 |
| Counter output data (words) | 8 |

## I/O configuration

The module does not store configuration data itself. It gets its parameterization data from the master device of the I/O bus (CPU or communication interface module) during power-up of the system.

Hence, replacing I/O modules is possible without any re-parameterization via software.

## Parameterization

| Firmware version | Configuration |
| :--- | :--- |
| Firmware version > V2.0.0 | The arrangement of the parameter data is per- <br> formed by Control Builder Plus/ Automation <br> Builder software. |

The parameter data directly influences the functionality of modules.
For non-standard applications, it is necessary to adapt the parameters to your system configuration.

Module: Module slot address: Y = 1 ... 10

For non-standard applications, it is necessary to adapt the parameters to your system configuration.

Module slot address: $\mathrm{Y}=1$... 7

| No. | Name | Value | Internal <br> value | Internal <br> value, <br> type | Default | Min. | Max. | EDS <br> Slot/ <br> Index |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| 1 | Module <br> ID | Internal | 1500 <br> $1)$ | Word | 1500 <br> $0 \times 05 \mathrm{dc}$ | 0 | 65535 | $0 \times 0 \mathrm{Y01}$ |
| 2 | Ignore <br> module <br> $2)$ | No <br> Yes | 0 <br> 1 | Byte | No <br> $0 \times 00$ |  |  | not for <br> FBP |


| No. | Name | Value | Internal value | Internal <br> value, type | Default | Min. | Max. | EDS <br> Slot/ Index |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 3 | Parameter length in bytes | Internal | 37 | Byte | $\begin{aligned} & 37-\mathrm{CPU} \\ & 37-\mathrm{FBP} \end{aligned}$ | 0 | 255 | 0x0Y02 |
| 4 | Check supply | $\begin{aligned} & \text { Off } \\ & \text { On } \end{aligned}$ | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | Byte | $\begin{aligned} & \text { On } \\ & 0 \times 01 \end{aligned}$ | 0 | 1 | 0x0Y03 |
| 5 | Analog data format | Default | 0 | Byte | Default <br> 0x00 |  |  | 0x0Y04 |
| 6 | Behaviour of outputs at com-munication errors | Off <br> Last value <br> Substitute value | $\begin{aligned} & 0 \\ & 1+\left(n^{*} 5\right) \\ & 2+\left(n^{*} 5\right), \\ & n \leq 2 \end{aligned}$ | Byte | $\begin{array}{\|l\|l\|l\|l\|l\|l} \text { Off } \\ 0 \times 00 \end{array}$ | 0 | 2 | 0x0Y05 |
| 7 | Channel configuration Input channel 0 | See table configura 4) Table nel config on page | 'Channel tion' <br> 87 "Chan uration ${ }^{2}$ )" 82 | Byte | Default <br> 0x00 | 0 | 19 | 0x0Y06 |
| 8 | Channel monitoring Input channel 0 | See table monitorin 4y Table nel moni on page | 'Channel 88 "Chan ring ${ }^{3}$ )" 830 | Byte | Default <br> 0x00 | 0 | 3 | 0x0Y07 |
| $\begin{aligned} & 9 \\ & \text { to } \\ & 22 \end{aligned}$ | Channel configu- <br> ration and channel monitoring of the input channels 1 ... 7 | See <br> table 'Ch figuration « Table nel config on page and table 'Ch monitorin 4y Table nel monit on page | annel con- <br> 187 "Chan uration ${ }^{2}$ )" 29 <br> nnel <br> 88 "Chan <br> oring ${ }^{3}$ )" <br> 830 | Byte <br> Byte | $\begin{aligned} & \text { Default } \\ & 0 \times 00 \\ & 0 \times 00 \end{aligned}$ | $\begin{aligned} & 0 \\ & 0 \end{aligned}$ | $\begin{aligned} & 19 \\ & 3 \end{aligned}$ | 0x0Y08 to 0x0Y15 |
| 23 | Channel configuration <br> Output channel 0 | See table configura ③ Table nel config on page | 'Channel tion' 187 "Chan uration 2)" 829 | Byte | Default <br> 0x00 | 0 | 130 | 0x0Y16 |


| No. | Name | Value | Internal value | Internal <br> value, type | Default | Min. | Max. | $\begin{array}{\|l\|} \hline \text { EDS } \\ \text { Slot/ } \\ \text { Index } \end{array}$ |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 24 | Channel monitoring Output channel 0 | See table 'Channel monitoring' ② Table 188 "Chan nel monitoring ${ }^{3}$ )" on page 830 |  | Byte | $\begin{array}{\|l\|} \hline \text { Default } \\ 0 \times 00 \end{array}$ | 0 | 3 | 0x0Y17 |
| 25 | Substi- <br> tute <br> value <br> Output <br> channel <br> 0 | only valid for output channel 0 | 0 ... 0xffff | Word | Default <br> 0x0000 | 0 | 65535 | 0x0Y18 |
| 26 to 31 | Channel configu- <br> ration and channel monitoring of the output channels 1 ... 3 | See <br> table 'Channel configuration' <br> « Table 187 "Chan nel configuration ${ }^{2}$ )" on page 829 <br> and <br> table 'Channel monitoring' Table 188 "Chan nel monitoring ${ }^{3}$ )" on page 830 |  | $\begin{array}{\|l\|l\|} \text { Byte } \\ \text { Byte } \end{array}$ | Default 0x00 0x00 | $0$ | $\begin{aligned} & 130 \\ & 3 \end{aligned}$ | 0x0Y19 to 0x0Y1E |
| 32 | Channel configuration <br> Output channel 4 | See table 'Channel configuration' <br> \& Table 187 "Chan nel configuration ${ }^{2}$ )" on page 829 |  | Byte | $\begin{array}{\|l\|} \hline \text { Default } \\ 0 \times 00 \end{array}$ | 0 | 128 | 0x0Y1F |
| 33 | Channel monitoring Output channel 4 | See table 'Channel monitoring' « Table 188 "Chan nel monitoring ${ }^{3}$ )" on page 830 |  | Byte | Default <br> 0x00 | 0 | 3 | 0x0Y20 |
| $\begin{aligned} & 34 \\ & \text { to } \\ & 39 \end{aligned}$ | Channel <br> configu- <br> ration <br> and <br> channel monitoring of the output channels 5 ... 7 | See <br> table 'Channel configuration' <br> ② Table 187 "Chan nel configuration ${ }^{2}$ )" on page 829 <br> and <br> table 'Channel monitoring' ③ Table 188 "Chan nel monitoring ${ }^{3}$ )" on page 830 |  | Byte <br> Byte | $\begin{array}{\|l} \text { Default } \\ 0 \times 00 \\ 0 \times 00 \end{array}$ | $0$ | $\begin{aligned} & 128 \\ & 3 \end{aligned}$ | $0 \times 0 Y 21$ <br> to <br> 0x0Y26 |

[^14]GSD file:

| Ext_User_Prm_Data_Len = | 24 |
| :---: | :---: |
| Ext_User_Prm_Data_Const(0) = | 0x05, 0xe2, 0x15, 1 |
|  | 0x01, 0x00, 0x00 |
|  | $\begin{aligned} & 0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00 \text {, } \\ & 0 \times 00,1 \end{aligned}$ |
|  | 0x00, 0x00, $0 \times 00,0 \times 00,1$ |
|  | 0x00, $0 \times 00,0 \times 00,0 \times 00,0 \times 00,0 \times 00$; |

Table 186: Input channel (4x)

| No. | Name | Internal value, type | Default |
| :--- | :--- | :--- | :--- |
| 1 | Channel configuration <br> see table $\left.{ }^{2}\right)$ | Byte | 0 |
| 2 | Channel monitoring $_{\left.\text {see table }{ }^{3}\right)}$ | Byte | $0 \times 00$ see table ${ }^{2}$ ) <br> sx00 see table $\left.{ }^{3}\right)$ |

Table 187: Channel configuration ${ }^{2}$ )

| Internal value | Operating modes of the analog inputs, individually configurable |
| :---: | :---: |
| 0 | Unused (default) |
| 1 | Analog input 0 V ... 10 V |
| 2 | Digital input |
| 3 | Analog input 0 mA ... 20 mA |
| 4 | Analog input 4 mA ... 20 mA |
| 5 | Analog input -10 V ... +10 V |
| 8 | Analog input Pt100, $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ (2-wire) |
| 9 | Analog input Pt100, $-50^{\circ} \mathrm{C} \ldots+400{ }^{\circ} \mathrm{C}$ (3-wire), requires 2 channels *) |
| 10 | Analog input $0 \ldots 10 \mathrm{~V}$ via differential inputs, requires 2 channels *) |
| 11 | Analog input -10 V ... +10 V via differential inputs, requires 2 channels *) |
| 14 | Analog input Pt100, $-50^{\circ} \mathrm{C} \ldots+70{ }^{\circ} \mathrm{C}$ (2-wire) |
| 15 | Analog input Pt100, $-50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ (3-wire), requires 2 channels *) |
| 16 | Analog input Pt1000, $-50^{\circ} \mathrm{C} \ldots+400{ }^{\circ} \mathrm{C}$ (2-wire) |
| 17 | Analog input Pt1000, $-50{ }^{\circ} \mathrm{C} \ldots+400{ }^{\circ} \mathrm{C}$ (3-wire), requires 2 channels *) |
| 18 | Analog input Ni1000, $-50^{\circ} \mathrm{C} \ldots+150{ }^{\circ} \mathrm{C}$ (2-wire) |
| 19 | Analog input Ni1000, $-50{ }^{\circ} \mathrm{C} \ldots+150{ }^{\circ} \mathrm{C}$ (3-wire), requires 2 channels *) |
|  | ${ }^{*}$ ) In the operating modes with 3-wire configuration or with differential inputs, two adjacent analog inputs belong together (e.g. the channels 0 and 1). In these cases, both channels are configured in the desired operating mode. The lower address must be the even address (channel 0). The next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1 ). |

Table 188: Channel monitoring ${ }^{3}$ )

| Internal value | Monitoring |
| :--- | :--- |
| 0 | Plausibility, open-circuit (broken wire) and short circuit |
| 3 | No monitoring |

Table 189: Output channel 0 (1 channel)

| No. | Name | Value | Internal value | Internal value, type | Default |
| :---: | :---: | :---: | :---: | :---: | :---: |
| 1 | Channel configuration | see table ${ }^{4}$ ) | see table ${ }^{4}$ ) | Byte | see table ${ }^{4}$ ) |
| 2 | Channel monitoring | see table ${ }^{5}$ ) | see table ${ }^{5}$ ) | Byte | see table ${ }^{5}$ ) |
| 3 | Substitute value see table ${ }^{6}$ ) | 0 ... 65535 | 0 ... <br> 0xffff | Word | 0 |

Table 190: Output channels 1 ... 3 (3x)

| No. | Name | Internal value, type |
| :--- | :--- | :--- |
| 1 | Channel configuration $_{\left.\text {see table }{ }^{4}\right)}$ | Byte |
| 2 | Channel monitoring $\left.^{\text {see table }}{ }^{6}\right)$ | Byte |

Table 191: Channel configuration ${ }^{4}$ )

| Internal value | Operating modes of the analog outputs, individually configurable |
| :--- | :--- |
| 0 | Unused (default) |
| 128 | Analog output $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ |
| 129 | Analog output $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ (not with the channels $4 \ldots 7$ and $12 \ldots$ 15) |
| 130 | Analog output $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ (not with the channels $4 \ldots 7$ and $12 \ldots$ 15) |

Table 192: Channel monitoring ${ }^{5}$ )

| Internal value | Monitoring |
| :--- | :--- |
| 0 | Plausibility, open circuit (broken wire) and short circuit (default) |
| 3 | No monitoring |

Table 193: Substitute value ${ }^{6}$ )

| Intended behaviour of <br> output channel when the <br> control system stops | Required setting of the <br> module parameter "Behav- <br> iour of outputs in case of a <br> communication error" | Required setting of the <br> channel parameter "Substi- <br> tute value" |
| :--- | :--- | :--- |
| Output OFF | Off | 0 |
| Last value infinite | Last value | 0 |


| Intended behaviour of <br> output channel when the <br> control system stops | Required setting of the <br> module parameter "Behav- <br> iour of outputs in case of a <br> communication error" | Required setting of the <br> channel parameter "Substi- <br> tute value" |
| :--- | :--- | :--- |
| Last value for 5 s and then <br> turn off | Last value 5 sec | 0 |
| Last value for 10 s and then <br> turn off | Last value 10 sec | 0 |
| Substitute value infinite | Substitute value | Depending on configuration |
| Substitute value for 5 s and <br> then turn off | Substitute value 5 sec | Depending on configuration |
| Substitute value for 10 s and <br> then turn off | Substitute value 10 sec | Depending on configuration |

Diagnosis
Table 194: Possible diagnosis of I/O channels

| Output range | Condition |  |
| :--- | :--- | :--- |
|  | Output value in the PLC <br> underflow | Output value in the PLC overflow |
| $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | Error identifier $=7$ | Error identifier $=4$ |
| $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ |  |  |
| $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ |  |  |


| Input range |  |  |  |  |
| :--- | :--- | :--- | :--- | :--- |
|  | Condition | Short circuit | Input value under- <br> flow | Input value over- <br> flow |
| $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | no diagnosis possible | no diagnosis possible | no diagnosis possible | Error identifier $=48$ |
| $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | Error identifier $=7$ | Error identifier $=7$ | Error identifier $=7$ | Error identifier $=48$ |
| $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | no diagnosis possible | Error identifier $=48$ | Error identifier $=7$ | Error identifier $=48$ |

Table 195: Content of diagnosis messages

| E1 ... E4 | d1 | d2 | d3 | d4 |  | Identifier 000 ... 063 | AC500 display | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch |  | Err | $\begin{aligned} & \text { PS501 } \\ & \text { PLC } \\ & \text { browser } \end{aligned}$ |  |  |
| Byte 6 <br> Bit 6 ... 7 | - | Byte 3 | Byte 4 | Byte 5 |  | Byte 6 <br> Bit 0 ... 5 | FBP diagnosis block |  |  |
| Class | Interface | Device | Module | Channel |  | Error Identifier | Error message |  | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | $\left.{ }^{3}\right)$ | ${ }^{4}$ ) |  |  |  |  |  |
| Module error |  |  |  |  |  |  |  |  |  |
| 3 | 14 | 1 ... 10 | 31 | 31 |  | 19 | Checksum error in the I/O module |  | $\begin{aligned} & \text { Replace } \\ & 1 / 0 \\ & \text { module } \end{aligned}$ |
|  | 11 / 12 | ADR | 1... 10 |  |  |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 |  | 3 | Timeout in the I/O module |  | $\begin{array}{\|l\|} \hline \text { Replace } \\ 1 / 0 \\ \text { module } \\ \hline \end{array}$ |
|  | 11 / 12 | ADR | 1... 10 |  |  |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 |  | 40 | Different hard-/firmware versions in the module |  | Replace I/O module |
|  | 11 / 12 | ADR | 1... 10 |  |  |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 |  | 43 | Internal error in the module |  | $\begin{aligned} & \text { Replace } \\ & 1 / 0 \\ & \text { module } \end{aligned}$ |
|  | 11 / 12 | ADR | 1... 10 |  |  |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 |  | 36 | Internal data exchange failure |  | Replace I/O module |
|  | 11 / 12 | ADR | 1... 10 |  |  |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 |  | 9 | Overflow diagnosis buffer |  | New start |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 |  | 26 | Parameter error |  | Check master |
|  | 11 / 12 | ADR | 1... 10 |  |  |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 |  | 11 | Process voltage too low |  | Check process voltage |
|  | 11 / 12 | ADR | 1... 10 |  |  |  |  |  |  |
| 4 | 14 | 1... 10 | 31 | 31 |  | 45 | Process voltage is switched off (ON -> OFF) |  | Process voltage ON |
|  | 11 / 12 | ADR | 1... 10 |  |  |  |  |  |  |
| Channel error |  |  |  |  |  |  |  |  |  |
|  |  |  |  | AX521 | AX522 |  |  |  |  |
| 4 | 14 | 1 ... 10 | 1 | $0 \ldots 3$ | 0 ... 7 | 48 | Analog value overflow or broken wire at an analog input |  | Check input value or terminal |
|  | 11 / 12 | ADR | 1... 10 |  |  |  |  |  |  |
| 4 | 14 | 1... 10 | 1 | 0... 3 | $0 . . .7$ | 7 | Analog value underflow at an analog input |  | Check input value |
|  | 11 / 12 | ADR | 1... 10 |  |  |  |  |  |  |
| 4 | 14 | 1... 10 | 1 | $0 \ldots 3$ | $0 \text {... } 7$ | 47 | Short circuit at an analog input |  | Check terminal |
|  | $11 / 12$ | ADR | $1 . .10$ |  |  |  |  |  |  |


| E1 ... E4 | d1 | d2 | d3 | d4 |  | Identifier $\begin{aligned} & 000 \ldots \\ & 063 \end{aligned}$ | AC500 display | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch |  | Err | $\begin{array}{\|l\|} \hline \text { PS501 } \\ \text { PLC } \\ \text { browser } \end{array}$ |  |  |
| Byte 6 <br> Bit 6 ... 7 | - | Byte 3 | Byte 4 | Byte 5 |  | Byte 6 <br> Bit 0 ... 5 | FBP diagnosis block |  |  |
| Class | Interface | Device | Module | Channel |  | Error Identifier | Error message |  | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | $\left.{ }^{3}\right)$ | ${ }^{4}$ ) |  |  |  |  |  |
| 4 | 14 | $1 \ldots 10$ | 3 | $4 \ldots 7$ | 8 ... 15 | 4 | Analog value overflow at an analog output |  | Check output value |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |  |  |  |  |
| 4 | 14 | $1 \ldots 10$ | 3 | $4 \ldots 7$ | $8 \ldots 15$ | 7 | Analog value underflow at an analog output |  | Check output value |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |  |  |  |  |

Remarks:

| ${ }^{1}$ ) | In AC500, the following interface identifier applies: $14=\mathrm{I} / \mathrm{O}$ bus, 11 = COM1 (e.g. CS31 bus), $12=$ COM2. <br> The FBP diagnosis block does not contain this identifier. |
| :---: | :---: |
| ${ }^{2}$ ) | With "Device" the following allocation applies: <br> 31 = module itself, $1 \ldots 10$ = expansion module $1 \ldots 10$, ADR = hardware address (e.g. of the DC551) |
| ${ }^{3}$ ) | With "Module" the following allocation applies depending on the master: <br> Module error: I/O bus or FBP: 31 = module itself; COM1/COM2: 1 ... $10=$ expansion 1 ... 10 <br> Channel error: I/O bus or FBP = module type ( $1=\mathrm{AI}, 3=\mathrm{AO}$ ); COM1/COM2: 1 ... 10 = expansion 1 ... 10 |
| ${ }^{4}$ ) | In case of module errors, with channel "31 = Module itself" is output. |

## State LEDs

During the power ON procedure, the module initializes automatically. All LEDs (except the channel LEDs) are ON during this time.


## Measuring ranges

## Input ranges of voltage, current and digital input

The represented resolution corresponds to 16 bits.

| Range | 0 V ... 10 V | $\begin{aligned} & \hline-10 \mathrm{~V} \ldots \\ & +10 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 0 \mathrm{~mA} . . . \\ & 20 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 4 \mathrm{~mA} \ldots \\ & 20 \mathrm{~mA} \end{aligned}$ | Digital input | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  | Decimal | Hex. |
| Overflow | >11.7589 | >11.7589 | >23.5178 | >22.8142 |  | 32767 | 7FFF |
| Measured value too high | $\begin{aligned} & 11.7589 \\ & \cdot \\ & \cdot \\ & 10.0004 \end{aligned}$ | $\begin{aligned} & 11.7589 \\ & : \\ & 10.0004 \end{aligned}$ | $\begin{aligned} & 23.5178 \\ & : \\ & 20.0007 \end{aligned}$ | $\begin{aligned} & 22.8142 \\ & : \\ & 20.0006 \end{aligned}$ |  | $\begin{aligned} & 32511 \\ & : \\ & 27649 \end{aligned}$ | $\begin{aligned} & \text { 7EFF } \\ & : \\ & 6 \mathrm{C} 01 \end{aligned}$ |
| Normal range | $\begin{aligned} & 10.0000 \\ & : \\ & 0.0004 \end{aligned}$ | $\begin{aligned} & 10.0000 \\ & : \\ & 0.0004 \end{aligned}$ | $\begin{aligned} & 20.0000 \\ & : \\ & 0.0007 \end{aligned}$ | $\begin{aligned} & 20.0000 \\ & : \\ & 4.0006 \end{aligned}$ | ON | $\begin{aligned} & 27648 \\ & : \\ & 1 \end{aligned}$ | $\begin{aligned} & \text { 6C00 } \\ & \text { : } \\ & 0001 \end{aligned}$ |
|  | 0.0000 | 0.0000 | 0 | 4 | OFF | 0 | 0000 |


| Range | 0 V ... 10 V | $\begin{aligned} & -10 \mathrm{~V} . . \\ & +10 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 0 \mathrm{~mA} \ldots \\ & 20 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 4 \mathrm{~mA} . . . \\ & 20 \mathrm{~mA} \end{aligned}$ | Digital input | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  | Decimal | Hex. |
| Normal range or measured value too low | $\begin{array}{r} -0.0004 \\ -1.7593 \end{array}$ | $\begin{aligned} & -0.0004 \\ & : \\ & : \\ & -10.0000 \end{aligned}$ |  | $\begin{aligned} & 3.9994 \\ & 1.1858 \end{aligned}$ |  | -1 -4864 \|-27648 | $\begin{aligned} & \text { FFFF } \\ & \text { ED00 } \\ & : \\ & 9400 \end{aligned}$ |
| Measured value too low |  | $\begin{array}{\|l} \hline-10.0004 \\ : \\ -11.7589 \end{array}$ |  |  |  | $\begin{aligned} & \hline-27649 \\ & : \\ & -32512 \end{aligned}$ | $\begin{aligned} & 93 F F \\ & : \\ & 8100 \end{aligned}$ |
| Underflow | <-1.7593 | <-11.7589 | <0.0000 | <1.1858 |  | -32768 | 8000 |

Input ranges resistance temperature detector

| Range | $\begin{array}{\|l\|} \hline \mathrm{Pt100} / \mathrm{Pt} \\ 1000 \\ -50^{\circ} \mathrm{C} \ldots+70 \end{array}$ | $\begin{array}{\|l\|} \hline \mathrm{Pt} 100 \mathrm{I} \\ \mathrm{Pt} 1000 \\ { }^{\circ} \cdot 50^{\circ} \mathrm{C} \ldots+400 \end{array}$ | $\begin{aligned} & \mathrm{Ni} 1000 \\ & -50^{\circ} \mathrm{C} \ldots+150 \\ & { }^{\circ} \mathrm{C} \end{aligned}$ | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  | D6cimal | Hex. |
| Overflow | $>+80.0{ }^{\circ} \mathrm{C}$ | $>+450.0^{\circ} \mathrm{C}$ | $>+160.0{ }^{\circ} \mathrm{C}$ | 32767 | 7FFF |
| Measured value too high |  | $\begin{aligned} & +450.0^{\circ} \mathrm{C} \\ & \vdots \\ & +400.1^{\circ} \mathrm{C} \end{aligned}$ |  | $\begin{aligned} & 4500 \\ & : \\ & 4001 \end{aligned}$ | 1194 <br> 0FA1 |
|  |  |  | $\begin{aligned} & +160.0^{\circ} \mathrm{C} \\ & : \\ & +150.1^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & 1600 \\ & : \\ & 1501 \end{aligned}$ | $\begin{aligned} & 0640 \\ & : \\ & 05 D D \end{aligned}$ |
|  | $\begin{aligned} & +80.0^{\circ} \mathrm{C} \\ & : \\ & +70.1^{\circ} \mathrm{C} \end{aligned}$ |  |  | $\begin{aligned} & 800 \\ & : \\ & 701 \end{aligned}$ | $\begin{aligned} & 0320 \\ & : \\ & \text { 02BD } \end{aligned}$ |
| Normal range | $\begin{aligned} & +70.0^{\circ} \mathrm{C} \\ & : \\ & +0.1^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & +400.0^{\circ} \mathrm{C} \\ & : \\ & : \\ & : \\ & +0.1^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & +150.0^{\circ} \mathrm{C} \\ & : \\ & : \\ & +0.1^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & 4000 \\ & 1500 \\ & 700 \\ & : \\ & 1 \end{aligned}$ | $\begin{aligned} & \hline \text { 0FA0 } \\ & \text { 05DC } \\ & \text { 02BC } \\ & : \\ & 0001 \end{aligned}$ |
|  | $0.0{ }^{\circ} \mathrm{C}$ | $0.0{ }^{\circ} \mathrm{C}$ | $0.0{ }^{\circ} \mathrm{C}$ | 0 | 0000 |
|  | $\begin{aligned} & -0.1^{\circ} \mathrm{C} \\ & : \\ & -50.0^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & -0.1^{\circ} \mathrm{C} \\ & : \\ & -50.0^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & -0.1^{\circ} \mathrm{C} \\ & : \\ & -50.0^{\circ} \mathrm{C} \end{aligned}$ | \|-1 -500 | $\begin{aligned} & \text { FFFF } \\ & : \\ & \text { FEOC } \end{aligned}$ |
| Measured value too low | $\begin{aligned} & -50.1^{\circ} \mathrm{C} \\ & : \\ & -60.0^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & -50.1^{\circ} \mathrm{C} \\ & : \\ & -60.0^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & -50.1^{\circ} \mathrm{C} \\ & : \\ & -60.0^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & -501 \\ & : \\ & -600 \end{aligned}$ | $\begin{aligned} & \text { FEOB } \\ & : \\ & \text { FDA8 } \end{aligned}$ |
| Underflow | $<-60.0^{\circ} \mathrm{C}$ | $<-60.0{ }^{\circ} \mathrm{C}$ | $<-60.0{ }^{\circ} \mathrm{C}$ | -32768 | 8000 |

## Output ranges voltage and current

The represented resolution corresponds to 16 bits.

| Range | -10 V ...+10 V | $0 \mathrm{~mA} \ldots 20 \mathrm{~mA} 4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ |  | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  | Decimal | Hex. |
| Overflow | 0 V | 0 mA | 0 mA | > 32511 | > 7EFF |
| Value too high | $\begin{aligned} & 11.7589 \mathrm{~V} \\ & : \\ & 10.0004 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 23.5178 \mathrm{~mA} \\ & : \\ & 20.0007 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 22.8142 \mathrm{~mA} \\ & : \\ & 20.0006 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 32511 \\ & : \\ & 27649 \end{aligned}$ | $\begin{aligned} & \text { 7EFF } \\ & : \\ & 6 \mathrm{C} 01 \end{aligned}$ |
| Normal range | $\begin{aligned} & 10.0000 \mathrm{~V} \\ & : \\ & 0.0004 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 20.0000 \mathrm{~mA} \\ & : \\ & 0.0007 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 20.0000 \mathrm{~mA} \\ & : \\ & 4.0006 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 27648 \\ & : \\ & 1 \end{aligned}$ | $\begin{aligned} & \hline 6 \mathrm{C} 00 \\ & : \\ & 0001 \end{aligned}$ |
|  | 0.0000 V | 0.0000 mA | 4.0000 mA | 0 | 0000 |
|  | $\begin{aligned} & -0.0004 \mathrm{~V} \\ & : \\ & -10.0000 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 0 \mathrm{~mA} \\ & : \\ & 0 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 3.9994 \mathrm{~mA} \\ & 0 \mathrm{~mA} \\ & 0 \mathrm{~mA} \end{aligned}$ | $\begin{array}{\|l\|} \hline-1 \\ -6912 \\ -27648 \end{array}$ | $\begin{aligned} & \text { FFFF } \\ & \text { E500 } \\ & 9400 \end{aligned}$ |
| Value too low | -10.0004 V -11.7589 V | $\begin{aligned} & 0 \mathrm{~mA} \\ & : \\ & 0 \mathrm{~mA} \end{aligned}$ | 0 mA <br> 0 mA | $\begin{aligned} & -27649 \\ & : \\ & -32512 \end{aligned}$ | $\begin{aligned} & \text { 93FF } \\ & : \\ & 8100 \end{aligned}$ |
| Underflow | 0 V | 0 mA | 0 mA | <-32512 | < 8100 |

## Technical data

## Technical data of the module

The system data of AC500 and S500 are applicable to the standard version Chapter 5.1.2 "System data AC500" on page 166.
The system data of AC500-XC are applicable to the XC version * Chapter 5.1.3 "System data AC500-XC" on page 169.
Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter |  | Value |
| :--- | :--- | :--- |
| Process voltage |  |  |
|  | Connections | Terminals 1.8, 2.8, 3.8 and 4.8 for +24 V <br> (UP) as well as 1.9, 2.9, 3.9 and 4.9 for 0 <br> V (ZP) |
|  | Rated value | 24 V DC |
|  | Max. ripple | $5 \%$ |
|  | Protection against reversed voltage | Yes |
|  | Rated protection fuse on UP | 10 A fast |
|  | Galvanic isolation | Yes, per module |
| Current consumption |  |  |


| Parameter | Value |
| :---: | :---: |
| From 24 V DC power supply at the terminals UP/L+ and ZP/M of the CPU/communication interface module | Ca. 2 mA |
| From UP at normal operation | 0.15 A + output loads |
| Inrush current from UP (at power up) | $0.020 \mathrm{~A}^{2} \mathrm{~s}$ |
| Max. length of analog cables, conductor cross section $>0.14 \mathrm{~mm}^{2}$ | 100 m |
| Weight | 300 g |
| Mounting position | Horizontal or vertical with derating (output load reduced to $50 \%$ at $40^{\circ} \mathrm{C}$ per group) |
| Cooling | The natural convection cooling must not be hindered by cable ducts or other parts in the control cabinet. |

## Technical data of the analog inputs

| Parameter | Value |  |
| :---: | :---: | :---: |
| Number of channels per module | 8 |  |
| Distribution of channels into groups | 1 group of 8 channels |  |
| Connections of the channels 10- ...17- | Terminals 1.0 ... 1.7 |  |
| Connections of the channels 10+ ... 17+ | Terminals 2.0 ... 2.3 |  |
| Input type | Bipolar (not with current or Pt100/Pt1000/Ni1000) |  |
| Galvanic isolation | Against internal supply and other modules |  |
| Configurability | $0 \vee \ldots 10 \mathrm{~V},-10 \mathrm{~V} \ldots+10 \mathrm{~V}, 0 \mathrm{~mA} . .22 \mathrm{~mA}$, 4 mA ... 20 mA, Pt100/1000, Ni1000 (each input can be configured individually) |  |
| Channel input resistance | Voltage: > $100 \mathrm{k} \Omega$ <br> Current: ca. $330 \Omega$ |  |
| Time constant of the input filter | Voltage: $100 \mu \mathrm{~s}$ current: $100 \mu \mathrm{~s}$ |  |
| Indication of the input signals | One LED per channel |  |
| Conversion cycle | 2 ms (for 8 inputs +8 outputs), with $\mathrm{Pt} / \mathrm{Ni} . . .1 \mathrm{~s}$ |  |
| Resolution | Range 0 V ... 10 V : 12 bits <br> Range -10 V ... +10 V: 12 bits including sign <br> Range $0 \mathrm{~mA} . . .20 \mathrm{~mA}: 12$ bits <br> Range $4 \mathrm{~mA} . . .20 \mathrm{~mA}$ : 12 bits |  |
| Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range | Typ. | $\begin{aligned} & \pm 0.5 \% \text { of full scale } \\ & \text { at } 25^{\circ} \mathrm{C} \end{aligned}$ |


| Parameter | Value |  |
| :--- | :--- | :--- |
|  | Max. | $\pm 1 \%$ of full scale (all ranges) <br> at $0^{\circ} \mathrm{C} \ldots 60^{\circ} \mathrm{C}$ or EMC disturbance |
| Unused voltage inputs | Are configured as "unused" |  |
| Unused current inputs | Have a low resistance, can be left open-circuited |  |
| Overvoltage protection | Yes |  |

## Technical data of the analog inputs, if used as digital Inputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | Max. 8 |
| Distribution of channels into groups | 1 group of 8 channels |
| Connections of the channels I0+ ... I7+ | Terminals $2.0 \ldots 2.7$ |
| Reference potential for the inputs | Terminals $1.9,2.9,3.9$ and $4.9(\mathrm{ZP})$ |
| Input signal delay | Typ. 8 ms, configurable from $0.1 \mathrm{~ms} \ldots 32 \mathrm{~ms}$ |
| Indication of the input signals | 1 LED per channel |
| Input signal voltage | 24 V DC |
|  | Signal 0 |
|  | Undefined signal |
|  | Signal 1 |
| Input current per channel | $+50 \mathrm{~V} \ldots+5 \mathrm{~V} . .+13 \mathrm{~V}$ |
|  | Input voltage +24 V |
|  | Input voltage +5 V |
|  | Input voltage +15 V |
|  | Input voltage +30 V |
| Input resistance | Typ. 7 mA |

## Technical data of the analog outputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 8, all channels for voltage, the first 4 channels <br> also for current |
| Distribution of channels into groups | 1 group of 8 channels |
|  | Channels O0- .. O7- |
|  | Terminals $3.0 \ldots 3.7$ |
| Output type | Terminals $4.0 \ldots 4.7$ |
| Galvanic isolation | Bipolar with voltage, unipolar with current |
| Configurability | Against internal supply and other modules |
|  | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}, 0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$, <br> $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}($ each output can be config- <br> ured individually), current outputs only chan- <br> nels $0 \ldots 3$ |
| Output resistance (load), as current output | $0 \Omega \ldots 500 \Omega$ |


| Parameter | Value |  |
| :--- | :--- | :---: |
| Output loadability, as voltage output | Max. $\pm 10 \mathrm{~mA}$ |  |
| Indication of the output signals | One LED per channel |  |
| Resolution | 12 bits including sign |  |
| Settling time for full range change (resistive <br> load, output signal within specified tolerance) | Typ. 5 ms |  |
| Conversion error of the analog values caused <br> by non-linearity, adjustment error at factory <br> and resolution within the normal range | Typ. |  |
|  | $\pm 0.5 \%$ of full scale <br> at $25^{\circ} \mathrm{C}$ |  |
|  | Max. |  |
| Relationship between output signal and hex full scale (all ranges) <br> code $0^{\circ} \mathrm{C} \ldots 60^{\circ} \mathrm{C}$ or EMC disturbance |  |  |
| Unused outputs |  |  |

## Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm

The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 250 000 R0001 | AX522, analog input/output module, <br> 8 AI, 8 AO, U/I/Pt100, <br> 12 bits including sign, 2-wires | Active |
| 1SAP 450 000 R0001 | AX522-XC, analog input/output <br> module, 8 AI, 8 AO, UI//Pt100, <br> 12 bits including sign, 2-wires, <br> XC version | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.4.5 Digital/Analog I/O modules

### 5.2.4.5.1 S500 and S500-XC

DA501 - Digital/Analog input/output module

## Features

- 16 digital inputs 24 V DC
- 8 configurable digital inputs/outputs 24 V DC, 0.5 A max.
- 4 analog inputs, voltage, current and RTD.

Resolution 12 bits including sign

- 2 analog outputs, voltage and current

Resolution 12 bits including sign

- Fast counter
- Module-wise galvanically isolated
- XC version for use in extreme ambient conditions available


1 I/O bus
2 Allocation between terminal number and signal name
316 yellow LEDs to display the signal states of the digital inputs DIO ... DI15
44 yellow LEDs to display the signal states of the analog inputs AIO ... AI3
2 yellow LEDs to display the signal states of the analog outputs AOO ... AO1
68 yellow LEDs to display the signal state of the configurable digital inputs/outputs DC16 ... DC23
71 green LED to display the state of the process supply voltage UP
4 red LEDs to display errors
Label
10 Terminal unit
11 DIN rail


Intended purpose
The device can be used as a decentralized I/O extension module for S500 communication interface modules (e. g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs.

## Functionality

- 16 digital inputs 24 V DC
- 8 configurable digital inputs/outputs 24 V DC, 0.5 A max.
- 4 analog inputs, voltage, current and RTD.

Resolution 12 bits including sign

- 2 analog outputs, voltage and current

Resolution 12 bits including sign

- Fast counter

| Parameter | Value |
| :--- | :--- |
| Fast Counter | Integrated, many configurable operating <br> modes |
| Power supply | From the process supply voltage UP |
| LED displays | For system displays, signal states, errors and <br> power supply |
| Internal supply voltage | Via the I/O bus interface (I/O bus) |
| External supply voltage | Via terminals UP and ZP (process supply <br> voltage 24 V DC) |
| Effect of incorrect input terminal connection | Wrong or no signal detected, no damage up to <br> 35 V |
| Required terminal unit | TU515 or TU516 « Chapter 5.2.5.2 "TU515, <br> TU516, TU541 and TU542 for I/O modules" <br> on page 938 |

## Connections

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

The connection is carried out by using the 40 terminals of the terminal unit TU515/TU516 « Chapter 5.2.5.2 "TU515, TU516, TU541 and TU542 for I/O modules" on page 938.
The assignment of the terminals:

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 1.0 | DI0 | Signal of the digital input DI0 |
| 1.1 | DI1 | Signal of the digital input DI1 |
| 1.2 | DI2 | Signal of the digital input DI2 |
| 1.3 | DI3 | Signal of the digital input DI3 |
| 1.4 | DI4 | Signal of the digital input DI4 |
| 1.5 | DI5 | Signal of the digital input DI5 |
| 1.6 | DI6 | Signal of the digital input DI6 |
| 1.7 | DI7 | Signal of the digital input DI7 |
| 1.8 | UP | Process voltage UP (24 V DC) |
| 1.9 | ZP | Process voltage ZP (0 V DC) |
| 2.0 | DI8 | Signal of the digital input DI8 |
| 2.1 | DI9 | Signal of the digital input DI9 |
| 2.2 | DI10 | Signal of the digital input DI10 |


| Terminal | Signal | Description |
| :---: | :---: | :---: |
| 2.3 | DI11 | Signal of the digital input DI11 |
| 2.4 | DI12 | Signal of the digital input DI12 |
| 2.5 | DI13 | Signal of the digital input DI13 |
| 2.6 | DI14 | Signal of the digital input DI14 |
| 2.7 | DI15 | Signal of the digital input DI15 |
| 2.8 | UP | Process voltage UP (24V DC) |
| 2.9 | ZP | Process voltage ZP (0 V DC) |
| 3.0 | Al0+ | Positive pole of analog input signal 0 |
| 3.1 | Al1+ | Positive pole of analog input signal 1 |
| 3.2 | Al2+ | Positive pole of analog input signal 2 |
| 3.3 | Al3+ | Positive pole of analog input signal 3 |
| 3.4 | Al- | Negative pole of analog input signals 0 to 3 |
| 3.5 | AO0+ | Positive pole of analog output signal 0 |
| 3.6 | AO1+ | Positive pole of analog output signal 1 |
| 3.7 | AO- | Negative pole of analog output signals 0 and 1 |
| 3.8 | UP | Process voltage UP (24V DC) |
| 3.9 | ZP | Process voltage ZP (0 V DC) |
| 4.0 | C16 | Signal of the configurable digital input/ output C16 |
| 4.1 | C17 | Signal of the configurable digital input/ output C17 |
| 4.2 | C18 | Signal of the configurable digital input/ output C18 |
| 4.3 | C19 | Signal of the configurable digital input/ output C19 |
| 4.4 | C20 | Signal of the configurable digital input/ output C20 |
| 4.5 | C21 | Signal of the configurable digital input/ output C21 |
| 4.6 | C22 | Signal of the configurable digital input/ output C22 |
| 4.7 | C23 | Signal of the configurable digital input/ output C23 |
| 4.8 | UP | Process voltage UP (24 V DC) |
| 4.9 | ZP | Process voltage ZP (0 V DC) |

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a communication interface module or a CPU). Thus, the current consumption from 24 V DC power supply at the terminals L+/UP and M/ZP of the CPU/communication interface module increases by 2 mA per DA501.
The external power supply connection is carried out via the UP (+24 V DC) and the ZP (0 V DC) terminals.

## WARNING!

## Removal/Insertion under power

Removal or insertion under power is permissible only if all conditions for hot swapping are fullfilled.
« Chapter 4.9.3 "Replace an I/O module with hot swap" on page 153
The devices are not designed for removal or insertion under power when the conditions for hot swap do not apply. Because of unforeseeable consequences, it is not allowed to plug in or unplug devices with the power being ON.
Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while they are energized in a hazardous location could result in an electric arc, which could create an ignition source resulting in fire or explosion.

Prior to proceeding, make sure that power is been disconnected and that the area has been thoroughly checked to ensure that flammable materials are not present.

The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

## Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.


## NOTICE!

Risk of damaging the PLC modules!
The PLC modules must not be removed while the plant is connected to a power supply.
Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove or replace a module.


## CAUTION!

## Risk of imprecise and faulty measurements!

Analog signals may be distorted seriously by external electromagnetic influences.

Use shielded wires when wiring analog signal sources. The cable shield must be grounded at both ends of the cable. Provide a potential equalization of a low resistance to avoid high potential differences between different parts of the plant.


Fig. 138: Terminal assignment of the module
The module provides several diagnosis functions ${ }^{«}$ Chapter 5.2.4.5.1.1.8 "Diagnosis" on page 861.

## Connection of the digital inputs

The following figure shows the connection of the digital input DIO. Proceed with the digital inputs DI1 ... DI15 in the same way.


Fig. 139: Connection of the module
The meaning of the LEDs is described in the Displays chapter $\Leftrightarrow$ Chapter 5.2.4.5.1.1.9 "State LEDs" on page 863.

## Connection of the configurable digital inputs/outputs



Fig. 140: Connection of configurable digital inputs/outputs to the module (DC16 ... DC23) (DC16 as an input, DC17 as an output)

## CAUTION!

## Risk of influences to the connected sensors!

Some sensors may be influenced by the deactivated module outputs of DA501.
If the inputs are used as fast counter inputs, connect a $470 \Omega / 1 \mathrm{~W}$ resistor in series to inputs DC16/DC17.

## Connection of resistance thermometers in 2-wire configuration to the analog inputs

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the module DA501 provides a constant current source which is multiplexed over the max. 4 analog input channels.


Fig. 141: Connection of resistance thermometers in 2-wire configuration to the analog inputs (AlO to Al3)

The following measuring ranges can be configured $\Leftrightarrow$ Chapter 5.2.4.5.1.1.7 "Parameterization" on page 856:

| Pt100 | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 2-wire configuration, 1 <br> channel used |
| :--- | :--- | :--- |
| $\mathrm{Pt1000}$ | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 2-wire configuration, 1 <br> channel used |
| Ni 1000 | $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ | 2-wire configuration, 1 <br> channel used |

The function of the LEDs is described under Diagnosis and displays / Displays $\Leftrightarrow$ Chapter 5.2.4.5.1.1.9 "State LEDs" on page 863.

The module DA501 performs a linearization of the resistance characteristic.
To avoid error messages from unused analog input channels, configure them as "unused".

## Connection of resistance thermometers in 3-wire configuration to the analog inputs

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the module DA501 provides a constant current source which is multiplexed over the max. 4 analog input channels.


Fig. 142: Connection of resistance thermometers in 3-wire configuration to the analog inputs (AIO ... Al3)

With 3-wire configuration, 2 adjacent analog channels belong together (e. g. the channels 0 and 1). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0), the next higher address must be the odd address (channel 1).
The constant current of one channel flows through the resistance thermometer. The constant current of the other channel flows through one of the cores. The module calculates the measured value from the two voltage drops and stores it under the input with the higher channel number (e. g. I1).

In order to keep measuring errors as small as possible, it is necessary to have all the involved conductors in the same cable. All the conductors must have the same cross section.
The following measuring ranges can be configured $\Leftrightarrow$ Chapter 5.2.4.5.1.1.7 "Parameterization" on page 856:

| $\operatorname{Pt100}$ | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 3-wire configuration, 2 chan- <br> nels used |
| :--- | :--- | :--- |
| $\operatorname{Pt1000}$ | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 3-wire configuration, 2 chan- <br> nels used |
| Ni 1000 | $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ | 3-wire configuration, 2 chan- <br> nels used |

The function of the LEDs is described under Diagnosis and displays / Displays ${ }^{\mu}$ Chapter 5.2.4.5.1.1.8 "Diagnosis" on page 861.

0
The module DA501 performs a linearization of the resistance characteristic.
To avoid error messages from unused analog input channels, configure them as "unused".

## Connection of active-type analog sensors (Voltage) with galvanically isolated power supply to the analog inputs



Fig. 143: Connection of active-type analog sensors (voltage) with galvanically isolated power supply to the analog inputs (AIO ... AI3)
The following measuring ranges can be configured $\Longleftrightarrow$ Chapter 5.2.4.5.1.1.7 "Parameterization" on page 856 :

| Voltage | $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ | 1 channel used |
| :--- | :--- | :--- |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | 1 channel used |

The function of the LEDs is described under Diagnosis and displays / Displays ${ }_{\mu}{ }^{\circ}$ Chapter 5.2.4.5.1.1.9 "State LEDs" on page 863.

To avoid error messages from unused analog input channels, configure them as "unused".

## Connection of active-type analog sensors (Current) with galvanically isolated power supply to the analog inputs



Fig. 144: Connection of active-type analog sensors (current) with galvanically isolated power supply to the analog inputs (AIO ... AI3)
The following measuring ranges can be configured $\Rightarrow$ Chapter 5.2.4.5.1.1.7 "Parameterization" on page 856:

| Current | $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 1 channel used |
| :--- | :--- | :--- |
| Current | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 1 channel used |

The function of the LEDs is described under Diagnosis and displays / Displays $\stackrel{\leftrightarrow}{~}$ Chapter 5.2.4.5.1.1.9 "State LEDs" on page 863.

Unused input channels can be left open-circuited, because they are of low resistance.

## Connection of active-type analog sensors (Voltage) with no galvanically isolated power supply to the analog inputs



Fig. 145: Connection of active-type sensors (voltage) with no galvanically isolated power supply to the analog inputs (AIO ... AI3)

## CAUTION!

Risk of faulty measurements!
The negative pole at the sensors must not have too big a potential difference with respect to ZP (max. $\pm 1 \mathrm{~V}$ within the full signal range).
Make sure that the potential difference never exceeds $\pm 1 \mathrm{~V}$.

The following measuring ranges can be configured $\stackrel{y}{ }{ }^{\circ}$ Chapter 5.2.4.5.1.1.7 "Parameterization" on page 856 :

| Voltage | $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ | 1 channel used |
| :--- | :--- | :--- |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | 1 channel used |

For a description of the function of the LEDs, please refer to the Diagnosis and displays / Displays chapter ${ }^{\wedge}$ Chapter 5.2.4.5.1.1.9 "State LEDs" on page 863.
To avoid error messages from unused analog input channels, configure them as "unused".

## Connection of passive-type analog sensors (Current) to the analog inputs



Fig. 146: Connection of passive-type analog sensors (current) to the analog inputs (AIO to AI3) The following measuring ranges can be configured ${ }^{\Leftrightarrow}$ Chapter 5.2.4.5.1.1.7 "Parameterization" on page 856:

| Current | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 1 channel used |
| :--- | :--- | :--- |

For a description of function of the LEDs, please refer to the Diagnosis and displays / Displays chapter ${ }^{〔}$ Chapter 5.2.4.5.1.1.9 "State LEDs" on page 863.

## CAUTION!

## Risk of overloading the analog input!

If an analog current sensor supplies more than 25 mA for more than 1 second during initialization, this input is switched off by the module (input protection).

Only use sensors with fast initialization or without current peaks higher than 25 mA . If not possible, connect a 10 -volt Zener diode in parallel to I+ and ZP.

Unused input channels can be left open-circuited, because they are of low resistance.

## Connection of active-type analog sensors (Voltage) to differential analog inputs

Differential inputs are very useful if analog sensors which are remotely non-isolated (e.g. the negative terminal is remotely grounded) are used.

Using differential inputs helps to considerably increase the measuring accuracy and to avoid ground loops.
With differential input configurations, two adjacent analog channels belong together (e.g. the channels 0 and 1 ). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0), the next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1).
The analog value is calculated by subtraction of the input value with the higher address from the input value of the lower address.

The converted analog value is available at the odd channel (higher address).

## CAUTION!

## Risk of faulty measurements!

The negative pole at the sensors must not have too large a potential difference with respect to ZP (max. $\pm 1 \mathrm{~V}$ within the full signal range).
Make sure that the potential difference never exceeds $\pm 1 \mathrm{~V}$.

The following figure shows the connection of active-type analog sensors (voltage) to differential analog inputs AIO and AI1. Proceed with AI2 and AI3 in the same way.


Fig. 147: Connection of active-type analog sensors (voltage) to differential analog inputs (AIO ... Al3)
The following measuring ranges can be configured ${ }^{4}$ Chapter 5.2.4.5.1.1.7 "Parameterization" on page 856:

| Voltage | $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ | with differential inputs, 2 chan- <br> nels used |
| :--- | :--- | :--- |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | with differential inputs, 2 chan- <br> nels used |

For a description of the function of the LEDs, please refer to the Diagnosis and displays / Displays chapter ${ }^{\Leftrightarrow} \Rightarrow$ Chapter 5.2.4.5.1.1.9 "State LEDs" on page 863.
To avoid error messages from unused analog input channels, configure them as "unused".

## Use of analog inputs as digital inputs

Several (or all) analog inputs can be configured as digital inputs. The inputs are not galvanically isolated against the other analog channels.


Fig. 148: connection of digital sensors to the analog inputs (AIO ... AI3)
The following measuring ranges can be configured $\Leftrightarrow$ Chapter 5.2.4.5.1.1.7 "Parameterization" on page 856:

| Digital input | 24 V | 1 channel used |
| :--- | :--- | :--- |

For a description of the function of the LEDs, please refer to the Diagnosis and displays / Displays chapter ${ }^{\mu}$ Chapter 5 .2.4.5.1.1.9 "State LEDs" on page 863.

## Connection of analog output loads (Voltage)



Fig. 149: Connection of analog output loads (voltage) to the analog outputs (AOO and AO1)

The following measuring ranges can be configured ${ }^{\Perp}$ Chapter 5.2.4.5.1.1.7 "Parameterization" on page 856 :

| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | Load $\pm 10 \mathrm{~mA}$ max. | 1 channel used |
| :--- | :--- | :--- | :--- |

For a description of the function of the LEDs, please refer to the Diagnosis and displays / Displays chapter ${ }^{\text {es }}$ Chapter 5.2.4.5.1.1.9 "State LEDs" on page 863.
Unused analog outputs can be left open-circuited.

## Connection of analog output loads (Current)



Fig. 150: Connection of analog output loads (current) to the analog outputs (AOO and AO1)
The following measuring ranges can be configured $«$ Chapter 5.2.4.5.1.1.7 "Parameterization" on page 856:
0

| Current | $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | Load $0 \Omega \ldots 500 \Omega$ | 1 channel used |
| :--- | :--- | :--- | :--- |
| Current | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | $\operatorname{Load} 0 \Omega \ldots 50 \Omega$ | 1 channel used |

For a description of the function of the LEDs, please refer to the Diagnosis and displays / Displays chapter ${ }^{4}$ Chapter 5.2.4.5.1.1.9 "State LEDs" on page 863.
Unused analog outputs can be left open-circuited.

## Internal data exchange

|  | Without the fast counter | With the fast counter (only <br> with AC500) |
| :--- | :--- | :--- |
| Digital inputs (bytes) | 3 | 5 |
| Digital outputs (bytes) | 1 | 3 |
| Analog inputs (words) | 4 | 4 |


|  | Without the fast counter | With the fast counter (only <br> with AC500) |
| :--- | :--- | :--- |
| Digital outputs (words) | 2 | 2 |
| Counter input data (words) | 0 | 4 |
| Counter output data (words) | 0 | 8 |

## I/O configuration

The module does not store configuration data itself. It gets its parameterization data from the master device of the I/O bus (CPU or communication interface module) during power-up of the system.
Hence, replacing I/O modules is possible without any re-parameterization via software.

## Parameterization

| Firmware version | Configuration |
| :--- | :--- |
| Firmware version $>\mathrm{V} 2.0 .0$ | The arrangement of the parameter data is per- <br> formed by Control Builder Plus/ Automation <br> Builder software. |

The parameter data directly influences the functionality of modules.
For non-standard applications, it is necessary to adapt the parameters to your system configuration.

Module: Module slot address: $\mathrm{Y}=1$... 10

| Name | Value | Internal value | Internal <br> value, type | Default | EDS Slot $/$ <br> Index |
| :--- | :--- | :--- | :--- | :--- | :--- |
| Module ID <br> ${ }^{1}$ ) | Internal | 1810 | WORD | 1810 | $0 x 0$ Y01 |
| Ignore module <br> see table $\left.{ }^{2}\right)$ | Internal | Yes <br> No | BYTE | No | not for FBP |
| Parameter <br> length | Internal | 8 | BYTE | 8 | $0 x$ Y02 |
| Check supply | off | 0 | BYTE | 1 | $0 x Y 03$ |
|  | on | 1 |  |  |  |

$\left.\begin{array}{|l|l|l|l|l|l|}\hline \text { Name } & \text { Value } & \text { Internal value } & \begin{array}{l}\text { Internal } \\ \text { value, type }\end{array} & \text { Default } & \begin{array}{l}\text { EDS Slot / } \\ \text { Index }\end{array} \\ \hline \begin{array}{l}\text { Fast counter } \\ { }^{3} \text { ) }\end{array} & 0 & 0 & \text { BYTE } & 0 & \text { not for FBP } \\ : & 10 \\ 4)\end{array}\right)$

| ${ }^{2}$ ) | Setting | Description |
| :--- | :--- | :--- |
|  | On | Error LED lights up at errors of all error classes, Failsafe <br> mode off |
|  | Off by E4 | Error LED lights up at errors of error classes E1, E2 and E3, <br> Failsafe mode off |
|  | Off by E3 | Error LED lights up at errors of error classes E1 and E2, <br> Failsafe mode off |
|  | Off by E4 + Failsafe | Error LED lights up at errors of error classes E1, E2 and E3, <br> mode on *) <br> Failsafe mode on *) |
|  | Off by E3 + Failsafe | Error LED lights up at errors of error classes E1 and E2, <br> Failsafe mode on *) |

## Remarks:

${ }^{1}$ ) With a faulty ID, the Modules reports a "parameter error" and does not perform cyclic process data transmission
${ }^{2}$ ) Not for FBP
${ }^{3}$ ) With FBP or CS31 without the parameter "Fast Counter"

The fast counter of the module does not work if the module is connected to an FBP interface module or CS31 bus module.
${ }^{4}$ ) For counter operating modes, please refer to the description of the fast counter $\Leftrightarrow$ Chapter 5.2.4.3.2.9 "Fast counter" on page 606
${ }^{5}$ ) The parameter Behavior outputs at comm. error is only analyzed if the Failsafe-mode is ON.

## Group parameters for the digital part

| Name | Value | Internal value | Internal <br> value, type | Default | EDS Slot $/$ <br> Index |
| :--- | :--- | :--- | :--- | :--- | :--- |
| Input delay | 0.1 ms <br> 1 ms <br> 8 ms <br> 32 ms | 0 | 1 | BYTE | 0.1 ms <br> 0 <br> 3 |
| Detect short <br> circuit at out- <br> puts | Off <br> On | 0 | $0 x 0 \mathrm{Y} 05$ |  |  |
| Substitute <br> value at <br> output | $0 \ldots 255$ | $00 \mathrm{~h} \ldots$ FFh | BYTE | 0 <br> $0 x 0000$ | BYTE |

${ }^{*}$ ) The parameters Behavior DO at comm. error is only analyzed if the Failsafe mode is ON.

Group parameters for the analog part

| Name | Value | Internal value | Internal <br> value, type | Default | EDS Slot $/$ <br> Index |
| :--- | :--- | :--- | :--- | :--- | :--- |
| Analog data <br> format | Standard <br> Reserved | 255 <br> RYTE | 0 | $0 \times 0 \mathrm{Y04}$ |  |

${ }^{*}$ ) The parameter Behavior AO at comm. error is only analyzed if the Failsafe mode is ON.

Channel parameters for the analog inputs (4x)

| Name | Value | Internal value | Internal value, type | Default | EDS Slot / Index |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Input 0, Channel configuration | see table 'Channel configuration' Table 196 " Channel configuration" on page 859 | see table 'Channel configuration' ※ Table 196 " Channel configuration" on page 859 | BYTE | 0 | 0x0Y09 |
| Input 0, Check channel | see table 'Channel monitoring' ② Table 197" Channel monitoring" on page 859 | see table 'Channel monitoring' » Table 197" Channel monitoring" on page 859 | BYTE | 0 | 0xOYOA |
| : | : | : | : | : |  |
| : | : | : | : | : |  |


| Name | Value | Internal value | Internal value, type | Default | EDS Slot / Index |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Input 3, Channel configuration | see table 'Channel configuration' Table 196 " Channel configuration" on page 859 | see table 'Channel configuration' *) Table 196 " Channel configuration" on page 859 | BYTE | 0 | 0x0Y0F |
| Input 3, Check channel | see table 'Channel monitoring' Table 197" Channel monitoring" on page 859 | see table 'Channel monitoring' Table 197" Channel monitoring" on page 859 | BYTE | 0 | 0x0Y10 |

Table 196: Channel configuration

| Internal value | Operating modes of the analog inputs, individually configurable |
| :---: | :---: |
| 0 (default) | Not used |
| 1 | 0 V ... 10 V |
| 2 | Digital input |
| 3 | 0 mA ... 20 mA |
| 4 | $4 \mathrm{~mA} . . .20 \mathrm{~mA}$ |
| 5 | -10 V ... +10 V |
| 8 | 2-wire Pt100-50 ${ }^{\circ} \mathrm{C} \ldots+400{ }^{\circ} \mathrm{C}$ |
| 9 | 3-wire Pt100-50 ${ }^{\circ} \mathrm{C} \ldots+40{ }^{\circ} \mathrm{C}$ *) |
| 10 | 0 V ... 10 V (voltage diff.) *) |
| 11 | -10 V ... +10 V (voltage diff.) *) |
| 14 | 2-wire Pt100-50 ${ }^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ |
| 15 | 3-wire Pt100 -50 ${ }^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ *) |
| 16 | 2-wire Pt1000-50 ${ }^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ |
| 17 | 3-wire Pt1000-50 ${ }^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ *) |
| 18 | 2-wire Ni1000-50 ${ }^{\circ} \mathrm{C} \ldots+150{ }^{\circ} \mathrm{C}$ |
| 19 | 3-wire Ni1000-50 ${ }^{\circ} \mathrm{C} \ldots+150{ }^{\circ} \mathrm{C}{ }^{*}$ ) |
|  | *) In the operating modes with 3-wire configuration or with differential inputs, two adjacent analog inputs belong together (e.g. the channels 0 and 1). In these cases, both channels are configured in the desired operating mode. The lower address must be the even address (channel 0). The next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1). |

Table 197: Channel monitoring

| Internal Value | Check Channel |
| :--- | :--- |
| 0 (default) | Plausibility, wire break, short circuit |
| 3 | Not used |

## Channel parameters for the analog outputs (2x)

| Name | Value | Internal value | Internal value, type | Default | $\begin{array}{\|l} \text { EDS Slot / } \\ \text { Index } \end{array}$ |
| :---: | :---: | :---: | :---: | :---: | :---: |
| 0 <br> Output 0, Channel configuration | see table 'Channel configuration' Table 198" Channel configuration" on page 860 | see table 'Channel configuration' (3) Table 198" Channel configuration" on page 860 | BYTE | 0 | 0x0Y11 |
| Output 0, Check channel | see table 'Channel monitoring' « Table 199" Channel monitoring" on page 861 | see table 'Channel monitoring' ( ) Table 199" Channel monitoring" on page 861 | BYTE | 0 | 0x0Y12 |
| Output 0 , Substitute value | see table <br> 'Substitute value' <br> © Table 200" <br> Substitute value" on page 861 | see table <br> 'Substitute value' <br> を Table 200 " <br> Substitute value" on page 861 | WORD | 0 | 0x0Y13 |
| Output 1, Channel configuration | see table 'Channel configuration' Table 198" Channel configuration" on page 860 | see table 'Channel configuration' (3) Table 198" Channel configuration" on page 860 | BYTE | 0 | 0x0Y14 |
| Output 1, Check channel | see table 'Channel monitoring' ③ Table 199" Channel monitoring" on page 861 | see table 'Channel monitoring' ② Table 199" Channel monitoring" on page 861 | BYTE | 0 | 0x0Y15 |
| Output 1, Substitute value | see table <br> 'Substitute value' <br> © Table 200" <br> Substitute value" on page 861 | see table 'Substitute value' を Table 200 " Substitute value" on page 861 | WORD | 0 | 0x0Y16 |

Table 198: Channel configuration

| Internal value | Operating modes of the analog outputs, individually configurable |
| :--- | :--- |
| 0 (default) | Not used |
| 128 | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ |
| 129 | $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ |
| 130 | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ |

Table 199: Channel monitoring

| Internal value | Check channel |
| :--- | :--- |
| 0 | Plausibility, wire break, short circuit |
| 3 | None |

Table 200: Substitute value

| Intended behavior of output <br> channel when the control <br> system stops | Required setting of <br> the module parameter <br> "Behavior of outputs in <br> case of a communication <br> error" | Required setting of the <br> channel parameter "Substi- <br> tute value" |
| :--- | :--- | :--- |
| Output OFF | Off | 0 |
| Last value infinite | Last value | 0 |
| Last value for 5 s and then <br> turn off | Last value 5 sec | 0 |
| Last value for 10 s and then <br> turn off | Last value 10 sec | Depending on configuration |
| Substitute value infinite | Substitute value | Depending on configuration |
| Substitute value for 5 s and <br> then turn off | Substitute value 5 sec | Depending on configuration |
| Substitute value for 10 s and <br> then turn off | Substitute value 10 sec |  |

## Diagnosis

In cases of short circuit or overload, the digital outputs are turned off. The module performs reactivation automatically. Thus an acknowledgement of the errors is not necessary. The error message is stored via the LED.

| E1 ... E4 | d1 | d2 | d3 | d4 | $\begin{array}{\|l\|} \hline \text { Identifier } \\ 000 \ldots 063 \end{array}$ | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err |  |  |
| Byte 6 <br> Bit 6 ... 7 | - | Byte 3 | Byte 4 | Byte 5 | Byte 6 <br> Bit 0 ... 5 |  |  |
| Class | Interface | Device | Module | Channel | Error Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | $\left.{ }^{3}\right)$ | ${ }^{4}$ ) |  |  |  |
| Module error |  |  |  |  |  |  |  |
| 0 | 14 | $1 . .10$ | 31 | 31 | 19 | Checksum error in the I/O module | Replace I/O module |
| 3 | 11 / 12 | ADR | 1... 10 |  |  |  |  |
| 3 | 14 | $1 \ldots 10$ | 31 | 31 | 3 | Timeout in the I/O module |  |
|  | $11 / 12$ | ADR | 1... 10 |  |  |  |  |
| 3 | 14 | $1 \ldots 10$ | 31 | 31 | 40 | Different hard-/firmware versions in the module |  |
|  | 11 / 12 | ADR | 1... 10 |  |  |  |  |


| E1 ... E4 | d1 | d2 | d3 | d4 | $\begin{aligned} & \text { Identifier } \\ & 000 \ldots 063 \end{aligned}$ | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err |  |  |
| Byte 6 <br> Bit 6 ... 7 | - | Byte 3 | Byte 4 | Byte 5 | Byte 6 <br> Bit 0 ... 5 |  |  |
| Class | Interface | Device | Module | Channel | Error Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | $\left.{ }^{2}\right)$ | $\left.{ }^{3}\right)$ | $\left.{ }^{4}\right)$ |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 43 | Internal error in the module |  |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 36 | Internal data exchange failure |  |
|  | $11 / 12$ | ADR | 1... 10 |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 9 | Overflow diagnosis buffer | New start |
|  | $11 / 12$ | ADR | 1... 10 |  |  |  |  |
| 3 | 14 | $1 . .10$ | 31 | 31 | 26 | Parameter error | Check master |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 11 | Process voltage too low | Check process voltage |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |
| 4 | 14 | 1... 10 | 31 | 31 | 45 | Process voltage is switched off (ON -> OFF) | Process voltage ON |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |
| Channel error DA501 |  |  |  |  |  |  |  |
| 4 | 14 | 1... 10 | 2 | $\left.\begin{array}{c} 22 \ldots 29 \\ 5 \end{array}\right)$ | 47 | Short circuit at a digital output | Check connection |
|  | 11 / 12 | ADR | 1... 10 |  |  |  |  |
| Channel error DA501 |  |  |  |  |  |  |  |
| 4 | 14 | 1... 10 | 1 | $\left.\begin{array}{c} 16 \ldots 19 \\ 6 \end{array}\right)$ | 48 | Analog value overflow or broken wire at an analog input | Check input value or terminal |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |
| 4 | 14 | 1... 10 | 1 | $\begin{gathered} 16 \ldots 19 \\ 6 \end{gathered}$ | 7 | Analog value underflow at an analog input | Check input value |
|  | 11 / 12 | ADR | 1 ... 10 |  |  |  |  |
| 4 | 14 | 1... 10 | 1 | $\left.\begin{array}{c} 16 \ldots 19 \\ 6 \end{array}\right)$ | 47 | Short circuit at an analog input | Check terminal |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |
| 4 | 14 | 1... 10 | 3 | $\left.\begin{array}{c} 20 \ldots 21 \\ 7 \end{array}\right)$ | 4 | Analog value overflow at an analog output | Check output value |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |
| 4 | 14 | 1... 10 | 3 | $\left.\begin{array}{c} 20 \ldots 21 \\ 7 \end{array}\right)$ | 7 | Analog value underflow at an analog output | Check output value |
|  | 11 / 12 | ADR | $1 . . .10$ |  |  |  |  |

Remarks:

| ${ }^{1}$ ) | In AC500, the following interface identifier applies: $14=I / O$ bus, $11=$ COM1 (e.g. CS31 bus), $12=$ COM2 . <br> The FBP diagnosis block does not contain this identifier. |
| :---: | :---: |
| ${ }^{2}$ ) | With "Device" the following allocation applies: <br> 31 = module itself, <br> 1 ... $10=$ communication interface module 1 ... 10, <br> ADR = hardware address (e.g. of the DC551) |
| $\left.{ }^{3}\right)$ | With "Module" the following allocation applies depending on the master: <br> Module error: I/O bus or FBP: 31 = module itself; COM1/COM2: 1 ... $10=$ expansion 1 ... 10 <br> Channel error: I/O bus or FBP = module type ( 1 = AI, $3=\mathrm{AO}, 4=\mathrm{DC}$ ); COM1/ COM2: 1 ... 10 = expansion 1 ... 10 |
| $\left.{ }^{4}\right)$ | In case of module errors, with channel "31 = module itself" is output. |
| ${ }^{5}$ ) | Ch = 22 ... 29 indicates the digital inputs/outputs DC16 ... DC23 |
| ${ }^{6}$ ) | Ch = 16 ... 19 indicates the analog inputs AIO ... Al3 |
| ${ }^{7}$ ) | Ch $=20 \ldots 21$ indicates the analog outputs AO0 ... AO1 |

State LEDs

| LED |  |  | State | Color | LED = OFF | LED $=0 \mathrm{~N}$ | LED flashes |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| ATP DA501 | DA501 | DI0 ... DI15 | Digital input | Yellow | Input is OFF | Input is $\mathrm{ON}^{1}$ ) | -- |
|  |  | $\begin{aligned} & \text { DC16 ... } \\ & \text { DC23 } \end{aligned}$ | Digital input/ output | Yellow | Input/output is OFF | Input/output is ON ${ }^{1}$ ) | -- |
|  |  | AIO ... Al3 | Analog input | Yellow | Input is OFF | Input is $\mathrm{ON}^{2}$ ) | -- |
|  |  | $\begin{aligned} & \text { AOO ... } \\ & \text { AO1 } \end{aligned}$ | Analog output | Yellow | Output is OFF | Output is ON ${ }^{2}$ ) | -- |
|  |  | UP | Process supply voltage 24 V DC via terminal | Green | Process supply voltage is missing | Process supply voltage OK | -- |
|  |  | CH-ERR1 | Channel error, error messages in groups (digital inputs/ outputs combined into the groups 1, 2, 3, 4) | Red | No error or process supply voltage is missing | Severe error within the corresponding group | Severe error within the corresponding group (e.g. short circuit at an output) |
|  |  | CH-ERR2 |  | Red |  |  |  |
|  |  | CH-ERR3 |  | Red |  |  |  |
|  |  | CH-ERR4 |  | Red |  |  |  |
|  |  | CH-ERR ${ }^{3}$ ) | Module error | Red | -- | Internal error | -- |
|  |  | ${ }^{1}$ ) Indication LED is ON even if an input signal is applied to the channel and the supply voltage is off. In this case the module is not operating and does not generate an input signal. |  |  |  |  |  |
|  |  | ${ }^{2}$ ) Brightness depends on the value of the analog signal |  |  |  |  |  |
|  |  | ${ }^{3}$ ) All of the LEDs CH-ERR1 to CH-ERR4 light up together |  |  |  |  |  |

## Measuring ranges

Input ranges voltage, current and digital input

| Range | 0 V ... +10 | -10 V | 0 mA | 4 mA ... 20 | Digital | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  | Decimal | Hex. |
| Overflow | > 11.7589 | > 11.7589 | > 23.5178 | > 22.8142 |  | 32767 | 7FFF |
| Measured value too high | 11.7589 <br> 10.0004 | 11.7589 <br> 10.0004 | 23.5178 $:$ 20.0007 | 22.8142 $:$ 20.0006 |  | $\begin{aligned} & 32511 \\ & : \\ & 27649 \end{aligned}$ | $\begin{aligned} & \text { 7EFF } \\ & : \\ & 6 \mathrm{C} 01 \end{aligned}$ |
| Normal range <br> Normal <br> range or |  |  |  | 20.0000 $:$ 4.0006 | On | $\begin{array}{\|l} \hline 27648 \\ : \\ 1 \\ \hline \end{array}$ | $\begin{aligned} & \hline 6 \mathrm{C} 00 \\ & : \\ & 0001 \\ & \hline \end{aligned}$ |
| measured | 0.0000 | 0.0000 | 0 | 4 | Off | 0 | 0000 |
| $\begin{aligned} & \text { value too } \\ & \text { low } \end{aligned}$ | $\begin{array}{l\|} \hline-0.0004 \\ -1.7593 \end{array}$ | $\begin{aligned} & \hline-0.0004 \\ & : \\ & : \\ & -10,0000 \end{aligned}$ |  | $\begin{aligned} & 3.9994 \\ & 1.1858 \end{aligned}$ |  | $\begin{aligned} & -1 \\ & -4864 \\ & : \\ & -27648 \end{aligned}$ | $\begin{aligned} & \text { FFFF } \\ & \text { ED00 } \\ & : \\ & 9400 \end{aligned}$ |
| Measured value too low |  | -10.0004 <br> -11.7589 |  |  |  | -27649 <br> -32512 | $\begin{aligned} & \text { 93FF } \\ & : \\ & 8100 \end{aligned}$ |
| Underflow | < 1.7593 | <-11.7589 | < 0.0000 | < 1.1858 |  | -32768 | 8000 |

The represented resolution corresponds to 16 bits.

Input ranges resistance temperature detector

| Range | $\begin{aligned} & \text { Pt100 / Pt1000 } \\ & -50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \text { Pt100 / Pt1000 } \\ & -50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \mathrm{Ni} 1000 \\ & -50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C} \end{aligned}$ |
| :---: | :---: | :---: | :---: |
| Overflow | $>+80.0^{\circ} \mathrm{C}$ | $>+450.0{ }^{\circ} \mathrm{C}$ | $>+160.0{ }^{\circ} \mathrm{C}$ |
| Measured value too high |  | $\begin{aligned} & +450.0^{\circ} \mathrm{C} \\ & : \\ & +400.1^{\circ} \mathrm{C} \end{aligned}$ |  |
|  |  |  | $\begin{aligned} & +160.0^{\circ} \mathrm{C} \\ & : \\ & +150.1^{\circ} \mathrm{C} \end{aligned}$ |
|  | $\begin{aligned} & +80.0^{\circ} \mathrm{C} \\ & : \\ & +70.1^{\circ} \mathrm{C} \end{aligned}$ |  |  |
| Normal range | $\begin{aligned} & : \\ & : \\ & +70.0^{\circ} \mathrm{C} \\ & : \\ & +0.1^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & +400.0^{\circ} \mathrm{C} \\ & : \\ & \vdots \\ & +0.1^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & +150.0^{\circ} \mathrm{C} \\ & \vdots \\ & \vdots \\ & +0.1^{\circ} \mathrm{C} \end{aligned}$ |


| Range | $\begin{aligned} & \mathrm{Pt} 100 / \mathrm{Pt} 1000 \\ & -50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \text { Pt100 / Pt1000 } \\ & -50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \text { Ni1000 } \\ & -50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C} \end{aligned}$ |
| :---: | :---: | :---: | :---: |
|  | $0.0{ }^{\circ} \mathrm{C}$ | $0.0{ }^{\circ} \mathrm{C}$ | $0.0{ }^{\circ} \mathrm{C}$ |
|  | $\begin{aligned} & -0.1^{\circ} \mathrm{C} \\ & : \\ & -50.0^{\circ} \mathrm{C} \end{aligned}$ | $-0.1^{\circ} \mathrm{C}$ $:$ $-50.0^{\circ} \mathrm{C}$ | $-0.1^{\circ} \mathrm{C}$ <br> $-50.0^{\circ} \mathrm{C}$ |
| Measured value too low | $\begin{aligned} & -50.1^{\circ} \mathrm{C} \\ & = \\ & -60.0^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & -50.1^{\circ} \mathrm{C} \\ & : \\ & -60.0^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & -50.1^{\circ} \mathrm{C} \\ & : \\ & -60.0^{\circ} \mathrm{C} \end{aligned}$ |
| Underflow | <-60.0 ${ }^{\circ} \mathrm{C}$ | $<-60.0{ }^{\circ} \mathrm{C}$ | <-60.0 ${ }^{\circ} \mathrm{C}$ |


| Range | Digital value |  |
| :---: | :---: | :---: |
|  | Decimal | Hex. |
| Overflow | 32767 | 7FFF |
| Measured value too high | $\begin{aligned} & 4500 \\ & : \\ & 4001 \end{aligned}$ | $\begin{aligned} & 1194 \\ & : \\ & \text { 0FA1 } \end{aligned}$ |
|  | $\begin{aligned} & 1600 \\ & : \\ & 1501 \end{aligned}$ | $\begin{aligned} & 0640 \\ & : \\ & \text { 05DD } \end{aligned}$ |
|  | $\begin{aligned} & 800 \\ & : \\ & 701 \\ & 701 \end{aligned}$ | $\begin{aligned} & 0320 \\ & : \\ & \text { 02BD } \end{aligned}$ |
| Normal range | $\begin{aligned} & 4000 \\ & 1500 \\ & 700 \\ & : \\ & 1 \end{aligned}$ | $\begin{aligned} & \text { 0FAO } \\ & \text { 05DC } \\ & 02 B C \\ & \vdots \\ & 0001 \end{aligned}$ |
|  | 0 | 0000 |
|  | $-1$ $-500$ | $\begin{aligned} & \text { FFFF } \\ & : \\ & \text { FEOC } \end{aligned}$ |
| Measured value too low | $\begin{array}{\|l} \hline-501 \\ : \\ -600 \\ \hline \end{array}$ | $\begin{array}{\|l\|} \hline \text { FEOB } \\ : \\ \hline \end{array}$ |
| Underflow | -32768 | 8000 |

## Output ranges voltage and current

| Range | -10 V ... +10 V | $0 \mathrm{~mA} . . .20 \mathrm{~mA}$ | 4 mA ... 20 mA |
| :---: | :---: | :---: | :---: |
| Overflow | >11.7589 V | $>23.5178 \mathrm{~mA}$ | >22.8142 mA |
| Value too high | $\begin{aligned} & 11.7589 \mathrm{~V} \\ & : \\ & 10.0004 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 23.5178 \mathrm{~mA} \\ & : \\ & 20.0007 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 22.8142 \mathrm{~mA} \\ & : \\ & 20.0006 \mathrm{~mA} \end{aligned}$ |
| Normal range | $\begin{aligned} & 10.0000 \mathrm{~V} \\ & : \\ & 0.0004 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 20.0000 \mathrm{~mA} \\ & : \\ & 0.0007 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 20.0000 \mathrm{~mA} \\ & : \\ & 4.0006 \mathrm{~mA} \end{aligned}$ |
|  | 0.0000 V | 0.0000 mA | 4.0000 mA |
|  | $-0.0004 \mathrm{~V}$ $-10.0000 \mathrm{~V}$ | $\begin{aligned} & 0 \mathrm{~mA} \\ & : \\ & 0 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 3.9994 \mathrm{~mA} \\ & 0 \mathrm{~mA} \\ & 0 \mathrm{~mA} \end{aligned}$ |
| Value too low | $-10.0004 \mathrm{~V}$ -11.7589 V | $\begin{aligned} & 0 \mathrm{~mA} \\ & : \\ & 0 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 0 \mathrm{~mA} \\ & : \\ & 0 \mathrm{~mA} \end{aligned}$ |
| Underflow | 0 V | 0 mA | 0 mA |


| Range | Digital value | Hex. |
| :--- | :--- | :--- |
|  | Decimal | $>7$ EFF |
| Overflow | $>32511$ | 7EFF |
| Value too high | 32511 | 6 C01 |
|  | 27649 | 6 C00 |
| Normal range | 27648 | $:$ |
|  | 1 | 0001 |
|  | 0 | 0000 |
|  | -1 | FFFF |
|  | -6912 | E500 |
|  | -27648 | 9400 |
| Value too low | -27649 | $93 F F$ |
|  | $:$ | $:$ |
| Underflow | -32512 | 8100 |

The represented resolution corresponds to 16 bits.

## Technical data

## Technical data of the module

The system data of AC500 and S500 are applicable to the standard version ${ }^{*} \Rightarrow$ Chapter 5.1.2 "System data AC500" on page 166.

The system data of AC500-XC are applicable to the XC version ${ }^{〔}$ Chapter 5.1.3 "System data AC500-XC" on page 169.
Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter | Value |
| :---: | :---: |
| Process supply voltage |  |
| Connections | Terminals 1.8, 2.8, 3.8 and 4.8 for UP (+24 $\mathrm{V} D \mathrm{C}$ ) and 1.9, 2.9, 3.9 and 4.9 for $\mathrm{ZP}(0 \mathrm{~V}$ DC) |
| Protection against reverse voltage | yes |
| Rated protection fuse at UP | 10 A fast |
| Rated value | 24 V DC |
| Max. ripple | 5 \% |
| Current consumption |  |
| From UP | 0.07 A + max. 0.5 A per output |
| From 24 V DC power supply at the terminals UP/L+ and ZP/M of the CPU/communication interface module | ca. 2 mA |
| Inrush current from UP (at power-up) | $0.04 \mathrm{~A}^{2} \mathrm{~s}$ |
| Galvanic isolation | Yes, per module |
| Max. power dissipation within the module | 6 W (outputs unloaded) |
| Weight (without terminal unit) | ca. 125 g |
| Mounting position | Horizontal mounting or vertical with derating (output load reduced to $50 \%$ at $+40^{\circ} \mathrm{C}$ ) |
| Cooling | The natural convection cooling must not be hindered by cable ducts or other parts in the control cabinet. |

## NOTICE!

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and temporary overvoltage up to 30 V DC.

## Multiple overloads

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an internal smart high-side switch.

## Technical data of the digital inputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 16 |
| Distribution of the channels into groups | 2 groups of 8 channels |


| Parameter | Value |
| :---: | :---: |
| Terminals of the channels DIO ... DI7 | Terminals 1.0 ... 1.7 |
| Terminals of the channels DI8 ... DI15 | Terminals 2.0 ... 2.7 |
| Reference potential for all inputs | Terminals $1.9 \ldots 3.9$ (negative pole of the supply voltage, signal name ZP) |
| Indication of the input signals | 1 yellow LED per channel, the LED is ON when the input signal is high (signal 1) |
| Monitoring point of input indicator | LED is part of the input circuitry |
| Input type (according EN 61131-2) | Type 1 |
| Input delay (0->1 or 1->0) | Typ. 0.1 ms , configurable from $0.1 \ldots 32 \mathrm{~ms}$ |
| Input signal voltage | 24 V DC |
| 0-Signal | $-3 \vee \ldots+5 \mathrm{~V}$ |
| Undefined Signal | > +5V $\ldots<+15 \mathrm{~V}$ |
| 1-Signal | +15 V ... +30 V |
| Ripple with signal 0 | Within $-3 \vee \ldots+5 \mathrm{~V}$ |
| Ripple with signal 1 | Within +15 V ... +30 V |
| Input current per channel |  |
| Input voltage +24 V | Typ. 5 mA |
| Input voltage +5 V | > 1 mA |
| Input voltage +15 V | > 2 mA |
| Input voltage +30 V | $<8 \mathrm{~mA}$ |
| Max. cable length |  |
| Shielded | 1000 m |
| Unshielded | 600 m |

## Technical data of the configurable digital inputs/outputs

Each of the configurable digital I/O channels can be defined as input or output by the user program. This is done by interrogating or allocating the corresponding channel.

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 8 inputs/outputs (with transistors) |
| Distribution of the channels into groups | 1 group for 8 channels |
| If the channels are used as inputs |  |
|  | Channels DC16 ... DC23 |
| If the channels are used as outputs | Terminals 4.0 ... 4.7 |
|  | Channels DC16 ... DC23 |
| Indication of the input/output signals | Terminals 4.0 ... 4.7 <br> the input/output signal is high (signal 1) |
| Monitoring point of input/output indicator | LED is part of the input circuitry |
| Galvanic isolation | Yes, per module |

Technical data of the digital inputs/outputs if used as inputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 8 |
| Distribution of the channels into groups | 1 group of 8 channels |
| Terminals of the channels DC16 ... DC23 | Terminals 4.0 ... 4.7 |
| Reference potential for all inputs | Terminals 1.9 ... 4.9 (negative pole of the supply voltage, signal name ZP) |
| Indication of the input signals | 1 yellow LED per channel, the LED is ON when the input signal is high (signal 1) |
| Monitoring point of input/output indicator | LED is part of the input circuitry |
| Input type (according EN 61131-2) | Type 1 |
| Input delay (0->1 or 1->0) | Typ. 0.1 ms , configurable $0.1 \ldots 32 \mathrm{~ms}$ |
| Input signal voltage | 24 V DC |
| 0-Signal | -3 V ... +5 V |
| Undefined Signal | > +5V ... < +15V |
| 1-Signal | +15V ... +30 V |
| Ripple with signal 0 | Within $-3 \vee \ldots+5 \mathrm{~V}$ |
| Ripple with signal 1 | Within +15 V ... +30 V |
| Input current per channel |  |
| Input voltage +24 V | Typ. 5 mA |
| Input voltage +5 V | > 1 mA |
| Input voltage +15 V | > 2 mA |
| Input voltage +30 V | $<8 \mathrm{~mA}$ |
| Max. cable length |  |
| shielded | 1000 m |
| unshielded | 600 m |

* Due to the direct connection to the output, the demagnetizing varistor is also effective at the input (see figure) above. This is why the difference between UPx and the input signal must not exceed the clamp voltage of the varistor. The varistor limits the clamp voltage to approx. 36 V . Consequently, the input voltage must range from $-12 \mathrm{~V} \ldots+30 \mathrm{~V}$ when $\mathrm{UPx}=24 \mathrm{~V}$ and from $-6 \mathrm{~V} . .+30 \mathrm{~V}$ when $\mathrm{UPx}=30 \mathrm{~V}$.


## Technical data of the digital inputs/outputs if used as outputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 8 |
| Distribution of the channels into groups | 1 group of 8 channels |
| Terminals of the channels DC16 ...DC23 | Terminals 4.0 ... 4.7 |
| Reference potential for all outputs | Terminals $1.9 \ldots 4.9$ (negative pole of the <br> supply voltage, signal name ZP) |
| Common power supply voltage | For all outputs terminals 1.8, 2.8, 3.8 and 4.8 <br> (positive pole of the supply voltage, signal <br> name UP) |
| Output voltage for signal 1 | UP $(-0.8 \mathrm{~V})$ |


| Parameter | Value |
| :--- | :--- |
| Output delay (0->1 or 1->0) | On request |
| Output current |  |
|  | rated value per channel |
| max. value (all channels together) | 400 mA at UP $=24 \mathrm{~V}$ |
| Leakage current with signal 0 | $<0.5 \mathrm{~mA}$ |
| Fuse for UP | 10 A fast |
| Demagnetization with inductive DC load | Via internal varistors (see figure below this <br> table) |
| Output switching frequency |  |
|  | With resistive load |
| With inductive loads | On request |
| With lamp loads | 11 Hz max. at 5 W max. |
| Short-circuit-proof / overload-proof | Yes |
| Overload message (I > 0.7 A) | Yes, after ca. 100 ms |
| Output current limitation | Yes, automatic reactivation after short cir- <br> cuit/overload |
| Resistance to feedback against 24 V signals | Yes (software-controlled supervision) |
| Max. cable length |  |
|  | Shielded |
| Unshielded | 1000 m |

The following drawing shows the circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off.


Fig. 151: Digital input/output (circuit diagram)
1 Digital input/output
2 For demagnetization when inductive loads are turned off

## Technical data of the fast counter

The fast counter of the module does not work if the module is connected to an FBP interface module or CS31 bus module.

| Parameter | Value |
| :--- | :--- |
| Used inputs | DC16 / DC17 |
| Used outputs | DC18 |
| Counting frequency | Max. 50 kHz |

\& Chapter 6.8.2.12 "Fast counters in AC500 devices" on page 4536

## Technical data of the analog inputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 4 |
| Distribution of channels into groups | 1 group with 4 channels |
| Connection if channels $\mathrm{AlO}+\ldots \mathrm{Al} 3+$ | Terminals 3.0 ... 3.3 |
| Reference potential for $\mathrm{AlO}+\ldots \mathrm{Al3+}$ | Terminal 3.4 (AI-) for voltage and RTD measurement <br> Terminal 1.9, 2.9, 3.9 and 4.9 for current measurement |
| Input type |  |
| Unipolar | Voltage 0 V ... 10 V , current or Pt100/ Pt1000/Ni1000 |
| Bipolar | Voltage -10 V ... +10 V |
| Configurability | $0 \vee \ldots 10 \mathrm{~V},-10 \mathrm{~V} \ldots+10 \mathrm{~V}, 0 \mathrm{~mA} . .22 \mathrm{~mA}$, 4 mA ... $20 \mathrm{~mA}, \mathrm{Pt100/1000}, \mathrm{Ni1000} \mathrm{(each}$ input can be configured individually) |
| Channel input resistance | Voltage: > $100 \mathrm{k} \Omega$ <br> Current: ca. $330 \Omega$ |
| Time constant of the input filter | Voltage: $100 \mu \mathrm{~s}$ Current: $100 \mu \mathrm{~s}$ |
| Indication of the input signals | 1 LED per channel (brightness depends on the value of the analog signal) |
| Conversion cycle | 1 ms (for 4 inputs + 2 outputs); with RTDs Pt/Ni... 1 s |
| Resolution | Range 0 V ... 10 V : 12 bits <br> Range -10 V ... +10 V: 12 bits including sign <br> Range $0 \mathrm{~mA} . .20 \mathrm{~mA}$ : 12 bits <br> Range $4 \mathrm{~mA} . . .20 \mathrm{~mA}$ : 12 bits <br> Range RTD (Pt100, PT1000, Ni1000): 0.1 ${ }^{\circ} \mathrm{C}$ |
| Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range | Typ. 0.5 \%, max. 1 \% <br> For XC version below $0^{\circ} \mathrm{C}$ and above +60 ${ }^{\circ} \mathrm{C}$ : on request |
| Relationship between input signal and hex code | ²) Chapter 5.2.4.5.1.1.10.2 "Input ranges resistance temperature detector" on page 864 |
| Unused inputs | Are configured as "unused" (default value) |
| Overvoltage protection | Yes |

Technical data of the analog inputs, if used as digital inputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | Max. 4 |
| Distribution of channels into groups | 1 group of 4 channels |
| Connections of the channels $\mathrm{AlO+} . . \mathrm{Al} 3+$ | Terminals $3.0 \ldots 3.3$ |
| Reference potential for the inputs | Terminals $1.9,2.9,3.9$ and 4.9 (ZP) |
| Indication of the input signals | 1 LED per channel |
| Input signal voltage | 24 V DC |
|  | Signal 0 |
|  | Undefined signal |
|  | Signal 1 |
| Input current per channel | $+50 \mathrm{~V} \ldots+5 \mathrm{~V}$ |
|  | Input voltage +24 V |
|  | Input voltage +5 V |
|  | Input voltage +15 V |
|  | Input voltage +30 V |
| Input resistance | Typ. 7 mA |

## Technical data of the analog outputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 2 |
| Distribution of channels into groups | 1 group for 2 channels |
| Connection of the channels AO0+ ... AO1+ | Terminals 3.5 and 3.6 |
| Reference potential for AO0+ ... AO1+ | Terminal 3.7 (AO-) for voltage output <br> Terminals $1.9,2.9,3.9$ and 4.9 for current <br> output |
| Output type | Current |
|  | Unipolar |
| Bipolar | Against internal supply and other modules |
| Galvanic isolation | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}, 0 \mathrm{~mA} \ldots 20 \mathrm{~mA}, 4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ <br> (each output can be configured individually) |
| Configurability | $0 \Omega \ldots 500 \Omega$ |
| Output resistance (load) as current output | $\pm 10 \mathrm{~mA} \mathrm{max}$. |
| Output loadability as voltage output | 1 LED per channel (brightness depends on the <br> value of the analog signal) |
| Indication of the output signals | 12 bits including sign |
| Resolution | Typ. 5 ms |
| Settling time for full range change (resistive <br> load, output signal within specified tolerance) | Typ. $0.5 \%$, max. 1 \% <br> Conversion error of the analog values caused <br> by non-linearity, adjustment error at factory <br> and resolution within the normal range |


| Parameter | Value |
| :--- | :--- |
| Relationship between input signal and hex <br> code | „ Chapter 5.2.4.5.1.1.10.3 "Output ranges <br> voltage and current" on page 866 |
| Unused outputs | Are configured as "unused" (default value) and <br> can be left open-circuited |

## Internal data exchange

|  | Without the fast counter | With the fast counter (only <br> with AC500) |
| :--- | :--- | :--- |
| Digital inputs (bytes) | 3 | 5 |
| Digital outputs (bytes) | 1 | 3 |
| Analog inputs (words) | 4 | 4 |
| Analog outputs (words) | 2 | 2 |
| Counter input data (words) | 0 | 4 |
| Counter output data (words) | 0 | 8 |

## Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm

The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 250 700 R0001 | DA501, digital/analog input/output <br> module, 16 DI, 8 DC, 4 AI, 2 AO | Active |
| 1SAP 450 700 R0001 | DA501-XC, digital/analog input/output <br> module, 16 DI, 8 DC, 4 AI, 2 AO, <br> XC version | Active |

> *) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

## DA502 - Digital/Analog input/output module

## Features

- 16 digital outputs, $24 \mathrm{~V} \mathrm{DC}, 0.5 \mathrm{~A}$ max.
- 8 configurable digital inputs/outputs 24 V DC, 0.5 A max.
- 4 analog inputs, voltage, current and RTD, resolution 12 bits including sign
- 2 analog outputs, voltage and current, resolution 12 bits including sign
- Fast counter
- Module-wise galvanically isolated
- XC version for use in extreme ambient conditions available


1 I/O bus
2 Allocation between terminal number and signal name
316 yellow LEDs to display the signal states of the digital outputs DO0 ... DO15
44 yellow LEDs to display the signal states of the analog inputs AIO ... AI3
2 yellow LEDs to display the signal states of the analog outputs AO0 ... AO1
68 yellow LEDs to display the signal states of the configurable digital inputs/outputs DC16 ... DC23
71 green LED to display the state of the process supply voltage UP 4 red LEDs to display errors
Label
10 Terminal unit
11 DIN rail
楛 Sign for XC version

Intended purpose
The device can be used as a decentralized I/O extension module for S500 communication interface modules (e. g. CI592-CS31, CI501-PNIO, CI541-DP, CI581-CN) or as a centralized extension module for AC500 CPUs.

Functionality

| Parameter | Value |
| :--- | :--- |
| Fast counter | Integrated, many configurable operating <br> modes |
| Power supply | From the process supply voltage UP |
| LED displays | For system displays, signal states, errors and <br> power supply |
| Internal supply voltage | Via the I/O bus interface (I/O bus) |
| External supply voltage | Via terminals UP and ZP (process supply <br> voltage 24 V DC) |
| Effect of incorrect input terminal connection | Wrong or no signal detected, no damage up to <br> 35 V |
| Required terminal unit | TU515 or TU516 « Chapter 5.2.5.2 "TU515, <br> TU516, TU541 and TU542 for I/O modules" <br> on page 938 |

## Connections

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

The connection is carried out by using the 40 terminals of the terminal unit TU515/TU516 \# $\downarrow$ Chapter 5.2.5.2 "TU515, TU516, TU541 and TU542 for I/O modules" on page 938.
The assignment of the terminals:

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 1.0 | DO0 | Signal of the digital output DO0 |
| 1.1 | DO1 | Signal of the digital output DO1 |
| 1.2 | DO2 | Signal of the digital output DO2 |
| 1.3 | DO3 | Signal of the digital output DO3 |
| 1.4 | DO4 | Signal of the digital output DO4 |
| 1.5 | DO5 | Signal of the digital output DO5 |
| 1.6 | DO6 | Signal of the digital output DO6 |
| 1.7 | DO7 | Signal of the digital output DO7 |
| 1.8 | ZP | Process voltage UP (24 V DC) |
| 1.9 | DO8 | Process voltage ZP (0 V DC) |
| 2.0 | DO10 | Signal of the digital output DO8 |
| 2.1 | DO11 | Signal of the digigital output DO10 |
| 2.2 | DO12 | Signal of the digital output DO11 |
| 2.3 | DO13 | Signal of the digital output DO12 |
| 2.4 | DO14 | Signal of the digital output DO13 |
| 2.5 | Signal of the digital output DO14 |  |
| 2.6 |  |  |


| Terminal | Signal | Description |
| :---: | :---: | :---: |
| 2.7 | DO15 | Signal of the digital output DO15 |
| 2.8 | UP | Process voltage UP (24 V DC) |
| 2.9 | ZP | Process voltage ZP (0 V DC) |
| 3.0 | Al0+ | Positive pole of analog input signal 0 |
| 3.1 | Al1+ | Positive pole of analog input signal 1 |
| 3.2 | Al2+ | Positive pole of analog input signal 2 |
| 3.3 | Al3+ | Positive pole of analog input signal 3 |
| 3.4 | Al- | Negative pole of analog input signals 0 ... 3 |
| 3.5 | AOO+ | Positive pole of analog output signal 0 |
| 3.6 | AO1+ | Positive pole of analog output signal 1 |
| 3.7 | AO- | Negative pole of analog output signals 0 and 1 |
| 3.8 | UP | Process voltage UP ( 24 V DC) |
| 3.9 | ZP | Process voltage ZP (0 V DC) |
| 4.0 | DC16 | Signal of the configurable digital input/output DC16 |
| 4.1 | DC17 | Signal of the configurable digital input/output DC17 |
| 4.2 | DC18 | Signal of the configurable digital input/output DC18 |
| 4.3 | DC19 | Signal of the configurable digital input/output DC19 |
| 4.4 | DC20 | Signal of the configurable digital input/output DC20 |
| 4.5 | DC21 | Signal of the configurable digital input/output DC21 |
| 4.6 | DC22 | Signal of the configurable digital input/output DC22 |
| 4.7 | DC23 | Signal of the configurable digital input/output DC23 |
| 4.8 | UP | Process voltage UP ( 24 V DC) |
| 4.9 | ZP | Process voltage ZP (0 V DC) |

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a communication interface module or a CPU). Thus, the current consumption from 24 V DC power supply at the terminals L+/UP and M/ZP of the CPU/communication interface module increases by 2 mA per DA502.

The external power supply connection is carried out via the UP (+24 V DC) and the ZP (0 V DC) terminals.

## WARNING!

## Removal/Insertion under power

Removal or insertion under power is permissible only if all conditions for hot swapping are fullfilled.
« Chapter 4.9.3 "Replace an I/O module with hot swap" on page 153
The devices are not designed for removal or insertion under power when the conditions for hot swap do not apply. Because of unforeseeable consequences, it is not allowed to plug in or unplug devices with the power being ON.
Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while they are energized in a hazardous location could result in an electric arc, which could create an ignition source resulting in fire or explosion.

Prior to proceeding, make sure that power is been disconnected and that the area has been thoroughly checked to ensure that flammable materials are not present.

The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

## Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.


## NOTICE!

Risk of damaging the PLC modules!
The PLC modules must not be removed while the plant is connected to a power supply.
Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove or replace a module.


## CAUTION!

## Risk of imprecise and faulty measurements!

Analog signals may be distorted seriously by external electromagnetic influences.

Use shielded wires when wiring analog signal sources. The cable shield must be grounded at both ends of the cable. Provide a potential equalization of a low resistance to avoid high potential differences between different parts of the plant.


Fig. 152: Terminal assignment of the module
The module provides several diagnosis functions ${ }^{\mu}$ Chapter 5.2.4.5.1.2.8 "Diagnosis" on page 895.

## Connection of the digital outputs



Fig. 153: Connection of the digital outputs (DOO ... DO15)
For a description of the meaning of the LEDs, please refer to the Displays chapter $\Leftrightarrow$ Chapter 5.2.4.5.1.2.9 "State LEDs" on page 897.

## Connection of the configurable digital inputs/outputs



Fig. 154: Connection of the configurable digital input/outputs (DC16 ... DC23) (DC16 as an input, DC17 as an output)

## O NOTICE!

Risk of influences to the connected sensors!
Some sensors may be influenced by the deactivated module outputs of DA502.
Connect a $470 \Omega / 1 \mathrm{~W}$ resistor in series to inputs DC16/DC17 if they are used as fast counter inputs to avoid any influences.

For a description of the meaning of the LEDs, please refer to the Displays $\Leftrightarrow$ Chapter 5.2.4.5.1.2.9 "State LEDs" on page 897 chapter.

## Connection of resistance thermometers in 2-wire configuration to the analog inputs

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the module DA502 provides a constant current source which is multiplexed over max. 4 analog input channels.


Fig. 155: Connection of resistance thermometers in 2-wire configuration to the analog inputs (AIO ... Al3)

The following measuring ranges can be configured ${ }^{\wedge}$ Chapter 5.2.4.5.1.2.7 "Parameterization" on page $890 \stackrel{\leftrightarrow}{\otimes}$ Chapter 5.2.4.5.1.2.10 "Measuring ranges" on page 898:

| Pt100 | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 2-wire configuration, 1 <br> channel used |
| :--- | :--- | :--- |
| $\mathrm{Pt1000}$ | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 2-wire configuration, 1 <br> channel used |
| Ni1000 | $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ | 2-wire configuration, 1 <br> channel used |

For a description of the function of the LEDs, please refer to Diagnosis and displays / Displays « ${ }^{3}$ Chapter 5.2.4.5.1.2.9 "State LEDs" on page 897.

The module DA502 performs a linearization of the resistance characteristic.
To avoid error messages from unused analog input channels, configure them as "unused".

## Connection of resistance thermometers in 3-wire configuration to the analog inputs

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the module DA502 provides a constant current source which is multiplexed over max. 4 analog input channels.


Fig. 156: Connection of resistance thermometers in 3-wire configuration to the analog inputs (AIO ... Al3)

With 3-wire configuration, 2 adjacent analog channels belong together (e. g. the channels 0 and 1). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0), the next higher address must be the odd address (channel 1).
The constant current of one channel flows through the resistance thermometer. The constant current of the other channel flows through one of the cores. The module calculates the measured value from the two voltage drops and stores it under the input with the higher channel number (e. g. I1).

In order to keep measuring errors as small as possible, it is necessary to have all the involved conductors in the same cable. All the conductors must have the same cross section.
The following measuring ranges can be configured ${ }^{4}$, Chapter 5.2.4.5.1.2.7 "Parameterization" on page $890 \stackrel{\sharp}{\otimes}$ Chapter 5.2.4.5.1.2.10 "Measuring ranges" on page 898:

| Pt100 | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 3-wire configuration, 2 chan- <br> nels used |
| :--- | :--- | :--- |
| $\operatorname{Pt1000}$ | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 3-wire configuration, 2 chan- <br> nels used |
| Ni 1000 | $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ | 3-wire configuration, 2 chan- <br> nels used |

For a description of the function of the LEDs, please refer to Diagnosis and displays / Displays \& Chapter 5.2.4.5.1.2.9 "State LEDs" on page 897.
The module DA502 performs a linearization of the resistance characteristic.
To avoid error messages from unused analog input channels, configure them as "unused".

## Connection of active-type analog sensors (Voltage) with galvanically isolated power supply to the analog inputs



Fig. 157: Connection of active-type analog sensors (voltage) with galvanically isolated power supply to the analog inputs (AIO ... AI3)
The following measuring ranges can be configured ${ }^{\wedge}$ Chapter 5.2.4.5.1.2.7 "Parameterization" on page $890 \Leftrightarrow$ Chapter 5.2.4.5.1.2.10 "Measuring ranges" on page 898:

| Voltage | $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ | 1 channel used |
| :--- | :--- | :--- |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | 1 channel used |

For a description of the function of the LEDs, please refer to Diagnosis and displays / Displays Chapter 5.2.4.5.1.2.9 "State LEDs" on page 897.
To avoid error messages from unused analog input channels, configure them as "unused".

## Connection of active-type analog sensors (Current) with galvanically isolated power supply to the analog inputs



Fig. 158: Connection of active-type analog sensors (current) with galvanically isolated power supply to the analog inputs (AIO ... AI3)

The following measuring ranges can be configured $\Leftrightarrow$ Chapter 5.2.4.5.1.2.7 "Parameterization" on page $890 \Leftrightarrow$ Chapter 5.2.4.5.1.2.10 "Measuring ranges" on page 898:

| Current | $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 1 channel used |
| :--- | :--- | :--- |
| Current | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 1 channel used |

For a description of the function of the LEDs, please refer to Diagnosis and displays / Displays Chapter 5.2.4.5.1.2.9 "State LEDs" on page 897.
Unused input channels can be left open-circuited, because they are of low resistance.

## Connection of active-type analog sensors (Voltage) with no galvanically isolated power supply to the analog inputs



Fig. 159: Connection of active-type sensors (voltage) with no galvanically isolated power supply to the analog inputs (AIO ... AI3)

## CAUTION!

Risk of faulty measurements!
The negative pole at the sensors must not have too large a potential difference with respect to ZP (max. $\pm 1 \mathrm{~V}$ within the full signal range).
Make sure that the potential difference never exceeds $\pm 1 \mathrm{~V}$.

The following measuring ranges can be configured $\Leftrightarrow$ Chapter 5.2.4.5.1.2.7 "Parameterization" on page 890 « Chapter 5.2.4.5.1.2.10 "Measuring ranges" on page 898:

| Voltage | $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ | 1 channel used |
| :--- | :--- | :--- |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | 1 channel used |

For a description of the function of the LEDs, please refer to Diagnosis and displays / Displays \# Chapter 5.2.4.5.1.2.9 "State LEDs" on page 897.
To avoid error messages from unused analog input channels, configure them as "unused".

## Connection of passive-type analog sensors (Current) to the analog inputs



Fig. 160: Connection of passive-type analog sensors (current) to the analog inputs (AIO ... AI3)
The following measuring ranges can be configured ${ }^{4}$ Chapter 5.2.4.5.1.2.7 "Parameterization" on page $890 \stackrel{ }{y}$ Chapter 5.2.4.5.1.2.10 "Measuring ranges" on page 898:

| Current | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 1 channel used |
| :--- | :--- | :--- |

For a description of the function of the LEDs, please refer to Diagnosis and displays / Displays ${ }^{*}{ }^{*}$ Chapter 5.2.4.5.1.2.9 "State LEDs" on page 897.

## NOTICE!

## Risk of overloading the analog input!

If an analog current sensor supplies more than 25 mA for more than 1 second during initialization, this input is switched off by the module (input protection).

Use only sensors with fast initialization or without current peaks higher than 25 mA . If not possible, connect a 10 -volt Zener diode in parallel to I+ and ZP.

Unused input channels can be left open-circuited, because they are of low resistance.

## Connection of active-type analog sensors (Voltage) to differential analog inputs

Differential inputs are very useful if analog sensors which are remotely non-isolated (e.g. the negative terminal is remotely grounded) are used.

Using differential inputs helps to considerably increase the measuring accuracy and to avoid ground loops.
With differential input configurations, two adjacent analog channels belong together (e.g. the channels 0 and 1). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0), the next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1).
The analog value is calculated by subtraction of the input value with the higher address from the input value of the lower address.

The converted analog value is available at the odd channel (higher address).

## CAUTION!

## Risk of faulty measurements!

The negative pole at the sensors must not have too large a potential difference with respect to ZP (max. $\pm 1 \mathrm{~V}$ within the full signal range).
Make sure that the potential difference never exceeds $\pm 1 \mathrm{~V}$.


Fig. 161: Connection of active-type analog sensors (voltage) to differential analog inputs (AIO ... Al3)
The following measuring ranges can be configured ${ }^{4}$ Chapter 5.2.4.5.1.2.7 "Parameterization" on page $890 \Leftrightarrow$ Chapter 5.2.4.5.1.2.10 "Measuring ranges" on page 898:

| Voltage | $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ | with differential inputs, 2 chan- <br> nels used |
| :--- | :--- | :--- |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | with differential inputs, 2 chan- <br> nels used |

For a description of the function of the LEDs, please refer to Diagnosis and displays / Displays ${ }^{*}{ }^{\circ}$ Chapter 5.2.4.5.1.2.9 "State LEDs" on page 897.

To avoid error messages from unused analog input channels, configure them as "unused".

## Use of analog inputs as digital inputs

Several (or all) analog inputs can be configured as digital inputs. The inputs are not galvanically isolated against the other analog channels.


Fig. 162: connection of digital sensors to the analog input (AIO ... AI3)
The following measuring ranges can be configured $«$ Chapter 5.2.4.5.1.2.7 "Parameterization" on page 890 Chapter 5.2.4.5.1.2.10 "Measuring ranges" on page 898 :

| Digital input | 24 V | 1 channel used |
| :--- | :--- | :--- |

For a description of the function of the LEDs, please refer to Diagnosis and displays / Displays * Chapter 5.2.4.5.1.2.9 "State LEDs" on page 897.

## Connection of analog output loads (Voltage)



Fig. 163: Connection of analog output loads (voltage) to the analog outputs (AOO ... AO1)

The following measuring ranges can be configured ${ }^{\star}$ Chapter 5.2.4.5.1.2.7 "Parameterization" on page 890 \# $\Longleftrightarrow$ Chapter 5.2.4.5.1.2.10 "Measuring ranges" on page 898:

| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | Load $\pm 10 \mathrm{~mA}$ max. | 1 channel used |
| :--- | :--- | :--- | :--- |

For a description of the function of the LEDs, please refer to Diagnosis and displays / Displays * ${ }^{*}$ Chapter 5.2.4.5.1.2.9 "State LEDs" on page 897.

Unused analog outputs can be left open-circuited.

## Connection of analog output loads (Current)



Fig. 164: Connection of analog output loads (current) to the analog outputs (AO0 ... AO1)
The following measuring ranges can be configured $\Leftrightarrow$ Chapter 5.2.4.5.1.2.7 "Parameterization" on page 890 出 Chapter 5.2.4.5.1.2.10 "Measuring ranges" on page 898:

| Current | $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | Load $0 \Omega \ldots 500 \Omega$ | 1 channel used |
| :--- | :--- | :--- | :--- |
| Current | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | Load $0 \Omega \ldots 500 \Omega$ | 1 channel used |

For a description of the function of the LEDs, please refer to Diagnosis and displays / Displays \& Chapter 5.2.4.5.1.2.9 "State LEDs" on page 897.
Unused analog outputs can be left open-circuited.

Internal data exchange

|  | Without the fast counter | With the fast counter (only <br> with AC500) |
| :--- | :--- | :--- |
| Digital inputs (bytes) | 1 | 1 |
| Digital outputs (bytes) | 3 | 3 |
| Analog inputs (words) | 4 | 4 |
| Analog outputs (words) | 2 | 2 |


|  | Without the fast counter | With the fast counter (only <br> with AC500) |
| :--- | :--- | :--- |
| Counter input data (words) | 0 | 5 |
| Counter output data (words) | 0 | 9 |

## I/O configuration

The module itself does not store configuration data. It draws its parameterization data from the master device of the I/O bus (CPU or communication interface module) during power-up of the system.

Hence, replacing I/O modules is possible without any re-parameterization via software.


If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.

## Parameterization

| Firmware version | Configuration |
| :--- | :--- |
| Firmware version > V2.0.0 | The arrangement of the parameter data is per- <br> formed by Control Builder Plus/ Automation <br> Builder software. |

The parameter data directly influences the functionality of modules.
For non-standard applications, it is necessary to adapt the parameters to your system configuration.

Module: Module slot address: $\mathrm{Y}=1$... 10

| Name | Value | Internal value | Internal value, type | Default | EDS Slot / Index |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Module ID ${ }^{1}$ ) | Internal | 1815 | WORD | 1815 | 0x0Y01 |
| Ignore module | Internal | Yes <br> No | BYTE | No |  |
| Parameter length | Internal | 8 | BYTE | 8 | 0xY02 |
| Check supply | off | 0 | BYTE | 1 | 0xY03 |
|  | on | 1 |  |  |  |


| Name | Value | Internal value | Internal value, type | Default | EDS Slot / Index |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Fast counter ${ }^{3}$ ) | $\begin{aligned} & 0 \\ & : \\ & \left.10^{2}{ }^{2}\right) \end{aligned}$ | 0 $10$ | BYTE | 0 | Not for FBP |
| Behavior outputs at comm. error ${ }^{5}$ ) | Off Last value Last value 5 s Last value 10 s Substitute value <br> Substitute value 5 s <br> Substitute value 10 s | $\begin{array}{\|l} \hline 0 \\ 16 \\ 11 \\ 2 \\ 7 \\ 12 \end{array}$ | BYTE | $\begin{array}{\|l\|} \hline \text { Off } \\ 0 \times 00 \end{array}$ | 0x0Y07 |


| $\mathbf{2}^{\mathbf{2}}$ | Setting | Description |
| :--- | :--- | :--- |
|  | On | Error LED lights up at errors of all error <br> classes, Failsafe mode off |
|  | Off by E4 | Error LED lights up at errors of error <br> classes E1, E2 and E3, Failsafe mode off |
|  | Off by E3 | Error LED lights up at errors of error <br> classes E1 and E2, Failsafe mode off |
|  | Off by E4 + Failsafe | Error LED lights up at errors of all error <br> classes, Failsafe mode on *) |
|  | Off by E3 + Failsafe | Error LED lights up at errors of error <br> classes E1, E2 and E3, Failsafe mode on <br> $*)$ |
|  | Error LED lights up at errors of error <br> classes E1 and E2, Failsafe mode on * |  |

${ }^{1}$ ) With a faulty ID, the module reports a "parameter error" and does not perform cyclic process data transmission
${ }^{2}$ ) For a description of the counter operating modes, please refer to the 'Fast Counter' section « Chapter 5.2.4.3.2.9 "Fast counter" on page 606
${ }^{3}$ ) With CS31 without the parameter "Fast Counter"

The fast counter of the module does not work if the module is connected to a CS31 bus module.
${ }^{5}$ ) The parameter Behavior outputs at comm. error is only analyzed if the Failsafe mode is ON.

## Group parameters for the digital part

| Name | Value | Internal value | Internal <br> value, type | Default | EDS Slot $/$ <br> Index |
| :--- | :--- | :--- | :--- | :--- | :--- |
| Input delay | 0.1 ms <br> 1 ms <br> 8 ms <br> 32 ms | 0 | 1 | BYTE | 0.1 ms <br> 0 <br> 3 |
| Detect short <br> circuit at out- <br> puts | Off <br> On | 0 | $0 x 0 \mathrm{Y} 05$ |  |  |
| Substitute <br> value at <br> output | $0 \ldots 255$ | $00 \mathrm{~h} \ldots$ FFh | BYTE | 0 <br> $0 x 0000$ | BYTE |

${ }^{*}$ ) The parameters Behavior DO at comm. error is only analyzed if the Failsafe mode is ON.

Group parameters for the analog part

| Name | Value | Internal value | Internal <br> value, type | Default | EDS Slot $/$ <br> Index |
| :--- | :--- | :--- | :--- | :--- | :--- |
| Analog data <br> format | Standard <br> Reserved | 255 <br> RYTE | 0 | $0 \times 0 \mathrm{Y04}$ |  |

${ }^{*}$ ) The parameter Behaviour AO at comm. error is only analyzed if the Failsafe mode is ON.

Channel parameters for the analog inputs (4x)

| Name | Value | Internal value | Internal value, type | Default | EDS Slot / Index |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Input 0, Channel configuration | see table 'Channel configuration' Table 201" Channel configuration" on page 893 | see table 'Channel configuration' ※ Table 201" Channel configuration" on page 893 | BYTE | 0 | 0x0Y09 |
| Input 0, Check channel | see table 'Channel monitoring' ̌ Table 202" Channel monitoring" on page 893 | see table 'Channel monitoring' ² Table 202" Channel monitoring" on page 893 | BYTE | 0 | 0xOYOA |
| : | : | : | : | : |  |
| : | : | : | : | : |  |


| Name | Value | Internal value | Internal value, type | Default | EDS Slot / Index |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Input 3, Channel configuration | see table 'Channel configuration' Table 201 " Channel configuration" on page 893 | see table 'Channel configuration' « Table 201" Channel configuration" on page 893 | BYTE | 0 | 0x0Y0F |
| Input 3, Check channel | see table 'Channel monitoring' 4y Table 202 Channel monitoring" on page 893 | see table 'Channel monitoring' Table 202 " Channel monitoring" on page 893 | BYTE | 0 | 0x0Y10 |

Table 201: Channel configuration

| Internal value | Operating modes of the analog inputs, individually configurable |
| :---: | :---: |
| 0 (default) | Not used |
| 1 | 0 V ... 10 V |
| 2 | Digital input |
| 3 | 0 mA ... 20 mA |
| 4 | $4 \mathrm{~mA} . . .20 \mathrm{~mA}$ |
| 5 | -10 V ... +10 V |
| 8 | 2-wire Pt100-50 ${ }^{\circ} \mathrm{C} \ldots+400{ }^{\circ} \mathrm{C}$ |
| 9 | 3-wire Pt100-50 ${ }^{\circ} \mathrm{C} \ldots+40{ }^{\circ} \mathrm{C}$ *) |
| 10 | 0 V ... 10 V (voltage diff.) *) |
| 11 | -10 V ... +10 V (voltage diff.) *) |
| 14 | 2-wire Pt100-50 ${ }^{\circ} \mathrm{C} \ldots+70{ }^{\circ} \mathrm{C}$ |
| 15 | 3-wire Pt100 -50 ${ }^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ *) |
| 16 | 2-wire Pt1000-50 ${ }^{\circ} \mathrm{C} \ldots+400{ }^{\circ} \mathrm{C}$ |
| 17 | 3-wire Pt1000-50 ${ }^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}{ }^{*}$ ) |
| 18 | 2-wire Ni1000-50 ${ }^{\circ} \mathrm{C} \ldots+150{ }^{\circ} \mathrm{C}$ |
| 19 | 3-wire Ni1000-50 ${ }^{\circ} \mathrm{C} \ldots+150{ }^{\circ} \mathrm{C}$ *) |
|  | *) In the operating modes with 3-wire configuration or with differential inputs, two adjacent analog inputs belong together (e.g. the channels 0 and 1). In these cases, both channels are configured in the desired operating mode. The lower address must be the even address (channel 0). The next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1). |

Table 202: Channel monitoring

| Internal Value | Check Channel |
| :--- | :--- |
| 0 (default) | Plausibility, wire break, short circuit |
| 3 | Not used |

## Channel parameters for the analog outputs (2x)

| Name | Value | Internal value | Internal value, type | Default | $\begin{array}{\|l} \hline \text { EDS Slot / } \\ \text { Index } \end{array}$ |
| :---: | :---: | :---: | :---: | :---: | :---: |
| 0 <br> Output 0, Channel configuration | see table 'Channel configuration' (4) Table 203 Channel configuration" on page 894 | see table 'Channel configuration' Table 203" Channel configuration" on page 894 | BYTE | 0 | 0x0Y11 |
| Output 0, Check channe | see table 'Channel monitoring' ③ Table 204" Channel monitoring" on page 895 | see table 'Channel monitoring' ③ Table 204" Channel monitoring" on page 895 | BYTE | 0 | 0x0Y12 |
| Output 0, Substitute value | see table <br> 'Substitute <br> values' <br> ③ Table 205" <br> Substitute <br> value" <br> on page 895 | see table 'Substitute values' Table 205 " Substitute value" on page 895 | WORD | 0 | 0x0Y13 |
| Output 1, Channel configuration | see table 'Channel configuration' Table 203 Channel configuration" on page 894 | see table 'Channel configuration' * Table 203" Channel configuration" on page 894 | BYTE | 0 | 0x0Y14 |
| Output 1, Check channel | see table 'Channel monitoring' ⓨ Table 204 " Channel monitoring" on page 895 | see table 'Channel monitoring' ² Table 204" Channel monitoring" on page 895 | BYTE | 0 | 0x0Y15 |
| Output 1, Substitute value | see table <br> 'Substitute values' <br> Table 205" <br> Substitute value" on page 895 | see table 'Substitute values' ④ Table 205 " Substitute value" on page 895 | WORD | 0 | 0x0Y16 |

Table 203: Channel configuration

| Internal value | Operating modes of the analog outputs, individually configu- <br> rable |
| :--- | :--- |
| 0 (default) | Not used |
| 128 | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ |
| 129 | $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ |
| 130 | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ |

Table 204: Channel monitoring

| Internal value | Check channel |
| :--- | :--- |
| 0 | Plausibility, wire break, short circuit |
| 3 | None |

Table 205: Substitute value

| Intended behavior of output <br> channel when the control <br> system stops | Required setting of <br> the module parameter <br> "Behavior of outputs in <br> case of a communication <br> error" | Required setting of the <br> channel parameter "Substi- <br> tute value" |
| :--- | :--- | :--- |
| Output OFF | Off | 0 |
| Last value infinite | Last value | 0 |
| Last value for 5 s and then <br> turn off | Last value 5 s | 0 |
| Last value for 10 s and then <br> turn off | Last value 10 s | 0 |
| Substitute value infinite | Substitute value | Depending on configuration |
| Substitute value for 5 s and <br> then turn off | Substitute value 5 s | Depending on configuration |
| Substitute value for 10 s and <br> then turn off | Substitute value 10 s | Depending on configuration |

## Diagnosis

In cases of short circuit or overload, the digital outputs are turned off. The module performs reactivation automatically. Thus, an acknowledgement of the errors is not necessary. The error message is stored via the LED.

| E1 ... E4 | d1 | d2 | d3 | d4 | $\begin{array}{\|l\|} \hline \text { Identifier } \\ 000 \ldots 063 \end{array}$ | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err |  |  |
| Byte 6 <br> Bit 6 ... 7 | - | Byte 3 | Byte 4 | Byte 5 | Byte 6 <br> Bit 0 ... 5 |  |  |
| Class | Interface | Device | Module | Channel | Error Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | $\left.{ }^{3}\right)$ | ${ }^{4}$ ) |  |  |  |
| Module error |  |  |  |  |  |  |  |
| 3 | 14 | $1 . .10$ | 31 | 31 | 19 | Checksum error in the I/O module | Replace I/O module |
|  | 11 / 12 | ADR | 1... 10 |  |  |  |  |
| 3 | 14 | $1 \ldots 10$ | 31 | 31 | 3 | Timeout in the I/O module |  |
|  | 11 / 12 | ADR | 1... 10 |  |  |  |  |
| 3 | 14 | $1 \ldots 10$ | 31 | 31 | 40 | Different hard-/firmware versions in the module |  |
|  | 11 / 12 | ADR | 1... 10 |  |  |  |  |


| E1 ... E4 | d1 | d2 | d3 | d4 | $\begin{array}{\|l\|} \hline \text { Identifier } \\ 000 \ldots 063 \end{array}$ | AC500 display | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err | $\begin{array}{\|l} \text { PS501 } \\ \text { PLC } \\ \text { browser } \end{array}$ |  |  |
| Byte 6 <br> Bit 6 ... 7 | - | Byte 3 | Byte 4 | Byte 5 | Byte 6 <br> Bit 0 ... 5 | FBP diagnosis block |  |  |
| Class | Interface | Device | Module | Channel | Error Identifier | Error message |  | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | $\left.{ }^{3}\right)$ | ${ }^{4}$ ) |  |  |  |  |
| 3 | 14 | $1 \ldots 10$ | 31 | 31 | 43 | Internal error in the module |  |  |
|  | $11 / 12$ | ADR | $1 . .10$ |  |  |  |  |  |  |  |
| 3 | 14 | $1 . .10$ | 31 | 31 | 36 | Internal data exchange failure |  |  |
|  | $11 / 12$ | ADR | 1... 10 |  |  |  |  |  |  |  |
| 3 | 14 | $1 \ldots 10$ | 31 | 31 | 9 | Overflow diagnosis buffer |  | New start |
|  | $11 / 12$ | ADR | 1... 10 |  |  |  |  |  |  |  |
| 3 | 14 | $1 \ldots 10$ | 31 | 31 | 26 | Parameter error |  | Check master |
|  | $11 / 12$ | ADR | $1 . .10$ |  |  |  |  |  |  |  |
| 3 | 14 | $1 \ldots 10$ | 31 | 31 | 11 | Process voltage too low |  | Check process voltage |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |  |  |  |
| 4 | 14 | 1... 10 | 31 | 31 | 45 | Process voltage is switched off (ON -> OFF) |  | Process voltage ON |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |  |  |  |
| Channel error DA502 |  |  |  |  |  |  |  |  |
| 4 | 14 | $1 \ldots 10$ | 2 | $\left.\begin{array}{lll} 0 & \ldots & 15 \\ 22 & \ldots & 29 \\ 5 \end{array}\right)$ | 47 | Short-circuit at a digital output |  | Check connection |
|  | 11 / 12 | ADR | 1... 10 |  |  |  |  |  |  |  |
| Channel error DA502 |  |  |  |  |  |  |  |  |
| 4 | 14 | $1 \ldots 10$ | 1 | $\left.\begin{array}{c} 16 \ldots 19 \\ 6 \end{array}\right)$ | 48 | Analog value overflow or broken wire at an analog input |  | Check input value or terminal |
|  | 11 / 12 | ADR | 1... 10 |  |  |  |  |  |  |  |
| 4 | 14 | $1 . . .10$ | 1 | $\left.\begin{array}{c} 16 \ldots 19 \\ 6 \end{array}\right)$ | 7 | Analog value underflow at an analog input |  | Check input value |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |  |  |  |
| 4 | 14 | $1 \ldots 10$ | 1 | $\left.\begin{array}{c} 16 \ldots 19 \\ 6 \end{array}\right)$ | 47 | Short circuit at an analog input |  | Check terminal |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |  |  |  |
| 4 | 14 | $1 \ldots 10$ | 3 | $\begin{gathered} 20 \ldots 21 \\ 7 \end{gathered}$ | 4 | Analog value overflow at an analog output |  | Check output value |
|  | 11/12 | ADR | $1 . .10$ |  |  |  |  |  |  |  |
| 4 | 14 | 1... 10 | 3 | $\begin{gathered} 20 \ldots 21 \\ 7 \end{gathered}$ | 7 | Analog value underflow at an analog output |  | Check output value |
|  | 11 / 12 | ADR | $1 . .10$ |  |  |  |  |  |  |  |

Remarks:

| ${ }^{1}$ ) | In AC500, the following interface identifier applies: 14 = I/O bus, 11 = COM1 (e.g. CS31 bus), 12 = COM2. |
| :---: | :---: |
| ${ }^{2}$ ) | With "Device" the following allocation applies: <br> 31 = module itself, <br> 1 ... $10=$ communication interface module 1 ... 10 , <br> ADR = hardware address (e.g. of the DC551) |
| $\left.{ }^{3}\right)$ | With "Module" the following allocation applies depending on the master: Module error: I/O bus: 31 = Module itself; COM1/COM2: 1 ... 10 = expansion 1 ... 10 <br> Channel error: I/O bus = module type ( $1=\mathrm{AI}, 3=\mathrm{AO}, 4=\mathrm{DC}$ ); COM1/COM2: 1 ... 10 = expansion 1 ... 10 |
| ${ }^{4}$ ) | In case of module errors, with channel "31 = module itself" is output. |
| ${ }^{5}$ ) | Ch = 22 ... 29 indicate the digital inputs/outputs DC16 ... DC23 |
| ${ }^{6}$ ) | Ch = 16 ... 19 indicates the analog inputs AIO ... Al3 |
| ${ }^{7}$ ) | Ch = 20 ... 21 indicates the analog outputs AO0 ... AO1 |

## State LEDs

| LED |  |  | State | Color | LED = OFF | LED = ON | LED flashes |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| AB | DA502 | $\begin{aligned} & \hline \text { DO0 ... } \\ & \text { DO15 } \end{aligned}$ | Digital output | Yellow | Output is OFF | Output is ON | -- |
|  |  | $\begin{aligned} & \mathrm{DC16} \ldots \mathrm{D} \\ & \mathrm{C} 23 \end{aligned}$ | Digital input/ output | Yellow | Input/output is OFF | Input/output is ON ${ }^{1}$ ) | -- |
|  |  | AIO ... Al3 | Analog input | Yellow | Input is OFF | Input is $\mathrm{ON}^{2}$ ) | -- |
|  |  | $\begin{aligned} & \text { AO0 ... } \\ & \text { AO1 } \end{aligned}$ | Analog output | Yellow | Output is OFF | Output is ON ${ }^{2}$ ) | -- |
|  |  | UP | Process supply voltage 24 V DC via terminal | Green | Process supply voltage is missing | Process supply voltage OK | -- |
|  |  | CH-ERR1 | Channel error, error messages in groups (digital inputs/ outputs combined into the groups 1, 2, 3, 4) | Red | No error or process supply voltage is missing | Severe error within the corresponding group | Severe error within the corresponding group (e.g. short circuit at an output) |
|  |  | CH-ERR2 |  | Red |  |  |  |
|  |  | CH-ERR3 |  | Red |  |  |  |
|  |  | CH-ERR4 |  | Red |  |  |  |
|  |  | CH-ERR ${ }^{3}$ ) | Module error | Red | -- | Internal error | -- |
|  |  | ${ }^{1}$ ) Indication LED is ON even if an input signal is applied to the channel and the supply voltage is off. In this case the module is not operating and does not generate an input signal. |  |  |  |  |  |
|  |  | ${ }^{2}$ ) Brightness depends on the value of the analog signal |  |  |  |  |  |
|  |  | ${ }^{3}$ ) All of the LEDs CH-ERR1 to CH-ERR4 light up together |  |  |  |  |  |

## Measuring ranges

Input ranges voltage, current and digital input

| Range | 0 V ... +10 | $-10 \mathrm{~V} . .$ | $0 \mathrm{~mA}$ | $4 \mathrm{~mA} . . .20$ | Digital | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  | Decimal | Hex. |
| Overflow | > 11.7589 | > 11.7589 | > 23.5178 | > 22.8142 |  | 32767 | 7FFF |
| Measured value too high |  |  | 23.5178 <br> 20.0007 |  |  | $\begin{aligned} & 32511 \\ & : \\ & 27649 \end{aligned}$ | $\begin{aligned} & \text { 7EFF } \\ & : \\ & 6 \mathrm{C} 01 \end{aligned}$ |
| Normal range <br> Normal range or | $\begin{aligned} & 10.0000 \\ & : \\ & 0.0004 \end{aligned}$ | $\begin{aligned} & 10.0000 \\ & : \\ & 0.0004 \end{aligned}$ | $\begin{aligned} & 20.0000 \\ & : \\ & 0.0007 \end{aligned}$ | $\begin{aligned} & 20.0000 \\ & : \\ & 4.0006 \end{aligned}$ | On | $27648$ <br> 1 | $\begin{aligned} & 6 \mathrm{C} 00 \\ & : \\ & 0001 \end{aligned}$ |
| measured | 0.0000 | 0.0000 | 0 | 4 | Off | 0 | 0000 |
| low | $\begin{aligned} & \hline-0.0004 \\ & -1.7593 \end{aligned}$ | $\begin{aligned} & -0.0004 \\ & : \\ & : \\ & -10,0000 \end{aligned}$ |  | $\begin{aligned} & \hline 3.9994 \\ & 1.1858 \end{aligned}$ |  | $\begin{aligned} & -1 \\ & -4864 \\ & : \\ & -27648 \end{aligned}$ | $\begin{aligned} & \hline \text { FFFF } \\ & \text { EDOO } \\ & : \\ & 9400 \end{aligned}$ |
| Measured value too low |  | -10.0004 <br> -11.7589 |  |  |  | -27649 $:$ -32512 | $\begin{aligned} & \hline 93 F F \\ & : \\ & 8100 \\ & \hline \end{aligned}$ |
| Underflow | < 1.7593 | <-11.7589 | < 0.0000 | < 1.1858 |  | -32768 | 8000 |

The represented resolution corresponds to 16 bits.

Input ranges resistance temperature detector

| Range | $\begin{aligned} & \begin{array}{l} \mathrm{Pt} 100 \mathrm{I} \\ \mathrm{Pt} 1000 \\ -50^{\circ} \mathrm{C} \ldots+70 \end{array}{ }^{\circ} \mathrm{C} \text {. } \end{aligned}$ | $\begin{aligned} & \hline \mathrm{Pt} 100 \mathrm{I} \\ & \mathrm{Pt} 1000 \\ & -50^{\circ} \mathrm{C} \ldots \\ & +400^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \hline \mathrm{Ni} 1000 \\ & -50^{\circ} \mathrm{C} \ldots \\ & +150^{\circ} \mathrm{C} \end{aligned}$ | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  | Decimal | Hex. |
| Overflow | $>+80.0{ }^{\circ} \mathrm{C}$ | $>+450.0^{\circ} \mathrm{C}$ | $>+160.0{ }^{\circ} \mathrm{C}$ | 32767 | 7FFF |
| Measured value too high |  | $\begin{aligned} & +450.0^{\circ} \mathrm{C} \\ & : \\ & +400.1^{\circ} \mathrm{C} \end{aligned}$ |  | $\begin{aligned} & 4500 \\ & : \\ & 4001 \\ & \hline \end{aligned}$ | $\begin{aligned} & 1194 \\ & : \\ & \text { OFA1 } \\ & \hline \end{aligned}$ |
|  |  |  | $\begin{aligned} & +160.0^{\circ} \mathrm{C} \\ & : \\ & +150.1^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & 1600 \\ & : \\ & 1501 \end{aligned}$ | $\begin{aligned} & 0640 \\ & : \\ & 05 D D \end{aligned}$ |
|  | $\begin{aligned} & +80.0^{\circ} \mathrm{C} \\ & : \\ & +70.1^{\circ} \mathrm{C} \end{aligned}$ |  |  | $\begin{aligned} & 800 \\ & : \\ & 701 \end{aligned}$ | $\begin{aligned} & 0320 \\ & : \\ & 02 B D \end{aligned}$ |


| Range | $\begin{aligned} & \mathrm{Pt100} \text { I } \\ & \mathrm{Pt} 1000 \\ & -50^{\circ} \mathrm{C} \ldots+70 \\ & { }^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \mathrm{Pt} 100 \mathrm{I} \\ & \mathrm{Pt} 1000 \\ & -50^{\circ} \mathrm{C} \ldots \\ & +400^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \mathrm{Ni} 1000 \\ & -50^{\circ} \mathrm{C} \ldots \\ & +150^{\circ} \mathrm{C} \end{aligned}$ | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  | Decimal | Hex. |
| Normal range | $\begin{aligned} & +70.0^{\circ} \mathrm{C} \\ & : \\ & +0.1^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & +400.0^{\circ} \mathrm{C} \\ & : \\ & : \\ & \vdots \\ & +0.1^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & +150.0^{\circ} \mathrm{C} \\ & : \\ & \vdots \\ & +0.1^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & 4000 \\ & 1500 \\ & 700 \\ & : \\ & 1 \end{aligned}$ | $\begin{aligned} & \hline \text { 0FA0 } \\ & \text { 05DC } \\ & \text { 02BC } \\ & : \\ & 0001 \end{aligned}$ |
|  | $0.0{ }^{\circ} \mathrm{C}$ | $0.0{ }^{\circ} \mathrm{C}$ | $0.0{ }^{\circ} \mathrm{C}$ | 0 | 0000 |
|  | $\begin{aligned} & -0.1^{\circ} \mathrm{C} \\ & : \\ & -50.0^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & -0.1^{\circ} \mathrm{C} \\ & : \\ & -50.0^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & -0.1^{\circ} \mathrm{C} \\ & : \\ & -50,0^{\circ} \mathrm{C} \end{aligned}$ | $-1$ $-500$ | $\begin{aligned} & \text { FFFF } \\ & : \\ & \text { FEOC } \end{aligned}$ |
| Measured value too low | $\begin{aligned} & -50.1^{\circ} \mathrm{C} \\ & : \\ & -60.0^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & -50.1^{\circ} \mathrm{C} \\ & : \\ & -60.0^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & -50.1^{\circ} \mathrm{C} \\ & : \\ & -60.0^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & -501 \\ & : \\ & -600 \end{aligned}$ | $\begin{aligned} & \text { FEOB } \\ & : \\ & \text { FDA8 } \end{aligned}$ |
| Underflow | $<-60.0{ }^{\circ} \mathrm{C}$ | $<-60.0{ }^{\circ} \mathrm{C}$ | $<-60.0^{\circ} \mathrm{C}$ | -32768 | 8000 |

## Output ranges voltage and current

| Range | -10 V ... +10 V | $\begin{aligned} & 0 \mathrm{~mA} \ldots 20 \\ & \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 4 \mathrm{~mA} \ldots 20 \\ & \mathrm{~mA} \end{aligned}$ | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  | Decimal | Hex. |
| Overflow | 0 V | 0 mA | 0 mA | > 32511 | > 7EFF |
| Value too high | $\begin{aligned} & 11.7589 \mathrm{~V} \\ & : \\ & 10.0004 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 23.5178 \mathrm{~mA} \\ & : \\ & 20.0007 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 22.8142 \mathrm{~mA} \\ & : \\ & 20.0006 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 32511 \\ & : \\ & 27649 \end{aligned}$ | $\begin{aligned} & \text { 7EFF } \\ & : \\ & 6 \mathrm{C} 01 \end{aligned}$ |
| Normal range | $\begin{aligned} & 10.0000 \mathrm{~V} \\ & : \\ & 0.0004 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 20.0000 \mathrm{~mA} \\ & : \\ & 0.0007 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 20.0000 \mathrm{~mA} \\ & : \\ & 4.0006 \mathrm{~mA} \end{aligned}$ | $27648$ <br> 1 | $\begin{aligned} & \text { 6C00 } \\ & : \\ & 0001 \end{aligned}$ |
|  | 0.0000 V | 0.0000 mA | 4.0000 mA | 0 | 0000 |
|  | $\begin{aligned} & -0.0004 \mathrm{~V} \\ & : \\ & -10.0000 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 0 \mathrm{~mA} \\ & : \\ & 0 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 3.9994 \mathrm{~mA} \\ & 0 \mathrm{~mA} \\ & 0 \mathrm{~mA} \end{aligned}$ | $\begin{array}{\|l} \hline-1 \\ -6912 \\ -27648 \end{array}$ | $\begin{aligned} & \text { FFFF } \\ & \text { E500 } \\ & 9400 \end{aligned}$ |
| Value too low | $\begin{aligned} & -10.0004 \mathrm{~V} \\ & : \\ & -11.7589 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 0 \mathrm{~mA} \\ & : \\ & 0 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 0 \mathrm{~mA} \\ & : \\ & 0 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & -27649 \\ & : \\ & -32512 \end{aligned}$ | $\begin{aligned} & \text { 93FF } \\ & : \\ & 8100 \end{aligned}$ |
| Underflow | 0 V | 0 mA | 0 mA | $<-32512$ | <8100 |

The represented resolution corresponds to 16 bits.

## Technical data

## Technical data of the module

The system data of AC500 and S500 are applicable to the standard version ${ }^{*} \Rightarrow$ Chapter 5.1.2 "System data AC500" on page 166.

The system data of AC500-XC are applicable to the XC version $\stackrel{y}{ }{ }^{\circ}$ Chapter 5.1.3 "System data AC500-XC" on page 169.

Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter |  | Value |
| :--- | :--- | :--- |
| Process supply voltage |  |  |
|  | Connections | Terminals $1.8,2.8,3.8$ and 4.8 for UP $(+24$ <br> V DC) and $1.9,2.9,3.9$ and 4.9 for ZP $(0$ |
|  | Protection against reverse voltage | yes |
|  | Rated protection fuse at UP | 10 A fast |
|  | Rated value | 24 V DC |
|  | Max. ripple | $5 \%$ |
| Current consumption | $0.07 \mathrm{~A}+$ max. 0.5 A per output |  |
|  | From UP | ca. 2 mA |
|  | From 24 V DC power supply at the termi- <br> nals UP/L+ and ZP/M of the CPU/communi- <br> cation interface module | $0.04 \mathrm{~A}^{2} \mathrm{~s}$ |
|  | Inrush current from UP (at power-up) | Yes, per module |
| Galvanic isolation | 6 W (outputs unloaded) |  |
| Max. power dissipation within the module | ca. 125 g |  |
| Weight (without terminal unit) | Horizontal mounting or vertical with <br> derating (output load reduced to 50\% at <br> +40 |  |
| Mounting position | The natural convection cooling must not be <br> hindered by cable ducts or other parts in <br> the control cabinet. |  |

## NOTICE!

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and temporary overvoltage up to 30 V DC.

## Multiple overloads

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an internal smart high-side switch.

## Technical data of the digital outputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 16 outputs (with transistors) |
| Distribution of the channels into groups | 1 group of 16 channels |
| Connection of the channels |  |
| DO0 ... DO7 | Terminals 1.0 ... 1.7 |
| DO8 ... DO15 | Terminals 2.0 ... 2.7 |
| Indication of the output signals | 1 yellow LED per channel, the LED is ON if the output signal is high (signal 1) |
| Monitoring point of output indicator | LED is controlled by process CPU |
| Reference potential for all outputs | Terminals 1.9, 2.9, 3.9 and 4.9 (negative pole of the process supply voltage, signal name ZP) |
| Common power supply voltage | For all outputs: terminals 1.8, 2.8, 3.8 and 4.8 (positive pole of the process supply voltage, signal name UP) |
| Output voltage for signal 1 | UP (-0.8 V) |
| Output delay (0->1 or 1->0) | On request |
| Output current |  |
| Rated value, per channel | 500 mA at $\mathrm{UP}=24 \mathrm{~V}$ |
| Maximum value (channels O0 to O15) | 4 A |
| Leakage current with signal 0 | $<0.5 \mathrm{~mA}$ |
| Rated protection fuse on UP | 10 A fast |
| Demagnetization when inductive loads are switched off | With varistors integrated in the module (see figure below) |
| Switching frequency |  |
| With resistive load | On request |
| With inductive loads | Max. 0.5 Hz |
| With lamp loads | Max. 11 Hz with max. 5 W |
| Short-circuit-proof / overload-proof | Yes |
| Overload message ( $1>0.7 \mathrm{~A}$ ) | Yes, after ca. 100 ms |
| Output current limitation | Yes, automatic reactivation after short circuit/overload |
| Resistance to feedback against 24 V signals | Yes |
| Max. cable length |  |
| Shielded | 1000 m |
| Unshielded | 600 m |

## Technical data of the configurable digital inputs/outputs

Each of the configurable digital I/O channels can be defined as input or output by the user program. This is done by interrogating or allocating the corresponding channel.

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 8 inputs/outputs (with transistors) |
| Distribution of the channels into groups | 1 group for 8 channels |
| If the channels are used as inputs |  |
| Channels DC16 ... DC23 |  |
| If the channels are used as outputs | Terminals $4.0 \ldots 4.7$ |
| Channels DC16 ... DC23 |  |
| Indication of the input/output signals | 1 yerminals $4.0 ~ . . . ~ 4.7 ~$ <br> when the input/output signal is high (signal 1) |
| Monitoring point of input/output indicator | LED is part of the input circuitry |
| Galvanic isolation | Yes, per module |

## Technical data of the digital inputs/outputs if used as inputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 8 |
| Distribution of the channels into groups | 1 group of 8 channels |
| Terminals of the channels DC16 ... DC23 | Terminals 4.0 ... 4.7 |
| Reference potential for all inputs | Terminals 1.9 ... 4.9 (Negative pole of the supply voltage, signal name ZP) |
| Indication of the input signals | 1 yellow LED per channel, the LED is ON when the input signal is high (signal 1) |
| Monitoring point of input/output indicator | LED is part of the input circuitry |
| Input type (according EN 61131-2) | Type 1 |
| Input delay (0->1 or 1->0) | Typ. 0.1 ms , configurable from 0.1 ms ... 32 ms |
| Input signal voltage | 24 V DC |
| 0-Signal | -3V ... +5 V |
| Undefined Signal | > +5V ... < +15 V |
| 1-Signal | +15V ... +30 V |
| Ripple with signal 0 | Within $-3 \vee \ldots+5 \mathrm{~V}$ |
| Ripple with signal 1 | Within +15 V ... +30 V |
| Input current per channel |  |
| Input voltage +24 V | Typ. 5 mA |
| Input voltage +5 V | > 1 mA |
| Input voltage +15 V | $>2 \mathrm{~mA}$ |
| Input voltage +30 V | $<8 \mathrm{~mA}$ |
| Max. cable length |  |
| Shielded | 1000 m |
| Unshielded | 600 m |

* Due to the direct connection to the output, the demagnetizing varistor is also effective at the input (see figure) above. This is why the difference between UPx and the input signal must not exceed the clamp voltage of the varistor. The varistor limits the clamp voltage to approx. 36 V . Consequently, the input voltage must range from $-12 \mathrm{~V} \ldots+30 \mathrm{~V}$ when $\mathrm{UPx}=24 \mathrm{~V}$ and from $-6 \mathrm{~V} \ldots+30 \mathrm{~V}$ when UPx $=30 \mathrm{~V}$.


## Technical data of the digital inputs/outputs if used as outputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 8 |
| Distribution of the channels into groups | 1 group of 8 channels |
| Terminals of the channels DC16 ... DC23 | Terminals 4.0 ... 4.7 |
| Reference potential for all outputs | Terminals $1.9 \ldots 4.9$ (negative pole of the supply voltage, signal name ZP) |
| Common power supply voltage | For all outputs terminals 1.8, 2.8, 3.8 and 4.8 (positive pole of the supply voltage, signal name UP) |
| Output voltage for signal 1 | UP (-0.8 V) |
| Output delay (0->1 or 1->0) | On request |
| Output current |  |
| rated value per channel | 500 mA at $\mathrm{UP}=24 \mathrm{~V}$ |
| max. value (all channels together) | 4 A |
| Leakage current with signal 0 | $<0.5 \mathrm{~mA}$ |
| Fuse for UP | 10 A fast |
| Demagnetization with inductive DC load | Via internal varistors (see figure below this table) |
| Output switching frequency |  |
| With resistive load | On request |
| With inductive loads | Max. 0.5 Hz |
| With lamp loads | 11 Hz max. at 5 W max. |
| Short-circuit-proof / overload-proof | Yes |
| Overload message ( $1>0.7 \mathrm{~A}$ ) | Yes, after ca. 100 ms |
| Output current limitation | Yes, automatic reactivation after short circuit/overload |
| Resistance to feedback against 24 V signals | Yes (software-controlled supervision) |
| Max. cable length |  |
| Shielded | 1000 m |
| Unshielded | 600 m |

The following drawing shows the circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off.


Fig. 165: Digital input/output (circuit diagram)
1 Digital input/output
2 For demagnetization when inductive loads are turned off

## Technical data of the fast counter

The fast counter of the module does not work if the module is connected to a CS31 bus module.

| Parameter | Value |
| :--- | :--- |
| Counting frequency | Max. 50 kHz |

« Chapter 6.8.2.12 "Fast counters in AC500 devices" on page 4536

## Technical data of the analog inputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 4 |
| Distribution of channels into groups | 1 group with 4 channels |
| Connection if channels $\mathrm{AlO}+\ldots \mathrm{Al3+}$ | Terminals 3.0 ... 3.3 |
| Reference potential for $\mathrm{AlO}+\ldots \mathrm{Al} 3+$ | Terminal 3.4 (AI-) for voltage and RTD measurement <br> Terminal 1.9, 2.9, 3.9 and 4.9 for current measurement |
| Input type |  |
| Unipolar | Voltage 0 V ... 10 V, current or Pt100/Pt1000/ Ni1000 |
| Bipolar | Voltage -10 V ... +10 V |
| Configurability | 0 V ... $10 \mathrm{~V},-10 \mathrm{~V} \ldots+10 \mathrm{~V}, 0 \mathrm{~mA} . . .20 \mathrm{~mA}$, 4 mA ... $20 \mathrm{~mA}, \mathrm{Pt} 100 / 1000$, Ni1000 (each input can be configured individually) |
| Channel input resistance | Voltage: > $100 \mathrm{k} \Omega$ Current: ca. $330 \Omega$ |
| Time constant of the input filter | Voltage: $100 \mu \mathrm{~s}$ Current: $100 \mu \mathrm{~s}$ |
| Indication of the input signals | 1 LED per channel (brightness depends on the value of the analog signal) |
| Conversion cycle | 1 ms (for 4 inputs + 2 outputs); with RTDs Pt/ Ni... 1 s |
| Resolution | Range 0 V ... 10 V : 12 bits <br> Range $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ : 12 bits including sign <br> Range $0 \mathrm{~mA} . . .20 \mathrm{~mA}$ : 12 bits <br> Range $4 \mathrm{~mA} . . .20 \mathrm{~mA}$ : 12 bits <br> Range RTD (Pt100, PT1000, Ni1000): $+0.1^{\circ} \mathrm{C}$ |
| Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range | Typ. 0.5 \%, max. 1 \% <br> For XC version below $0^{\circ} \mathrm{C}$ and above $+60^{\circ} \mathrm{C}$ : on request |


| Parameter | Value |
| :--- | :--- |
| Relationship between input signal and hex <br> code | r Chapter 5.2.4.5.1.2.10.2 "Input ranges <br> resistance temperature detector" on page 898 |
| Unused inputs | Are configured as "unused" (default value) |
| Overvoltage protection | Yes |

## Technical data of the analog inputs, if used as digital inputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | Max. 4 |
| Distribution of channels into groups | 1 group of 4 channels |
| Connections of the channels AI0+ ... Al3+ | Terminals 3.0... 3.3 |
| Reference potential for the inputs | Terminals 1.9, 2.9, 3.9 and 4.9 (ZP) |
| Indication of the input signals | 1 LED per channel |
| Input signal voltage | 24 V DC |
|  | Signal 0 |
|  | Undefined signal |
|  | Signal 1 |
| Input current per channel | $+5 \mathrm{~V} \ldots+13 \mathrm{~V}$ |
|  | Input voltage +24 V |
|  | $+13 \mathrm{~V} \ldots+30 \mathrm{~V}$ |
|  | Input voltage +5 V |
| Input voltage +15 V | Typ. 7 mA |
|  | Input voltage +30 V |
| Input resistance | Typ. 1.4 mA |

## Technical data of the analog outputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 2 |
| Distribution of channels into groups | 1 group for 2 channels |
| Connection of the channels $\mathrm{AO} 0+\ldots \mathrm{AO1+}$ | Terminals 3.5 and 3.6 |
| Reference potential for $\mathrm{AO}+\ldots \mathrm{AO} 1+$ | Terminal 3.7 (AO-) for voltage output <br> Terminals $1.9,2.9,3.9$ and 4.9 for current <br> output |
| Output type | Current |
|  | Unipolar |
| Bipolar | Voltage |
| Galvanic isolation | Against internal supply and other modules |
| Configurability | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}, 0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$, <br> $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}(e a c h ~ o u t p u t ~ c a n ~ b e ~ c o n f i g-~$ <br> ured individually) |


| Parameter | Value |
| :--- | :--- |
| Output resistance (load), <br> as current output | $0 \Omega \ldots 500 \Omega$ |
| Output loadability, <br> as voltage output | $\pm 10 \mathrm{~mA}$ max. |
| Indication of the output signals | 1 LED per channel (brightness depends on <br> the value of the analog signal) |
| Resolution | 12 bits including sign |
| Settling time for full range change (resistive <br> load, output signal within specified tolerance) | Typ. 5 ms |
| Conversion error of the analog values caused <br> by non-linearity, adjustment error at factory <br> and resolution within the normal range | Typ. 0.5 \%, max. 1 \% |
| Relationship between input signal and hex <br> code | « Chapter 5.2.4.5.1.2.10.3 "Output ranges <br> voltage and current" on page 899 |
| Unused outputs | Are configured as "unused" (default value) <br> and can be left open-circuited |

## Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm


## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 250 800 R0001 | DA502, digital/analog input/output <br> module, 16 DO, 8 DC, 4 AI, 2 AO | Active |
| 1SAP 450 800 R0001 | DA502-XC, digital/analog input/output <br> module, 16 DO, 8 DC, 4 AI, 2 AO, <br> XC version | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.4.6 Function modules

5.2.4.6.1 S500 and S500-XC

CD522 - Encoder, counter and PWM module

## Features

- 2 encoder inputs with 2 integrated 5 -V-power-supplies for the encoders
- 2 PWM outputs -2 digital inputs 24 V DC
- 8 configurable digital inputs/outputs 24 V DC
- Fast counter
- Module-wise galvanically isolated
- XC version for use in extreme ambient conditions available


1 I/O bus
2 Allocation of terminal No. and signal name
33 yellow LEDs to display the signal states of the encoder 0 input
43 yellow LEDs to display the signal states of the encoder 1 input
2 green LEDs to display the 5-V-power-supply states
2 yellow LEDs to display the signal state of the digital input I3 and I11 8 yellow LEDs to display the input/output signal states
2 yellow LEDs to display the signal states of the PWM/pulse outputs
1 green LED to display the process voltage UP
103 red LEDs to display errors
11 Label
12 Terminal unit
13 DIN rail
${ }_{x}^{x_{k}+\underbrace{}_{k}}$ Sign for XC version

## Intended purpose

The encoder and PWM module CD522 can be used at the following devices:

- Communication interface modules (e. g. CI501-PNIO, CI541-DP)
- Processor modules

Features:

- 2 independent counting functions with up to 12 configurable modes (including incremental position encoder and frequency input up to 300 kHz )
- 2 independent PWM (pulse-width modulator) or pulse outputs with push-pull driver
- Dedicated inputs/outputs for specific counting functions (e.g. touch, set, reset)
- All unused inputs/outputs can be used with the specifications of standard inputs/outputs range
For use in extreme ambient conditions (e.g. wider temperature and humidity range), a special XC version of the device is available.

Depending on the configuration used, some inputs and outputs are dedicated to specific counting functions (touch, set, reset...). All unused inputs and outputs can be used with the specification of standard inputs/outputs range.

## Functionality

| Digital inputs/outputs | 24 V DC, dedicated inputs/outputs can be used for specific <br> counting functions: <br> - Catch/touch operation, counter value stored in separate vari- <br> able on external event (rising or falling edge) <br> - Set input to preset counter register with predefined value <br> - -Set input to reset counter register <br> - - End value output; the output is set when predefined value is <br> reached <br> - Reference point initialization (RPI) input for incremental <br> encoder initialization <br> All unused inputs/outputs can be used with the specification of <br> standard input/output range. <br> Effect of incorrect input terminal connection: Wrong or no <br> signal detected, no damage up to 35 V. |
| :--- | :--- |
| Fast counter/encoder | integrated, 2 counters (hardware interface with +24 V DC, <br> +5 V DC, differential and 1 Vpp sinus input) with up to 12 <br> configurable operation modes: <br> -32 bits one counter mode |
|  | -16 bits two counter mode <br> - - Incremental position encoder <br> - Absolute SSI encoder <br> - Time frequency meter <br> - Frequency input up to 300 kHz |
|  |  |


| PWM/pulse outputs | 2 pulse-width-modulators or pulse outputs <br> Output specification <br> - Push-pull output: 24 V DC, 100 mA max. <br> - Current limitation (thermal and over current) <br> PWM specification <br> - Frequency from 1 Hz to 100 kHz <br> - Value from 0 to 100 \% <br> Pulse specification <br> - Frequency from 1 Hz to 15 kHz <br> - Pulse emission from 1 to 65535 pulses <br> - Number of pulses emitted indicator (0 to $100 \%$ ) <br> Frequency specification <br> - Frequency output = 100 kHz when duty cycle set to $50 \%$ |
| :---: | :---: |
| Power supply for encoders | 25 V power supplies, max. 100 mA |
| LED displays | For signal states, errors and supply voltage |
| Internal power supply | Via I/O bus |
| External power supply | Via the terminals UP (process voltage 24 V DC ) and $\mathrm{ZP}(0 \mathrm{~V}$ DC) |
| Required terminal unit | TU515 or TU516 Chapter 5.2.5.2 "TU515, TU516, TU541 and TU542 for I/O modules" on page 938 |

How to prepare a device as fast counter and how to connect it to the PLC is described in an application example.

## Connections

The function module CD522 can be connected to the following devices via the I/O bus connector:

- CS31 bus module DC551-CS31
- AC500 CPU
- OtherAC500 I/O devices.

The connection is carried out by using the 40 terminals of the terminal unit TU515/TU516 \# $\downarrow$ Chapter 5.2.5.2 "TU515, TU516, TU541 and TU542 for I/O modules" on page 938.

Table 206: Assignment of the terminals

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 1.0 | IA0 | Inverted input signal A of encoder 0 |
| 1.1 | IB0 | Inverted input signal B of encoder 0 |
| 1.2 | IZ0 | Inverted input signal Z of encoder 0 |
| 1.3 | 5 V 0 | +5 V DC power supply output 0 for sensors |
| 1.4 | 0 V | 0 V reference input |
| 1.5 | O0 | Output signal of the fast output O0 |
| 1.6 | OV | 0 V reference input |
| 1.7 | O 1 | Output signal of the fast output O1 |


| Terminal | Signal | Description |
| :---: | :---: | :---: |
| 1.8 | UP | Process voltage UP (24 V DC) |
| 1.9 | ZP | Process voltage ZP (0 V DC) |
| 2.0 | A0 | Input signal A of encoder 0 |
| 2.1 | B0 | Input signal B of encoder 0 |
| 2.2 | Z0 | Input signal $Z$ of encoder 0 |
| 2.3 | 13 | Input signal I3 (standard input) |
| $2.4 \ldots 2.7$ | C4 ... C7 | Signal of the configurable digital input/output C4 ... C7 |
| 2.8 | UP | Process voltage UP (24 V DC) |
| 2.9 | ZP | Process voltage ZP (0 V DC) |
| 3.0 | /A1 | Inverted input signal A of encoder 1 |
| 3.1 | /B1 | Inverted input signal B of encoder 1 |
| 3.2 | IZ1 | Inverted input signal $Z$ of encoder 1 |
| 3.3 | 5V1 | +5 V DC power supply output 1 for sensors |
| 3.4...3.7 | OV | 0 V reference input |
| 3.8 | UP | Process voltage UP (24 V DC) |
| 3.9 | ZP | Process voltage ZP (0 V DC) |
| 4.0 | A1 | Input signal A of encoder 1 |
| 4.1 | B1 | Input signal B of encoder 1 |
| 4.2 | Z1 | Input signal $Z$ of encoder 1 |
| 4.3 | 111 | Input signal I11 (standard input) |
| $4.4 \ldots 4.7$ | C12 ... C15 | Signal of the configurable digital input/output C12 ... C15 |
| 4.8 | UP | Process voltage UP (24 V DC) |
| 4.9 | ZP | Process voltage ZP (0 V DC) |

The internal power supply voltage for the module's circuitry is carried out via the I/O bus (provided by a communication interface module or a processor module). Thus, the current consumption from $24 \vee$ DC power supply at the terminals L+/UP and M/ZP of the CPU/communication interface module increases by 2 mA per CD522.
The external power supply connection is carried out via the UP (+24 V DC) and the ZP (0 V DC) terminals.

## WARNING!

## Removal/Insertion under power

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.
Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.
The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

Risk of damaging the PLC modules!
Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

Connection of encoders with differential RS-422 signal

The encoder is powered by the 5 V power supply which is integrated in CD522.


Connection of The encoder is powered by the 5 V power supply which is integrated in the CD522. encoders with 5 V TTL signal


The wires $A, B$ and $Z$ need not to be connected to the module. They are left open.

When using different power supplies for the encoder device and the CD522, make sure that the reference potentials of both power supplies are interconnected.

Connection of encoders with 1 Vpp sine signal

The encoder is powered through the 5 V power supply which is integrated in the CD522.


Connection of absolute encoders with SSI interface and differential RS-422 signal

The encoder is powered by the 5 V power supply which is integrated in the CD522.


Connection of absolute encoders with an SSI interface and an optocoupler interface at CLK input

The encoder can optionally be powered by the 5 V power supply which is integrated in the CD522.


Connection of output loads to the PWM/Pulse putputs


## NOTICE!

Risk of damaging the module
The PWM outputs have no protection against reverse polarity.

Connection of Proceed with the inputs/outputs 111 and $\mathrm{C} 12 \ldots \mathrm{C} 15$ in the same way. standard inputs/ outputs


[^15]

Fig. 166: Example of the connection of sensors with frequency outputs to the input ZO of the CD522

## - NOTICE!

Risk of malfunctions!
The edges of a signal must be strong enough ( $0.4 \mathrm{~V} / \mu \mathrm{s}$ ) to be recognized correctly by the module.

Put a $1 \mathrm{k} \Omega$ resistor between 0 V and the Z terminal when using a standard output as time generator.

Connection of $\quad$ Proceed with the 5 V power supply 1 in the same way.
$\qquad$
Each 5-V-power supply provides a current of 100 mA max. It is possible to parallel both integrated power supplies. In this case, the max. current is 200 $m A$.


## - NOTICE! <br> Risk of damaging the module

The integrated 2 A fuse cannot be replaced. If it blows, the module must be replaced.
Ensure that the current per 0 V connection does not exceed 0.5 A .

## NOTICE!

## Risk of damaging the module

The two 5 V outputs have no protection against reverse polarity.

## Internal data exchange

| Parameter | Value |
| :--- | :--- |
| Digital inputs (bytes) | 0 |
| Digital outputs (bytes) | 0 |
| Analog inputs (words) | 12 |
| Analog outputs (words) | 16 |

The data will be transferred in 16-bit words and not in bytes. Two bytes are packed into one 16-bit word.
The bit strings are transmitted in big-endian byte order, so the bytes within the word are swapped. If several bytes are considered, the first byte (lowest address) is the largest (High Byte).

If used with AC500 please check the chapter how to configure CD522 within Automation Builder and use the CD522 library.

The types "structCD522In" and "structCD522Out" can be added in Automation Builder by using the command "Generate DUT" via the context menu of the CD522 device.

To use CD522 with CI50x-PNIO or CI54x-DP as unbundled IOs with other PLCs find the meaning of the IO-Data in following tables below:

## From CD522 to PLC

| TYPE structCD522In |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| STRUCT |  |  |  |  |  |
| StateBytePWM0 | High Byte | $0 . .100$ | Percentage of pulses already sent on channel 0 |  |  |
| StateBytePWM1 | Low Byte | $0 . .100$ | Percentage of pulses already sent on channel 1 |  |  |
| StateByteC0 | High Byte |  | *) |  |  |
| InputC0 | Low Byte |  | Name | Bitposition | Description |
|  |  |  | Input A | 0 | Digital Input A |
|  |  |  | Input B | 1 | Digital Input B |
|  |  |  | Input Z | 2 | Digital Input C |
|  |  |  | Input I3 | 3 | Digital Input I3 |
|  |  |  | Input I4 | 4 | Digital Input 14 |
|  |  |  | Input I5 | 5 | Digital Input I5 |
|  |  |  | Input I6 | 6 | Digital Input I6 |
|  |  |  | Input I7 | 7 | Digital Input I7 |
| Touch-CounterHiCO | WORD |  |  |  |  |
| Touch-CounterLoC0 | WORD |  |  |  |  |
| CounterHiC0 | WORD |  |  |  |  |
| CounterLoC0 | WORD |  |  |  |  |
| CounterHiC1 | WORD |  |  |  |  |
| CounterLoC1 | WORD |  |  |  |  |
| ReservedWC1 | WORD |  |  |  |  |
| StateByteC1 | High Byte |  | *) |  |  |
| InputC1 | Low Byte |  | Name | Bitposition | Description |
|  |  |  | Input A | 0 | Digital Input A |
|  |  |  | Input B | 1 | Digital Input B |
|  |  |  | Input Z | 2 | Digital Input C |
|  |  |  | Input I3 | 3 | Digital Input I3 |
|  |  |  | Input I4 | 4 | Digital Input 14 |
|  |  |  | Input I5 | 5 | Digital Input I5 |

## TYPE structCD522In

|  |  |  | Input I6 | 6 | Digital Input I6 |
| :--- | :--- | :--- | :--- | :--- | :--- |
|  |  |  | Input I7 | Digital Input I7 |  |
| Touch- <br> Coun- <br> terHiC1 | WORD |  |  |  |  |
| Touch- <br> Coun- <br> terLoC1 | WORD |  |  |  |  |
| END_STRUCT |  |  |  |  |  |
| END_TYPE |  |  |  |  |  |

Table 207: *) Status Byte C0/C1

| Bit | One Counter Modes (1,2,5,6,11,12,13,14) | Two Counter Modes (3,4) |
| :--- | :--- | :--- |
| 0 | CF0 <br> 1=End value 0 reached | not used |
| 1 | not used | not used |
| 2 | NCATCH <br> $1=$ New catch available | not used |
| $1=$ Counter 0 overflow (see Note 3 below) |  |  |
| 3 | SET0_INPUT <br> Logical OR on all inputs configured as set0 input | OVRFLW1 <br> $1=$ Counter 1 overflow (0x0000 <br> $\leftarrow \rightarrow$ 0xFFFF) |
| 4 | RESET0_INPUT <br> Logical OR on all inputs configured as reset0 input <br> $\leftarrow \rightarrow$ OxFFFF) |  |
| 5 | not used | RESET0_INPUT <br> Logical OR on all inputs con- <br> figured as reset0 input |
| 6 | not used | not used |
| 7 |  | RESET1_INPUT <br> Logical OR on all inputs con- <br> figured as reset1 input |


| Bit | 16-bit One Counter Mode (8) | Time frequency meter mode <br> $(15)$ |
| :--- | :--- | :--- |
| 0 | CF0 <br> $1=$ Zero crossover detected | not used |
| 1 | not used | not used |
| 2 | NCATCH <br> $1=$ New catch available | not used |
| 3 | not used | not used |
| 4 | SETO_INPUT <br> Logical OR on all inputs configured as set0 input | not used |


| Bit | 16-bit One Counter Mode (8) | Time frequency meter mode <br> $(15)$ |
| :--- | :--- | :--- |
| 5 | RESETO_INPUT <br> Logical OR on all inputs configured as reset0 input | not used |
| 6 | not used | NEW <br> $1=$ New timing value available |
| 7 | not used | not used |

## From PLC to CD522

| TYPE structCD522Out |  |  |  |
| :---: | :---: | :---: | :---: |
| STRUCT |  |  |  |
| FreqPWM0 | WORD | 0... 65535 | PWM frequency of channel 0 <br> Unit: Hz or 10 Hz (depending on control byte in slot 3 ) <br> Limit: 100kHz |
| DutyPulsePWM0 | WORD | 0... 1000 | PWM mode: <br> PWM duty cycle of channel 0 in $1 / 10$ percentage |
|  |  | 0... 65535 | Pulse mode: <br> Number of pulses to sent on channel 0 |
| ControlPWM0 | High Byte | Bit | Description |
|  |  | 0 | FREQU_X10 <br> FREQU_X10 1 = Frequency multiplier $\times 10$ enabled |
|  |  | 1 | not used |
|  |  | 2 | not used |
|  |  | 3 | PULSE_START <br> Rising edge = Start pulse emission channel 0 |
|  |  | 4 | not used |
|  |  | 5 | not used |
|  |  | 6 | not used |
|  |  | 7 | 1 = Enable Pulse/PWM channel 0 |
| ReservedBPWM0 | $\begin{array}{\|l\|} \hline \text { Low } \\ \text { Byte } \\ \hline \end{array}$ |  |  |
| ReservedWPWM0 | WORD |  |  |
| FreqPWM1 | WORD | 0...65535 | PWM frequency of channel 1 <br> Unit: Hz or 10 Hz (depending on control byte in slot 3 ) Limit: 100kHz |
| $\begin{aligned} & \text { DutyPul- } \\ & \text { sePWM1 } \end{aligned}$ | WORD | 0... 1000 | PWM mode: <br> PWM duty cycle of channel 1 in $1 / 10$ percentage |
|  |  | 0... 65535 | Pulse mode: <br> Number of pulses to sent on channel 1 |
| ControlPWM1 | High <br> Byte | Bit | Description |


| TYPE structCD522Out |  |  |  |
| :---: | :---: | :---: | :---: |
|  |  | 0 | FREQU_X10 <br> 1 = Frequency multiplier x10 enabled |
|  |  | 1 | not used |
|  |  | 2 | not used |
|  |  | 3 | PULSE_START <br> Rising edge = Start pulse emission channel 1 |
|  |  | 4 | not used |
|  |  | 5 | not used |
|  |  | 6 | not used |
|  |  | 7 | 1 = Enable Pulse/PWM channel 1 |
| OutputPW0PWM1 | $\begin{aligned} & \text { Low } \\ & \text { Byte } \end{aligned}$ | Bit $0=$ Digital output value of channel 0 <br> BIT 4 = Digital output value of channel 1 |  |
| ReservedWPWM1 | WORD |  |  |
| CounterSetHiC0 | WORD |  |  |
| Counter- <br> SetLoC0 | WORD |  |  |
| CtrlByteC0 | High Byte |  | **) |
| OutputC0 | $\begin{array}{\|l\|} \hline \text { Low } \\ \text { Byte } \end{array}$ |  |  |
| ReservedWC0 | WORD |  |  |
| CounterSetHiC1 | WORD |  |  |
| CounterSetLoC1 | WORD |  |  |
| CtrlByteC1 | High Byte |  | **) |
| OutputC1 | $\begin{array}{\|l\|} \hline \text { Low } \\ \text { Byte } \end{array}$ |  |  |
| ReservedWC1 | WORD |  |  |
| END_STRUCT |  |  |  |
| END_TYPE |  |  |  |

Table 208: **) Counter Control Byte C0/C1

| Bit | One Counter Modes (1,2,5,6,8) | Two Counter Modes (3,4) |
| :--- | :--- | :--- |
| 0 | EN <br> $0=$ counter disabled <br> $1=$ counter enabled | EN <br> $0=$ counter disabled <br> $1=$ counter enabled |
| 1 | SET_0 <br> $1=$ set counter 0 | not used |


| Bit | One Counter Modes (1,2,5,6,8) | Two Counter Modes (3,4) |
| :--- | :--- | :--- |
| 2 | RESET_0 <br> $1=$ reset counter 0 | RESET_0 <br> $1=$ reset counter 0 |
| 3 | not used | UP_DWN0 <br> $0=$ =up counter 0 <br> $1=$ down counter 0 |
| 4 | not used | RESET_1 <br> $1=$ reset counter 1 |
| 5 | UPDWN <br> $0=$ up counter <br> $1=$ down counter | UP_DWN1 <br> $0=$ up counter 1 <br> $1=$ down counter 1 |
| 6 | NCATCH <br> $0=$ no catch operation <br> $1=$ enable next catch operation | NCATCH <br> $0=$ no catch operation <br> $1=$ enable next catch operation |
| 7 | EDGECATCH <br> $0=$ catch on falling edge <br> $1=$ catch on rising edge | EDGECATCH <br> $0=$ catch on falling edge <br> $1=$ catch on rising edge |


| Bit | Relative encoder modes (Modes 11,12,13) | Time frequency meter (Mode 15) |
| :--- | :--- | :--- |
| 0 | EN <br> $0=$ counter disabled <br> $1=$ counter enabled | EN <br> $0=$ counter disabled <br> $1=$ counter enabled |
| 1 | SET_0 <br> $1=$ set counter 0 | EN_1_0 <br> $1=$ enable time capture on falling <br> edge |
| 2 | RESET_0 <br> $1=$ reset counter 0 | EN_0_1 <br> $1=$ enable time capture on rising <br> edge |
| 3 | not used | FREQ_0 <br> $0=$ time measure mode <br> $1=$ frequency and RPM measure <br> mode |
| 4 | RPI, Reference Point Indicator | RESET_NEW <br> $1=$ time/frequency/RPM measure- <br> ment is in reset. NEW flag is cleared. |
| 5 | not used | not used |
| 6 | NCATCH <br> $0=$ no catch operation <br> $1=$ nable next catch operation | EDGECATCH <br> $0=$ catch on falling edge <br> $1=$ catch on rising edge |
| 7 |  | not used |


| Bit | SSI, absolute encoder (Mode 14) |
| :--- | :--- |
| 0 | EN <br> $0=$ counter disabled <br> $1=$ counter enabled |
| 1 | not used |
| 2 | not used |
| 3 | not used |
| 4 | not used |
| 5 | not used |
| 6 | NCATCH <br> $0=$ no catch operation <br> $1=$ enable next catch operation |
| 7 | EDGECATCH <br> $0=$ catch on falling edge <br> $1=$ catch on rising edge |

## I/O configuration

The module itself does not store configuration data. It receives its parameterization data from the master device of the I/O bus (CPU or communication interface module) during power-up of the system.
Hence, replacing I/O modules is possible without any re-parameterization via software.

> If the external power supply voltage via UP/ZP terminals fails, the I/O module loses its configuration data. The whole station has to be switched off and on again to re-configure the module.

## Parameterization

| Firmware version | Configuration |
| :--- | :--- |
| Firmware version > V2.0.0 | The arrangement of the parameter data is per- <br> formed by Control Builder Plus/ Automation <br> Builder software. |

The parameter data directly influences the functionality of modules.
For non-standard applications, it is necessary to adapt the parameters to your system configuration.

Module: Module slot address: $\mathrm{Y}=1$... 10

| Name | Value | Internal <br> value | Internal <br> value, <br> Type | Default | Min. | Max. | EDS Slot <br> Index |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Module ID | Intern | $1805^{1}$ ) | WORD | $0 x 070 D$ | 0 | 65535 | $0 x 0 Y 01$ |
| Ignore <br> module $^{2}$ ) | No <br> Yes | 0 | BYTE |  |  |  |  |
| 1 | No <br> $0 x 00$ |  |  | Not for <br> FBP |  |  |  |


| Name | Value | Internal value | Internal value, Type | Default | Min. | Max. | EDS Slot Index |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Parameter length | Internal | 42 | BYTE | 0 | 0 | 255 | xx02 ${ }^{3}$ ) |
| Check supply | $\begin{aligned} & \text { Off } \\ & \text { On } \end{aligned}$ | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | BYTE | $\begin{array}{\|l\|} \hline \text { On } \\ 0 \times 01 \end{array}$ |  |  | 0x0Y03 |
| Input delay | $\begin{aligned} & 0.1 \mathrm{~ms} \\ & 1 \mathrm{~ms} \\ & 8 \mathrm{~ms} \\ & 32 \mathrm{~ms} \end{aligned}$ | $\begin{aligned} & 0 \\ & 1 \\ & 2 \\ & 3 \end{aligned}$ | BYTE | $\begin{aligned} & 8 \mathrm{~ms} \\ & 0 \times 02 \end{aligned}$ | 0 | 3 | 0x0Y04 |
| Mode Counter 0 | see table below | 0 | BYTE | 0x00 | 0 | 15 | 0x0Y05 |
| Counter 0 frequency limit | No filter $\left\lvert\, \begin{aligned} & 50 \mathrm{~Hz} \\ & 500 \mathrm{~Hz} \\ & 5 \mathrm{kHz} \\ & 20 \mathrm{kHz} \end{aligned}\right.$ | $\begin{aligned} & 0 \\ & 1 \\ & 2 \\ & 3 \\ & 4 \end{aligned}$ | BYTE | No filter $0 \times 00$ | 0 | 4 | 0x0Y06 |
| Counter 0 input level | $\begin{aligned} & 0-24 \mathrm{~V} \text { DC } \\ & 0-5 \mathrm{~V} \text { DC } \\ & \text { Differen- } \\ & \text { tial } \\ & 1 \mathrm{Vpp} \\ & \text { sinus } \end{aligned}$ | $\begin{aligned} & 0 \\ & 1 \\ & 2 \\ & 3 \end{aligned}$ | BYTE | $\begin{aligned} & 0-24 \mathrm{~V} D C \\ & 0 \times 00 \end{aligned}$ | 0 | 3 | 0X0Y07 |
| SSI 0 frequency | $\begin{aligned} & 200 \mathrm{kHz} \\ & 500 \mathrm{kHz} \\ & 1 \mathrm{MHz} \end{aligned}$ | $\begin{aligned} & 2 \\ & 3 \\ & 4 \end{aligned}$ | BYTE | $\begin{aligned} & 200 \mathrm{kHz} \\ & 0 \times 02 \end{aligned}$ | 0 | 4 | 0x0Y08 |
| SSI 0 resolution (in bit) | 8 to 32 bit |  | BYTE | 16 bit 16 | 8 | 32 | 0x0Y09 |
| SSI 0 code type | Binary | 0 | BYTE | Binary 0 | 0 | 0 | 0x0YOA |
| SSI 0 polling time | 10 ms |  | BYTE | 10 | 1 | 255 | 0x0YOB |
| $\begin{array}{\|l} 5 \mathrm{~V} \\ \text { sensor } 0 \\ \text { supply } \\ \hline \end{array}$ | $\begin{aligned} & \text { Off } \\ & \text { On } \end{aligned}$ | 0 | BYTE | $\begin{aligned} & \text { Off } \\ & 0 \times 00 \end{aligned}$ | 0 | 1 | 0xOYOC |
| Mode Counter 1 | see table below | 0 | BYTE | 0x00 | 0 | 15 | 0x0YOD |
| Counter 1 frequency limit | No filter $\begin{aligned} & 50 \mathrm{~Hz} \\ & 500 \mathrm{~Hz} \\ & 5 \mathrm{kHz} \\ & 20 \mathrm{kHz} \end{aligned}$ | $\begin{aligned} & 0 \\ & 1 \\ & 2 \\ & 3 \\ & 4 \end{aligned}$ | BYTE | No filter $0 \times 00$ | 0 | 4 | 0xOYOE |


| Name | Value | Internal value | Internal value, Type | Default | Min. | Max. | EDS Slot Index |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Counter 1 input level | $\begin{aligned} & 0-24 \mathrm{~V} \text { DC } \\ & 0-5 \mathrm{~V} D \mathrm{C} \\ & \text { Differen- } \\ & \text { tial } \\ & 1 \mathrm{Vpp} \\ & \text { sinus } \end{aligned}$ | $\begin{aligned} & \hline 0 \\ & 1 \\ & 2 \\ & 3 \end{aligned}$ | BYTE | $\begin{aligned} & \text { 0-24 V DC } \\ & 0 \times 00 \end{aligned}$ | 0 | 3 | OXOYOF |
| SSI 1 frequency | $\begin{aligned} & 200 \mathrm{kHz} \\ & 500 \mathrm{kHz} \\ & 1 \mathrm{MHz} \end{aligned}$ | $\begin{aligned} & 2 \\ & 3 \\ & 4 \end{aligned}$ | BYTE | $\begin{aligned} & 200 \mathrm{kHz} \\ & 0 \times 02 \end{aligned}$ | 2 | 4 | 0x0Y10 |
| SSI 1 resolution (in bit) | 8 to 32 bit |  | BYTE | $\begin{aligned} & 16 \text { bit } \\ & 16 \end{aligned}$ | 8 | 32 | 0x0Y11 |
| SSI 1 code type | Binary | 0 | BYTE | Binary 0 | 0 | 0 | 0x0Y12 |
| SSI 1 polling time | 10 ms |  | BYTE | 10 | 1 | 255 | 0x0Y13 |
| $\begin{array}{\|l} 5 \mathrm{~V} \\ \text { sensor } 1 \\ \text { supply } \end{array}$ | $\begin{aligned} & \text { Off } \\ & \text { On } \end{aligned}$ | 0 | BYTE | $\begin{aligned} & \hline \text { Off } \\ & 0 \times 00 \end{aligned}$ | 0 | 1 | 0x0Y14 |
| Detection SC on sensors | $\begin{aligned} & \text { Off } \\ & \text { On } \end{aligned}$ | 0 | BYTE | $\begin{aligned} & \hline \text { Off } \\ & 0 \times 00 \end{aligned}$ | 0 | 1 | 0x0Y15 |
| Output behaviour com fault | Off <br> Last value <br> Substitute <br> Last value 5s <br> Substitute 5s <br> Last value 10s Substitute 10s | $\begin{aligned} & \hline 0 \\ & 1 \\ & 2 \\ & 3 \\ & 4 \\ & 5 \\ & 6 \end{aligned}$ | BYTE | $\begin{aligned} & \hline \text { Off } \\ & 0 \times 00 \end{aligned}$ | 0 | 1 | 0x0Y16 |
| Substitute value | 0 | 0 | WORD | Default 0x0000 | 0 | 65536 | $0 \times 0 Y 17$ |

${ }^{1}$ ) With CS31 and addresses smaller than 70 and FBP, the value is increased by 1
${ }^{2}$ ) Not with FBP
${ }^{3}$ ) Value is hexadecimal: HighByte is slot ( $x x$ : $1 \ldots 10$ ), LowByte is index (1 ... n)

Table 209: Operating modes for counters 0 and 1, configuration table

| Internal value | Operating modes of counter |
| :--- | :--- |
| 0 | No counter / No PWM (default value) |
| 1 | $1-1$ UpDown counter (A) |
| 2 | $2-1$ UpDown with release input |


| Internal value | Operating modes of counter |
| :--- | :--- |
| 3 | $3-2$ UpDown counters (A, B) |
| 4 | $4-2$ UpDown (A, B on falling edges) |
| 5 | $5-1$ UpDown dynamic set (B) / rising edge |
| 6 | $6-1$ UpDown dynamic set (B) / falling edge |
| 7 | Not used |
| 8 | $8-1$ UpDown with release (B), 0 cross detection |
| $9-19$ | Not used |
| 20 | $11-1$ Incremental encoder |
| 21 | $12-2$ Incremental encoder X2 |
| 22 | $13-1$ Incremental encoder X4 |
| 30 | $14-1$ SSI, absolute encoder |
| 40 | $15-1$ Time frequency meter |

Table 210: GSD file

| Ext_User_Prm_Data_Len $=$ |  |
| :--- | :--- |
| Ext_User_Prm_Data_Const $(0)=$ | 25 |
|  | $0 \times 07,0 \times 0 \mathrm{E}, 0 \times 17,1$ |
|  | $0 \times 01,0 \times 02,1$ |
|  | $0 \times 00,0 \times 00,0 \times 00,0 \times 02,0 \times 10,0 \times 00,0 \times 0 \mathrm{~A}$, |
| $0 \times 00,1$ |  |
| $0 \times 00,0 \times 00,0 \times 00,0 \times 02,0 \times 10,0 \times 00,0 \times 0 \mathrm{~A}$, |  |
|  | $0 \times 00,1$ |
|  | $0 x 00,0 \times 00,0 \times 00,0 \times 00 ;$ |

## Diagnosis

| E1...E4 | d1 | d2 | d3 | d4 | Identi- <br> fier <br> $000 \ldots$ <br> 063 | AC500 display | <- Display in |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err | PS501 PLC browser |  |
| $\begin{aligned} & \hline \text { Byte } 6 \\ & \text { Bit } 6 \ldots \\ & 7 \\ & \hline \end{aligned}$ | - | Byte 3 | Byte 4 | Byte 5 | Byte 6 <br> Bit 0 ... 5 | FBP diagnosis block |  |
| Class | Interface | Device | Module | Channel | Error identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) | ${ }^{4}$ ) |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 19 | Checksum error | Replace I/O |
|  | 11 / 12 | ADR | 1... 10 |  |  | in the I/O module | e module |
| 3 | 14 | 1... 10 | 31 | 31 | 9 | Overflow diag- | New start |
|  | 11 / 12 | ADR | 1... 10 |  |  | nosis buffer |  |
| 3 | 14 | 1... 10 | 31 | 31 | 26 | Parameter error | Check master |


| E1...E4 | d1 | d2 | d3 | d4 | Identi- <br> fier <br> $000 \ldots$ <br> 063 | $\begin{array}{\|l} \hline \begin{array}{l} \text { AC500 dis- } \\ \text { play } \end{array} \\ \hline \end{array}$ | <- Display in |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err | PS501 PLC browser |  |
| $\begin{array}{\|l} \hline \text { Byte } 6 \\ \text { Bit } 6 \ldots . \\ 7 \end{array}$ | - | Byte 3 | Byte 4 | Byte 5 | $\left\|\begin{array}{l} \text { Byte } 6 \\ \text { Bit } 0 \ldots 5 \end{array}\right\|$ | FBP diagnosis block |  |
| Class | Interface | Device | Module | Channel | Error identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) | ${ }^{4}$ ) |  |  |  |
|  | 11 / 12 | ADR | 1... 10 |  |  |  |  |
| 3 | 14 | 1... 10 | 31 | 31 | 11 | Process voltage | Check |
|  | 11 / 12 | ADR | 1... 10 |  |  | too low | process voltage |

Table 211: Channel error CD522
$\left.\begin{array}{|l|l|l|l|l|l|l|l|l|}\hline \text { E1...E4 } & \text { d1 } & \text { d2 } & \text { d3 } & \text { d4 } & \begin{array}{l}\text { Identi- } \\ \text { fier } \\ \text { 000 } \ldots\end{array} \\ \text { 063 }\end{array}\right)$

| E1...E4 | d1 | d2 | d3 | d4 | Identi- <br> fier <br> 000 ... <br> 063 | AC500dis- <br> play | <- Display in |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Class | Comp | Dev | Mod | Ch | Err | PS501 PLC <br> browser |  |
| Byte 6 <br> Bit 6 ... <br> 7 | - | Byte 3 | Byte 4 | Byte 5 | Byte 6 <br> Bit 0 ... 5 | FBP diag- <br> nosis block |  |
| Class | Inter- <br> face | Device | Module | Channel | Error <br> identi- <br> fier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | $\left.{ }^{3}\right)$ | $\left.{ }^{4}\right)$ |  |  |  |
| 4 | 14 | $1 \ldots 10$ | 1 | 0,1 | 18 | Internal fuse on 0 <br> V has blown, 0 V <br> not connected to <br> GND | Check <br> wiring, <br> replace <br> module |

Remarks:

| ${ }^{1}$ ) | In AC500, the following interface identifier applies: $14=\mathrm{I} / \mathrm{O}$ bus, $11=$ COM1 (e.g. CS31 bus), $12=$ COM2. <br> The FBP diagnosis block does not contain this identifier. |
| :---: | :---: |
| ${ }^{2}$ ) | With "Device" the following allocation applies: <br> $31=$ module itself, $1 \ldots 10=$ decentralized communication interface module $1 . . .10$, ADR = hardware address (e.g. of the DC551) |
| ${ }^{3}$ ) | With "Module" the following allocation applies depending on the master: <br> Module error: I/O bus or FBP: 31 = module itself; COM1/COM2: $1 \ldots 10=$ expansion 1... 10 <br> Channel error: I/O bus or FBP = module type ( 2 = DO); COM1/COM2: $1 . .10=$ expansion 1... 10 |
| ${ }^{4}$ ) | In case of module errors, with channel "31 = Module itself" is output. |

## State LEDs

During the power-on procedure, the module initializes automatically. All LEDs (except the LEDs for the signal states) are on during the initialization.


## Technical data

The system data of AC500 and S500 are applicable to the standard version \& Chapter 5.1.2 "System data AC500" on page 166.

The system data of AC500-XC are applicable to the XC version ${ }^{〔}$ Chapter 5.1.3 "System data AC500-XC" on page 169.
Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter | Value |
| :---: | :---: |
| Process supply voltage |  |
| Connections | Terminals 1.8, 2.8, 3.8 and 4.8 for UP (+24 V DC) and 1.9, 2.9, 3.9 and 4.9 for ZP ( 0 V ) |
| Protection against reverse voltage | Yes |
| Rated protection fuse at UP | 10 A fast |
| Rated value | 24 V DC |
| Max. ripple | 5 \% |
| Current consumption |  |
| From UP | 0.07 A + max. 0.008 A per input + max. 0.5 A per output + 0.01 A for $A, B$ and $Z$ inputs |
| Via I/O bus | Ca. 5 mA |
| Inrush current from UP (at power-up) | $0.04 \mathrm{~A}^{2} \mathrm{~s}$ |
| Galvanic isolation | Yes, per module |
| Max. power dissipation within the module | 6 W (outputs unloaded) |
| Weight (without terminal unit) | Ca. 125 g |
| Mounting position | Horizontal mounting or vertical with derating (output load reduced to $50 \%$ at $+40^{\circ} \mathrm{C}$ ) |
| Cooling | The natural convection cooling must not be hindered by cable ducts or other parts in the control cabinet. |

## NOTICE!

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and temporary overvoltage up to 30 V DC.

## Multiple overloads

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an internal smart high-side switch.

| Technical data <br> of the digital <br> inputs/outputs if <br> used as <br> standard inputs | Parameter | Value |
| :--- | :--- | :--- |
|  | Number of channels | $2+8$ configurable digital inputs/outputs |
|  | Reference potential for all <br> inputs | Terminals $1.9 \ldots 4.9$ (negative pole of the process supply <br> voltage, signal name ZP) |
|  | Galvanic isolation | From the rest of the module |
|  |  |  |


| Parameter | Value |
| :--- | :--- |
| Indication of the input signals | 1 yellow LED per channel, the LED is ON when the input <br> signal is high (signal 1) |
| Input type acc. to EN 61131-2 | Type 1 |
| Input delay (0->1 or 1->0) | Typ. 8 ms , configurable from 0.1 to 32 ms |
| Input data length | 24 bytes |
| Input signal voltage | 24 V DC |
|  | Signal 0 |
|  | Undefined signal |
|  | Signal 1 |
| Ripple with signal 0 | $>+5 \mathrm{~V} \ldots+5 \mathrm{~V}$ * +15 V |
| Ripple with signal 1 | $+15 \mathrm{~V} \ldots+30 \mathrm{~V}$ |
| Input current per channel | Within $-3 \mathrm{~V} \ldots+5 \mathrm{~V}$ * |
|  | Input voltage +24 V |
|  | Input voltage +5 V |
|  | Input voltage +15 V |
|  | Typ. 5 mA |
| Input voltage +30 V | $>5 \mathrm{~mA}$ |
| Max. cable length | $<8 \mathrm{~mA}$ |
|  | Shielded |
|  | Unshielded |

* Due to the direct connection to the output, the demagnetizing varistor is also effective at the input (see figure) above. This is why the difference between UPx and the input signal must not exceed the clamp voltage of the varistor. The varistor limits the clamp voltage to approx. 36 V . Consequently, the input voltage must range from $-12 \mathrm{~V} \ldots+30 \mathrm{~V}$ when $\mathrm{UPx}=24 \mathrm{~V}$ and from $-6 \mathrm{~V} \ldots+30 \mathrm{~V}$ when $\mathrm{UPx}=30 \mathrm{~V}$.

Technical data of the digital inputs/outputs if used as standard outputs

| Parameter | Value |
| :--- | :--- |
| Number of channels | 8 configurable digital inputs/outputs |
| Reference potential for all outputs | Terminals $1.9 \ldots 4.9$ (negative pole of the <br> process supply voltage, signal name ZP) |
| Common power supply voltage | For all outputs: terminals $1.8 \ldots 4.8$ (positive pole <br> of the process supply voltage, signal name UP) |
| Output voltage for signal 1 | UP (-0.8 V) |
| Output delay (0->1 or 1->0) | Typ. $10 \mu \mathrm{~s}$ |
| Output data length | 32 bytes |
| Output current | 500 mA at UP = 24 V |
| Rated value, per channel | 8 A |
| Maximum value (all channels together, <br> PWM included) | $<0.5 \mathrm{~mA}$ |
| Leakage current with signal 0 | 10 A fast |
| Rated protection fuse on UP | With varistors integrated in the module (see <br> figure below) |
| Demagnetization when inductive loads are <br> switched off |  |
| Switching frequency |  |


| Parameter |  | Value |
| :--- | :--- | :--- |
| With resistive load | On request |  |
|  | With inductive loads | Max. 0.5 Hz |
|  | With lamp loads | Max. 11 Hz with max. 5 W |
| Short-circuit-proof / overload-proof | Yes |  |
| Overload message (I > 0.7 A) | Yes, after ca. 100 ms |  |
| Output current limitation | Yes, automatic reactivation after short cir- <br> cuit/overload |  |
| Resistance to feedback against 24 V signals | Yes |  |
| Max. cable length |  |  |
|  | Shielded | 1000 m |
|  | Unshielded | 600 m |



Fig. 167: Digital input/output (circuit diagram)
1 UPx (+ 24 V )
2 Digital input/output
3 ZPx (0 V)
4 For demagnization when inductive loads are switched off

Technical data of the highspeed inputs
(AD, BO, ZoO; A1, Bi, Z1)

| Parameter | Value |  |
| :--- | :--- | :--- |
| Number of channels per module | 6 |  |
| Reference potential for all inputs | Terminal 1.9, 2.9, 3.9 and 4.9 (negative <br> pole of the process voltage, signal name <br> ZR) |  |
| Input Type | 24 V DC | 5 V DC / Differential <br> Sinus 1 Wp |
| Input current per channel |  |  |
|  | Input voltage +24 V |  |
|  | Input voltage +5 V | $>4.8 \mathrm{~mA}$ |
|  | Input voltage +15 V | $>12 \mathrm{~mA}$ |
|  | Input voltage +30 V | $<15 \mathrm{~mA}$ |
| Input type acc. to EN 61131-2 | Type 1 |  |
| Input frequency max. (fast counter) | 300 kHz | 300 kHz |


| Parameter | Value |  |
| :--- | :--- | :--- |
| Input frequency max. (frequency measurement) | 5 kHz | 5 kHz |
| Input signal voltage | 24 V DC | 5 V DC |
| Signal 0 | $-3 \mathrm{~V} \ldots+5 \mathrm{~V}$ | $-3 \mathrm{~V} \ldots+0,5 \mathrm{~V}$ |
| Undefined signal | $>+5 \mathrm{~V} \ldots<+15 \mathrm{~V}$ | -- |
| Signal 1 | $+15 \mathrm{~V} \ldots+30 \mathrm{~V}$ | $+0,5 \mathrm{~V} \ldots+30 \mathrm{~V}$ |
| Ripple with signal 0 | Within $-3 \mathrm{~V} \ldots+5 \mathrm{~V}$ | Within <br> $-3 \mathrm{~V} \ldots .+0.5 \mathrm{~V}$ |
| Ripple with signal 1 | Within <br> $+15 \mathrm{~V} \ldots+30 \mathrm{~V}$ | Within <br> $+0,5 \mathrm{~V} \ldots+30 \mathrm{~V}$ |
| Max. cable length |  |  |
| Shielded | 1000 m |  |
| Unshielded | 600 m |  |


| Parameter | Value |
| :---: | :---: |
| Number of channels | 2 |
| Reference potential for all outputs | Terminals 1.9 ... 4.9 (negative pole of the process supply voltage, signal name ZP) |
| Common power supply voltage | For all outputs: terminals 1.8 ... 4.8 (positive pole of the process supply voltage, signal name UP) |
| Indication of the output signals | Brightness of the LED depends on the number of pulses emitted ( $0 \%$ to $100 \%$ ) (pulse output mode only) |
| Output voltage for signal 1 | UP (-0.1 V) |
| Output voltage for signal 0 | ZP (+0.3 V) |
| Output delay (0->1 or 1->0) | Typ. $1 \mu \mathrm{~s}$ |
| Output current |  |
| Rated value, per channel | 100 mA at $\mathrm{UP}=24 \mathrm{~V}$ |
| Maximum value (all channels together, configurable outputs included)) | 8 A |
| Leakage current with signal 0 | $<0.5 \mathrm{~mA}$ |
| Rated protection fuse on UP | 10 A fast |
| De-magnetization when inductive loads are switched off | With varistors integrated in the module (see figure above) |
| Switching frequency | PWM: up to 100 kHz (min. step for PWM value: $2 \mu \mathrm{~s}$ ) <br> Pulse: up to 15 kHz |
| Short-circuit-proof / overload-proof | Yes |
| Overload message ( l > $0.1 \times \mathrm{A}$ ) | Yes, after ca. 100 ms |
| Output current limitation | Yes, automatic reactivation after short-circuit/overload |
| Resistance to feedback against 24 V signals | Yes |
| Resistance to feedback against reverse polarity | No |
| Max. cable length |  |


| Parameter |  | Value |
| :--- | :--- | :--- |
|  | Shielded | 1000 m |
|  | Unshielded | 600 m |


| Technical data of the fast outputs (SSI CLK output B0, B1 for optical interface) | Parameter | Value |
| :---: | :---: | :---: |
|  | Number of channels | 2 |
|  | Reference potential for all outputs | Terminals 1.9...4.9 (negative pole of the process supply voltage, signal name ZP) |
|  | Common power supply voltage | For all outputs: terminals 1.8 ... 4.8 (positive pole of the process supply voltage, signal name UP) |
|  | Output voltage for signal 0 | $\leq 1.5 \mathrm{~V}$ at 10 mA |
|  | Output delay (0->1 or 1->0) | Typ. $0.3 \mu \mathrm{~s}$ |
|  | Output current | $\leq 10 \mathrm{~mA}$ |
|  | Switching frequency | $<1 \mathrm{MHz}$ (depending on firmware) |
|  | Short-circuit-proof / overload-proof | Yes |
|  | Output current limitation | Yes, automatic reactivation after short circuit/overload |
|  | Resistance to feedback against 24 V signals | Yes |
|  | Resistance to feedback against reverse polarity | No |
|  | Max. cable length (shielded) | Typ. 12.5 m at 500 kHz (depending on sensor) |

## Technical data of the fast outputs (SSI CLK Output Differential)

| Parameter | Value |
| :--- | :--- |
| Number of channels | 2 |
| Reference potential for all outputs | Terminals $1.9 \ldots 4.9$ (negative pole of the <br> process supply voltage, signal name ZP) |
| Common power supply voltage | For all outputs: terminals $1.8 \ldots .8$ (positive <br> pole of the process supply voltage, signal <br> name UP) |
| Output voltage for signal 1 | $\geq 2.9 \mathrm{~V}$ at 10 mA |
| Output voltage for signal 0 | $\leq 1.3 \mathrm{~V}$ at 10 mA |
| Output delay (0->1 or 1->0) | Typ. $0.3 \mu \mathrm{~s}$ |
| Output current | $\leq 10 \mathrm{~mA}$ |
| Switching frequency | $<1 \mathrm{MHz}$ (depending on firmware) |
| Short-circuit-proof / overload-proof | Yes |
| Overload message (I > 0.1x A) | Yes, after ca. 100 ms |
| Output current limitation | Yes, automatic reactivation after short-cir- <br> cuit/overload |
| Resistance to feedback against 24 V signals | Yes |
| Resistance to feedback against reverse <br> polarity | No |
| Max. cable length (shielded) | 100 m |

Technical data of the 5 V sensor supply

| Parameter | Value |
| :--- | :--- |
| Number of supplies | 2 , independently configuration |
| Voltage supply (outputs unloaded) | 5 V DC +/- $5 \%$ |
| Resistance to feedback against reverse <br> polarity | No |
| Output current | 100 mA max. (independently) <br> 200 mA max. (parallel use) |
| Output diagnosis | Yes, with diagnosis LED and error message |


| Technical data <br> of the 0 V refer- <br> ence input | Parameter | Value |
| :--- | :--- | :--- |
|  | Number of reference inputs (internally con- <br> nected to ZP through internal fuse) | 6 |
| Max. current per connection | 0.5 A |  |
| Internal fuse protection |  |  |
|  | Terminals 1.4 and 1.6 | 2 A |
|  | Terminals $3.4 \ldots 3.7$ | 2 A |

## Dimensions



[^16]The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 260 300 R0001 | CD522, encoder \& PWM module, <br> 2 encoder inputs, 2 PWM outputs, <br> 2 digital inputs 24 V DC, 8 digital <br> outputs 24 V DC | Active |
| 1SAP 460 300 R0001 | CD522-XC, encoder \& PWM module, <br> 2 encoder inputs, 2 PWM outputs, <br> 2 digital inputs 24 V DC, 8 digital <br> outputs 24 V DC, XC version | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.5 Terminal units for $\operatorname{S500}(-X C)$ I/O modules

### 5.2.5.1 Safety instructions

## Hot swap

System requirements for hot swapping of I/O modules:

- Types of terminal units that support hot swapping of I/O modules have the appendix TU5xx-H.
- I/O modules as of index FO.

The following I/O bus masters support hot swapping of attached I/O modules:

- Communication interface modules $\mathrm{Cl} 5 x x$ as of index FO.
- Processor modules PM56xx-2ETH with firmware version as of V3.2.0.


## NOTICE!

## Risk of damage to I/O modules!

Hot swapping is only allowed for I/O modules.
Processor modules and communication interface modules must not be removed or inserted during operation.

## Conditions for hot swapping

- Digital outputs are not under load.
- Input/output voltages above safety extra low voltage/ protective extra low voltages (SELV/PELV) are switched off.
- Modules are completely plugged on the terminal unit with both snap fit engaged before switching on loads or input/output voltage.
* Further information about hot swap.


### 5.2.5.2 TU515, TU516, TU541 and TU542 for I/O modules

### 5.2.5.2.1 Features

- TU515, I/O terminal unit, 24 V DC, screw terminals
- TU516, I/O terminal unit, 24 V DC, spring terminals
- TU516-XC, I/O terminal unit, 24 V DC, spring terminals, XC version
- TU516-H, I/O terminal unit, hot swap, 24 V DC, spring terminals
- TU516-H-XC, I/O terminal unit, hot swap, $24 \vee \mathrm{DC}$, spring terminals, XC version
- TU541, I/O terminal unit, 24 V DC, screw terminals
- TU542, I/O terminal unit, 24 V DC, spring terminals
- TU542-XC, I/O terminal unit, 24 V DC , spring terminals, XC version
- TU542-H, I/O terminal unit, hot swap, 24 V DC, spring terminals
- TU542-H-XC, I/O terminal unit, hot swap, 24 V DC, spring terminals, XC version

The input/output modules plug into the I/O terminal unit. When properly seated, they are secured with two mechanical locks. All the connections are established via the terminal unit, which allows removal and replacement of the I/O modules without disturbing the wiring at the terminal unit.


1 I/O bus (10 pins, male) to connect the previous terminal unit, the CPU terminal base or the communication interface module to the terminal unit
2 I/O bus ( 10 pins, female) to connect other terminal units
3a Plug ( $2 \times 25$ pins) to connect the inserted I/O modules
3b Plug ( $2 \times 19$ pins) to connect the inserted I/O modules
4 With a screwdriver inserted in this place, the terminal unit and the adjacent terminal unit can be shoved from each other
5 Holes for screw mounting
640 terminals for signals and process supply voltage
7 DIN rail
8 White border signifies hot swap capability of the terminal unit

XC version $\quad X C=e X t r e m e$ Conditions


## Extreme conditions

Terminal units for use in extreme ambient conditions have no $\underset{x_{+}^{*}+\underset{*}{*} \text { sign for } X C}{ }$ version.
The figure 4 in the Part no. 1SAP4... (lable) identifies the XC version.

## Terminals



The following terminals are used for connection of the process supply voltage.

|  | Terminals |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Type | 1.8 2.8 | 3.8 | 4.8 | 1.9 | 2.9 | 3.9 | 4.9 |
| TU515, TU516 and TU516-H | These terminals are internally connected with assignment: process supply voltage UP $=+24 \mathrm{~V}$ DC |  |  | These terminals are internally connected with assignment: process supply voltage $\mathrm{ZP}=0 \mathrm{~V}$ |  |  |  |
| $\begin{aligned} & \text { TU541, } \\ & \text { TU542 } \\ & \text { and } \\ & \text { TU542-H } \end{aligned}$ | These terminals are internally connected with assignment: process voltage UP $=+24 \mathrm{~V}$ DC | Separate process supply voltage UP3 = +24 V DC | Separate process supply voltage UP4 = +24 V DC |  | minals <br> ally con- <br> h assigncess <br> tage $\mathrm{ZP}=$ | Separate process supply voltage $\mathrm{ZP}=0 \mathrm{~V}$ | Separate process supply voltage $\mathrm{ZP}=0 \mathrm{~V}$ |

The assignment of the other terminals depends on the inserted communication interface module (see the description of the respective module used).

### 5.2.5.2.2 Technical data

The system data of AC500 and S500 are applicable to the standard version ${ }^{\Leftrightarrow}$ Chapter 5.1.2 "System data AC500" on page 166.
The system data of AC500-XC are applicable to the XC version ${ }^{\mu}$ Chapter 5.1.3 "System data AC500-XC" on page 169.

Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | Max. 32 |
| Distribution of the channels into groups | 4 groups of 8 channels each $(1.0 \ldots 1.7,2.0 \ldots$ <br> $2.7,3.0 \ldots 3.7,4.0 \ldots 4.7)$, the allocation of the <br> channels is given by the inserted I/O module |
| Rated voltage | 24 V DC |
| Max. permitted total current | 10 A, per separated process voltage terminal <br> or for internal connection of process voltages |


| Parameter | Value |
| :--- | :--- |
| Grounding | Direct connection to the grounded DIN rail or <br> via the screws with wall mounting |
| Screw terminals | Front terminal, conductor connection vertically <br> with respect to the printed circuit board |
| Spring terminals | Front terminal, conductor connection vertically <br> with respect to the printed circuit board |
| Weight | 200 g |
| Mounting position | Horizontal or vertical |

### 5.2.5.2.3 Hot swap

Hot swap

## WARNING!

Risk of explosion or fire in hazardous environments during hot swapping!
Hot swap must not be performed in flammable environments to avoid life-threatening injury and property damage resulting from fire or explosion.

## WARNING!

Electric shock due to negligent behavior during hot swapping!
To avoid electric shock

- make sure the following conditions apply:
- Digital outputs are not under load.
- Input/output voltages above safety extra low voltage/ protective extra low voltage (SELV/PELV) are switched off.
- Modules are fully interlocked with the terminal unit with both snap-fits engaged before switching on loads or input/output voltage.
- Never touch exposed contacts (dangerous voltages).
- Stay away from electrical contacts to avoid arc discharge.
- Do not operate a mechanical installation improperly.


## NOTICE! <br> Risk of damage to I/O modules!

Hot swapping is only allowed for I/O modules.
Processor modules and communication interface modules must not be removed or inserted during operation.

H = Hot swap

## Hot swap

System requirements for hot swapping of I/O modules:

- Types of terminal units that support hot swapping of I/O modules have the appendix TU5xx-H.
- I/O modules as of index F0.

The following I/O bus masters support hot swapping of attached I/O modules:

- Communication interface modules CI5xx as of index F0.
- Processor modules PM56xx-2ETH with firmware version as of V3.2.0.

Hot swap is not supported by AC500-eCo V3 CPU!


The index of the module is in the right corner of the label.

## NOTICE!

Risk of damage to I/O modules!
Modules with index below F0 can be damaged when inserted or removed from the terminal unit in a powered system.

## NOTICE!

## Risk of damage to I/O modules!

Do not perform hot swapping if any I/O module with firmware version lower than 3.0.14 is part of the I/O configuration.

For min. required device index see table below.

| Device | Min. required device index for I/O module as of FW Version 3.0.14 |
| :---: | :---: |
| AC522(-XC) | F0 |
| Al523 (-XC) | D2 |
| Al531 | D4 |
| Al531-XC | D2 |
| Al561 | B2 |
| Al562 | B2 |
| Al563 | B3 |
| AO523 (-XC) | D2 |
| AO561 | B2 |
| AX521 (-XC) | D2 |
| AX522 (-XC) | D2 |
| AX561 | B2 |
| CD522 (-XC) | D1 |
| DA501 (-XC) | D2 |
| DA502 (-XC) | F0 |
| DC522 (-XC) | D2 |
| DC523 (-XC) | D2 |
| DC532 (-XC) | D2 |
| DC562 | A2 |
| DI524 (-XC) | D2 |
| DI561 | B2 |
| DI562 | B2 |
| DI571 | B2 |
| DI572 | A1 |
| DO524 (-XC) | A3 |
| DO526 | A2 |
| DO526-XC | A0 |
| DO561 | B2 |
| DO562 | A2 |
| DO571 | B3 |
| DO572 | B2 |
| DO573 | A1 |
| DX522 (-XC) | D2 |
| DX531 | D2 |
| DX561 | B2 |
| DX571 | B3 |
| FM562 | A1 |

### 5.2.5.2.4 Dimensions



Din rail 15 mm
2 Din rail 7.5 mm

The dimensions are in mm and in brackets in inch.

### 5.2.5.2.5 Ordering data

| Part no. | Description | Product life cycle phase *) |
| :---: | :---: | :---: |
| 1SAP 212200 R0001 | TU515, I/O terminal unit, 24 V DC, screw terminals | Active |
| 1SAP 212000 R0001 | TU516, I/O terminal unit, 24 V DC, spring terminals | Active |
| 1SAP 412000 R0001 | TU516-XC, I/O terminal unit, 24 V DC, spring terminals, XC version | Active |
| 1SAP 215000 R0001 | TU516-H, I/O terminal unit, hot swap, 24 V DC, spring terminals, XC version | Active |
| 1SAP 415000 R0001 | TU516-H-XC, I/O terminal unit, hot swap, 24 V DC, spring terminals | Active |
| 1SAP 213000 R0001 | TU541, I/O terminal unit, 24 V DC, screw terminals | Active |
| 1SAP 213200 R0001 | TU542, I/O terminal unit, 24 V DC, spring terminals | Active |
| 1SAP 413200 R0001 | TU542-XC, I/O terminal unit, 24 V DC, spring terminals, XC version | Active |
| 1SAP 215200 R0001 | TU542-H, I/O terminal unit, hot swap, 24 V DC, spring terminals | Active |
| 1SAP 415200 R0001 | TU542-H-XC, I/O terminal unit, hot swap, 24 V DC, spring terminals, XC version | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.5.3 TU531 and TU532 for I/O modules

### 5.2.5.3.1 Features

- TU531, I/O terminal unit, 120/230 V AC, screw terminals
- TU532, I/O terminal unit, 120/230 V AC, spring terminals
- TU532-XC, I/O terminal unit, 120/230 V AC, spring terminals, XC version
- TU532-H, I/O terminal unit, hot swap, 120/230 V AC, spring terminals
- TU532-H-XC, I/O terminal unit, hot swap, 120/230 V AC, spring terminals, XC version


1 I/O bus (10 pins, male) to connect the previous terminal unit, the CPU terminal base or the communication interface module to the terminal unit
2 I/O bus (10 pins, female) to connect other terminal units
3a Plug ( $2 \times 25$ pins) to connect the inserted I/O modules
3b Plug ( $3 \times 19$ pins) to connect the inserted I/O modules
4 With a screwdriver inserted in this place, the terminal unit and the adjacent I/O terminal unit can be shoved from each other
5 Holes for screw mounting
640 terminals for signals and process supply voltage
7 DIN rail
8 White border signifies hot swap capability of the terminal unit

The input/output modules (I/O modules) plug into the I/O terminal unit. When properly pluggedin, they are secured with two mechanical locks. All the connections are established via the terminal unit, which allows removal and replacement of the I/O modules without disturbing the wiring at the terminal unit.

The terminal units TU531 and TU532 are specifically designed for use with AC500/S500 I/O modules that incorporate 115 V AC ... 230 V AC inputs and/or 120/230 V AC relay outputs.

XC version XC = eXtreme Conditions


## Extreme conditions

 XC version.
The figure 4 in the Part no. 1SAP4... (label) identifies the XC version.

## Terminals

| Screw terminals |  |  | Spring terminals |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Conductor |  | Screwdriver | Conductor | $\left\|\left\|\left\|\begin{array}{l} \bigcirc_{1.5}^{1.5} \\ \bigcirc_{1.6} \\ O_{1.7} \\ O_{1.8} \\ O_{1.9} \end{array}\right\|\right.\right.$ | Screwdriver (opens terminal) |

The terminals $1.8 \ldots 4.8$ and $1.9 \ldots 4.9$ are electrically interconnected within the terminal unit and always have the same assignment, independent of the inserted module:

- Terminals $1.8 \ldots 4.8$ : process supply voltage UP $=+24 \mathrm{~V}$ DC
- Terminals $1.9 \ldots 4.9$ : process supply voltage $\mathrm{ZP}=0 \mathrm{~V}$

The assignment of the other terminals depends on the inserted communication interface module (see the description of the respective module used).
The supply voltage of 24 V DC for the module's circuitry comes from the I/O expansion bus (I/O bus).

### 5.2.5.3.2 Technical data

The system data of AC500 and S500 are applicable to the standard version ${ }^{*} \Rightarrow$ Chapter 5.1 .2 "System data AC500" on page 166.

The system data of AC500-XC are applicable to the XC version ${ }^{\circledR}$ Chapter 5.1.3 "System data AC500-XC" on page 169.

Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 32 |
| Distribution of the channels into groups | 4 groups of 8 channels each $(1.0 \ldots 1.7,2.0 \ldots$ <br> $2.7,3.0 \ldots 3.7,4.0 \ldots 4.7)$, the allocation of the <br> channels is given by the inserted I/O module |
|  |  |
| Terminals $1.8 \ldots 4.8$ and $1.9 \ldots 4.9$ | 30 V DC |
|  | Max. voltage |
| Max. permitted total current | 10 A |
| Terminals $1.0 \ldots 1.7,2.0 \ldots 2.7,3.0 \ldots 3.7,4.0 \ldots 4.7$ |  |
|  | Max. voltage |
| Max. permitted current | 300 V AC ${ }^{1}$ ) |
| Grounding | $\left.3 \mathrm{~A}^{2}\right)$ |
| Screw terminals | Direct connection to the grounded DIN rail or <br> via the screws with wall mounting |
| Spring terminals | Front terminal, conductor connection vertically <br> with respect to the printed circuit board |
| Weight | Front terminal, conductor connection vertically <br> with respect to the printed circuit board |
| Mounting position | 200 g |

${ }^{1}$ ) Only when the voltage is not limited by the specification of the I/O channel or the supply input which is internally connected to the terminal.
${ }^{2}$ ) The terminals are connected to the electronic module via internal connectors (X22 (or 3b), X23 (or 3b), X32, X33 and X34). The current per terminal is limited by the permitted current of these connectors.

### 5.2.5.3.3 Hot swap

## Hot swap

## WARNING!

Risk of explosion or fire in hazardous environments during hot swapping!
Hot swap must not be performed in flammable environments to avoid life-threatening injury and property damage resulting from fire or explosion.

## WARNING!

## Electric shock due to negligent behavior during hot swapping!

To avoid electric shock

- make sure the following conditions apply:
- Digital outputs are not under load.
- Input/output voltages above safety extra low voltage/ protective extra low voltage (SELV/PELV) are switched off.
- Modules are fully interlocked with the terminal unit with both snap-fits engaged before switching on loads or input/output voltage.
- Never touch exposed contacts (dangerous voltages).
- Stay away from electrical contacts to avoid arc discharge.
- Do not operate a mechanical installation improperly.


## NOTICE!

Risk of damage to I/O modules!
Hot swapping is only allowed for I/O modules.
Processor modules and communication interface modules must not be removed or inserted during operation.

H = Hot swap

## Hot swap

System requirements for hot swapping of I/O modules:

- Types of terminal units that support hot swapping of I/O modules have the appendix TU5xx-H.
- I/O modules as of index FO.

The following I/O bus masters support hot swapping of attached I/O modules:

- Communication interface modules CI5xx as of index F0.
- Processor modules PM56xx-2ETH with firmware version as of V3.2.0.

Hot swap is not supported by AC500-eCo V3 CPU!


The index of the module is in the right corner of the label.

## NOTICE!

## Risk of damage to I/O modules!

Modules with index below F0 can be damaged when inserted or removed from the terminal unit in a powered system.

## NOTICE!

## Risk of damage to I/O modules!

Do not perform hot swapping if any I/O module with firmware version lower than 3.0.14 is part of the I/O configuration.

For min. required device index see table below.

| Device | Min. required device index for I/O module as of <br> FW Version 3.0.14 |
| :--- | :--- |
| AC522(-XC) | F0 |
| Al523 (-XC) | D2 |
| Al531 | D4 |
| Al531-XC | B2 |
| Al561 | B2 |
| AI562 | B3 |
| AI563 | D2 |
| AO523 (-XC) | B2 |
| AO561 | D2 |
| AX521 (-XC) | D2 |
| AX522 (-XC) | B2 |
| AX561 | D1 |
| CD522 (-XC) | D2 |
| DA501 (-XC) | F0 |
| DA502 (-XC) | D2 |
| DC522 (-XC) | D2 |
| DC523 (-XC) | D2 |
| DC532 (-XC) | A2 |
| DC562 | D2 |
| DI524 (-XC) | B2 |
| DI561 | B2 |
| DI562 | A1 |
| DI571 | DI572 |
|  |  |


| Device | Min. required device index for I/O module as of <br> FW Version 3.0.14 |
| :--- | :--- |
| DO524 (-XC) | A3 |
| DO526 | A2 |
| DO526-XC | A0 |
| DO561 | B2 |
| DO562 | A2 |
| DO571 | B3 |
| DO572 | B2 |
| DO573 | A1 |
| DX522 (-XC) | D2 |
| DX531 | D2 |
| DX561 | B2 |
| DX571 | B3 |
| FM562 | A1 |

### 5.2.5.3.4 Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm

The dimensions are in mm and in brackets in inch.

### 5.2.5.3.5 Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 217 200 <br> R0001 | TU531, terminal unit, 120/230 V AC, <br> relays, screw terminals | Active |
| 1SAP 217 000 <br> R0001 | TU532, terminal unit, 120/230 V AC, <br> relays, spring terminals | Active |
| 1SAP 417 000 <br> R0001 | TU532-XC, terminal unit, <br> 120/230 V AC, relays, spring <br> terminals, XC version | Active |
| 1SAP 215 100 <br> R0001 | TU532-H, terminal unit, hot swap, <br> 120/230 V AC, relays, spring terminals | Active |
| 1SAP 415 100 <br> R0001 | TU532-H-XC, terminal unit, hot <br> swap, 120/230 V AC, relays, spring <br> terminals, XC version | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.6 Communication interface modules

### 5.2.6.1 Safety instructions

## Hot swap

System requirements for hot swapping of I/O modules:

- Types of terminal units that support hot swapping of I/O modules have the appendix TU5xx-H.
- I/O modules as of index FO.

The following I/O bus masters support hot swapping of attached I/O modules:

- Communication interface modules CI5xx as of index FO.
- Processor modules PM56xx-2ETH with firmware version as of V3.2.0.


## NOTICE!

## Risk of damage to I/O modules!

Hot swapping is only allowed for I/O modules.
Processor modules and communication interface modules must not be removed or inserted during operation.

## Conditions for hot swapping

- Digital outputs are not under load.
- Input/output voltages above safety extra low voltage/ protective extra low voltages (SELV/PELV) are switched off.
- Modules are completely plugged on the terminal unit with both snap fit engaged before switching on loads or input/output voltage.
$\rightleftarrows$ Further information about hot swap.


### 5.2.6.2 Compatibility of communication modules and communication interface modules

Table 212: Modbus TCP

| Communication <br> module | Communication <br> interface <br> module | l/O expansion <br> module <br> S500 | l/O expansion <br> module <br> s500-eCo | I/O expansion <br> module <br> S500-s | Applications |
| :--- | :--- | :--- | :--- | :--- | :--- |
| Onboard <br> Ethernet inter- <br> face | Cl521-MODTCP <br> CI522-MODTCP | x | x | -- | high availability, <br> remote I/O |
| Onboard <br> Ethernet inter- <br> face | Cl521-MODTCP <br> CI522-MODTCP | x | -- | hot-swap I/O |  |
| CM5640-2ETH | CI521-MODTCP <br> CI522-MODTCP | x | x | -- | high availability, <br> remote I/O |
| CM5640-2ETH | CI521-MODTCP <br> CI522-MODTCP | x | -- | hot-swap I/O |  |

Table 213: PROFIBUS DP

| Communication <br> module | Communication <br> interface <br> module | I/O expansion <br> module <br> S500 | I/O expansion <br> module <br> S500-eCo | I/O expansion <br> module <br> S500-S | Applications |
| :--- | :--- | :--- | :--- | :--- | :--- |
| CM592-DP <br> master | Cl541-DP <br> CI542-DP | x | x | -- | remote I/O |
| CM592-DP <br> master | Cl541-DP <br> CI542-DP | x | -- | -- | hot-swap I/O |

Table 214: PROFINET IO RT

| Communication <br> module | Communication <br> interface <br> module | I/O expansion <br> module <br> S500 | I/O expansion <br> module <br> S500-eCo | I/O expansion <br> module <br> S500-S | Applications |
| :--- | :--- | :--- | :--- | :--- | :--- |
| CM579-PNIO <br> controller | CI501-PNIO <br> CI502-PNIO | x | x | x | remote I/O, <br> safety I/O |
| CM579-PNIO <br> controller | CI501-PNIO <br> CI502-PNIO | x | -- | hot-swap I/O |  |

Table 215: CANopen

| Communication <br> module | Communication <br> interface <br> module | l/O expansion <br> module <br> S500 | l/O expansion <br> module <br> S500-eCo | I/O expansion <br> module <br> S500-S | Applications |
| :--- | :--- | :--- | :--- | :--- | :--- |
| Onboard CAN <br> interface | CI581-CN <br> CI582-CN | -- | -- | remote I/O |  |

Table 216: EtherCAT

| Communication <br> module | Communication <br> interface <br> module | l/O expansion <br> module <br> S500 | I/O expansion <br> module <br> S500-eCo | I/O expansion <br> module <br> S500-S | Applications |
| :--- | :--- | :--- | :--- | :--- | :--- |
| CM579-ETHCAT <br> master | CI511-ETHCAT <br> CI512-ETHCAT | x | x | -- | remote I/O |

### 5.2.6.3 CANopen

### 5.2.6.3.1 Comparison CI581 and CI582

The devices differ in their input and output characteristics.

| CI581-CN: Input/ Output characteristics | Parameter <br> Inputs and outputs | Value <br> 8 digital inputs ( 24 V DC; delay time configurable via software) <br> 8 digital transistor outputs ( 24 V DC, 0.5 A max.) <br> 4 analog inputs, configurable as: <br> - $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ <br> - $0 \mathrm{~V} . .+10 \mathrm{~V}$ <br> - -10 V ... +10 V (differential voltage) <br> - $0 \mathrm{~mA} . . .20 \mathrm{~mA}$ <br> - $4 \mathrm{~mA} . . .20 \mathrm{~mA}$ <br> - Pt100, Pt1000, Ni1000 (for each 2-wire and 3-wire) <br> - 24 V digital input function <br> 2 analog outputs, configurable as: <br> - $-10 \mathrm{~V} . . .+10 \mathrm{~V}$ <br> - $0 \mathrm{~mA} . . .20 \mathrm{~mA}$ <br> - $4 \mathrm{~mA} . .20 \mathrm{~mA}$ |
| :---: | :---: | :---: |
|  | Resolution of the analog channels | 12 bits |
|  | Fast counter | Integrated, configurable operating modes |


| CI582-CN: Input/ Output characteristics | Parameter | Value |
| :---: | :---: | :---: |
|  | Inputs and outputs | 8 digital inputs ( 24 V DC) |
|  |  | 8 digital transistor outputs ( 24 V DC, 0.5 A max.) |
|  |  | 8 configurable digital inputs/outputs ( 24 V DC, 0.5 A max.) |

### 5.2.6.3.2 CI581-CN

## Features

- 4 analog inputs (resolution 12 bits including sign)
- 2 analog outputs (resolution 12 bits including sign)
- 8 digital inputs 24 V DC
- 8 digital outputs 24 V DC, 0.5 A max
- Module-wise galvanically isolated
- Fast counter
- XC version for use in extreme ambient conditions available


I/O bus
2 Allocation between terminal No. and signal name
36 yellow LEDs to display the signal states of the analog inputs/outputs (AIO ... AI3, AO0 ... AO1)
48 yellow LEDs to display the signal states of the digital inputs (DI0 ... DI7)
58 yellow LEDs to display the signal states of the digital outputs (DO0 ... DO7)
2 green LEDs to display the supply voltage UP and UP3
3 red LEDs to display errors (CH-ERR1, CH-ERR2, CH-ERR3)
5 System LEDs: PWR/RUN, CN-RUN, CN-ERR, S-ERR, I/O-Bus
Label
102 rotary switches for setting the CANopen Node ID
10 terminals to connect the CANopen bus signals
Terminal unit
DIN rail
Sign for XC version

## Intended purpose

The CANopen communication interface module $\mathrm{CI} 581-\mathrm{CN}$ is used as decentralized I/O module in CANopen networks. Depending on the used terminal unit the network connection is performed either via 9-pin female D-sub or via 10 terminals (screw or spring terminals) which are integrated in the terminal unit. The communication interface module contains 22 I/O channels with the following properties:

- 4 analog inputs (2.0 ... 2.3)
- 2 analog outputs (2.5 ... 2.6)
- 8 digital inputs 24 V DC in 1 group (3.0 ... 3.7)
- 8 digital outputs 24 V DC in 1 group (4.0 ... 4.7)

The inputs/outputs are galvanically isolated from the CANopen network. There is no potential separation between the channels. The configuration of the analog inputs/outputs is performed by software.
For use in extreme ambient conditions (e.g. wider temperature and humidity range), a special XC version of the device is available.

## Functionality

| Parameter | Value |
| :---: | :---: |
| Interface | CAN |
| Protocol | CANopen |
| Power supply | From the process supply voltage UP |
| Supply of the electronic circuitry of the I/O modules attached | Through the I/O bus interface (I/O bus) |
| Rotary switches | For setting the CANopen Node ID for configuration purposes (00h to FFh) |
| LED displays | For system displays, signal states, errors and power supply |
| External supply voltage | Via terminals ZP, UP and UP3 (process supply voltage 24 V DC) |
| Transmission rates | 10 / 20 / 50 / 125 / $250 / 500 / 800 \mathrm{kbit} / \mathrm{s} 1$ Mbit/s Auto transmission rate detection is supported |
| Bus connection | Depending on used terminal unit TU510: 9-pin D-sub connector TU518: 10-pin terminal block |
| Processor | Hilscher NETX 100 |
| Expandability | CI58x can only be used on onboard CAN interface and without any I/O expansion module « Table 215 "CANopen" on page 952. |
| State display | Module state: PWR/RUN, CN-RUN, CN-ERR, E-ERR, I/O bus |
| Adjusting elements | 2 rotary switches for generation of the node address |
| Ambient temperature | System data AC500 ${ }^{\star}$ Chapter 5.1.2 "System data AC500" on page 166 <br> System data AC500 XC $\Rightarrow$ Chapter 5.1.3 <br> "System data AC500-XC" on page 169 |
| Current consumption | UP: 0.2 A UP3: 0.06 A + 0.5 A max. per output |


| Parameter | Value |
| :---: | :---: |
| Weight (without terminal unit) | Ca. 125 g |
| Process supply voltages UP/UP3 |  |
| Rated value | 24 V DC (for inputs and outputs) |
| Max. load for the terminals | 10 A |
| Protection against reversed voltage | Yes |
| Rated protection fuse on UP/UP3 | 10 A fast |
| Galvanic isolation | CANopen interface against the rest of the module |
| Inrush current from UP (at power up) | On request |
| Current consumption via UP (normal operation) | 0.2 A |
| Current consumption via UP3 | 0.06 A + 0.5 A max. per output |
| Connections | Terminals 2.8 and 3.8 for +24 V (UP) <br> Terminal 4.8 for +24 V (UP3) <br> Terminals 2.9, 3.9 and 4.9 for 0 V (ZP) |
| Max. power dissipation within the module | 6 W |
| Reference potential for all digital inputs and outputs | Negative pole of the supply voltage, signal name ZP |
| Setting of the CANopen Node ID identifier | With 2 rotary switches at the front side of the module |
| Mounting position | Horizontal <br> Or vertical with derating (output load reduced to $50 \%$ at $+40^{\circ} \mathrm{C}$ per group) |
| Cooling | The natural convection cooling must not be hindered by cable ducts or other parts in the control cabinet. |
| Effect of incorrect input terminal connection | Wrong or no signal detected, no damage up to 35 V |
| Required terminal unit | TU509, TU510, TU517 or TU518 <br> * Chapter 5.2.7.4 "TU517 and TU518 for communication interface modules" on page 1278 |

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and continuous overvoltage up to 30 V DC.

CI581-CN: Input/ Output characteristics

| Parameter | Value |
| :---: | :---: |
| Inputs and outputs | 8 digital inputs (24 V DC; delay time configurable via software) <br> 8 digital transistor outputs (24 V DC, 0.5 A max.) <br> 4 analog inputs, configurable as: <br> - $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ <br> - $0 \mathrm{~V} \ldots+10 \mathrm{~V}$ <br> - $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ (differential voltage) <br> - $0 \mathrm{~mA} . .20 \mathrm{~mA}$ <br> - $4 \mathrm{~mA} . .20 \mathrm{~mA}$ <br> - Pt100, Pt1000, Ni1000 (for each 2-wire and 3-wire) <br> - 24 V digital input function <br> 2 analog outputs, configurable as: <br> - $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ <br> - $0 \mathrm{~mA} . .20 \mathrm{~mA}$ <br> - $4 \mathrm{~mA} . .20 \mathrm{~mA}$ |
| Resolution of the analog channels | 12 bits |
| Fast counter | Integrated, configurable operating modes |

## Connections

## General

The CANopen communication interface module is plugged on the I/O terminal units TU517 ² Chapter 5.2.7.4 "TU517 and TU518 for communication interface modules" on page 1278 or TU518 $\stackrel{\wedge}{ }{ }^{\circ}$ Chapter 5.2.7.4 "TU517 and TU518 for communication interface modules" on page 1278 and accordingly TU509 or TU510. Properly position the module and press until it locks in place.

The connection of the I/O channels is established using the 30 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

The terminals $2.8,3.8,2.9,3.9$ and 4.9 are electrically interconnected within the terminal unit and always have the same assignment, irrespective of the inserted module:

Terminals 2.8 and 3.8: process supply voltage UP $=+24 \mathrm{~V}$ DC
Terminal 4.8: process supply voltage UP3 $=+24 \mathrm{~V}$ DC
Terminals 2.9, 3.9 and 4.9: process supply voltage $\mathrm{ZP}=0 \mathrm{~V}$

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.


With a separate UP3 power supply, the digital outputs can be switched off externally. This way, an emergency-off functionality can be realized.

Do not connect any voltages externally to the digital outputs!
Reason: External voltages at an output or several outputs may cause other outputs to be supplied via that voltage instead of voltage UP3 (reverse voltage). This ist not the intended use.

## CAUTION!

## Risk of malfunctions by unintended use!

If the function cut-off of the digital outputs is to be used by deactivation of the supply voltage UP3, be sure that no external voltage is connected at the outputs DO0 ... DO7 and DC0 ... DC7.

## Possibilities of connection

Mounting on ter- The assignment of the 9-pin female D-sub for the CANopen signals minal units TU509 or TU510

|  | 1 | --- | Reserved |
| :---: | :---: | :---: | :---: |
|  | 2 | CAN- | Inverted signal of the CAN bus |
|  | 3 | CAN_GND | Ground potential of the CAN bus |
|  | 4 | --- | Reserved |
|  | 5 | --- | Reserved |
|  | 6 | --- | Reserved |
|  | 7 | CAN+ | Non-inverted signal of the CAN bus |
|  | 8 | --- | Reserved |
|  | 9 | --- | Reserved |
|  | Shield | Cable shield | Functional earth |

Bus terminating The ends of the data lines have to be terminated with a $120 \Omega$ bus terminating resistor. The bus resistors terminating resistor is usually installed directly at the bus connector.


Fig. 168: CANopen interface, bus terminating resistors connected to the line ends

| 1 | CAN_GND |
| :--- | :--- |
| 2 | CAN_L |
| 3 | Shield |
| 4 | CAN_H |
| 5 | Data line, shielded twisted pair |
| 6 | COMBICON connection, CANopen interface |



Fig. 169: DeviceNet interface, bus terminating resistors connected to the line ends

| 6 | DeviceNet power supply |
| :--- | :--- |
| 7 | COMBICON connection, DeviceNet interface |
| 8 | Data lines, twisted pair cables |
| 9 | red |
| 10 | black |
| 11 | white |
| 12 | blue |
| 13 | bare |

The grounding of the shield should take place at the switchgear $\stackrel{\Perp}{ }$ Chapter 5.1.2 "System data AC500" on page 166.

Mounting on ter- Table 217: Assignment of the terminals minal units TU517 or TU518

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 1.0 | CAN+ | Non-inverted signal of the CAN bus |
| 1.1 | CAN+ | Non-inverted signal of the CAN bus |
| 1.2 | CAN- | Inverted signal of the CAN bus |
| 1.3 | Term + | Inverted signal of the CAN bus |
| 1.4 | Term- | CAN bus termination for CAN+ (for bus termination, <br> Term+ must be connected with CAN+) |
| 1.5 | CAN bus termination for CAN+ (connecting alterna- <br> tive for terminal 1.4) |  |
| 1.6 | CAN bus termination for CAN- (for bus termination, <br> Term- must be connected with CAN-) |  |
| 1.7 | CAN-GND | CAN bus termination for CAN- (connecting alterna- <br> tive for terminal 1.6) |
| 1.8 | CAN-GND | Ground potential of the CAN bus |
| 1.9 | Ground potential of the CAN bus |  |

At the line ends of a bus segment, terminating resistors must be connected. If TU517 or TU518 is used, the bus terminating resistors can be enabled by connecting the terminals Term+ and Term- to the data lines CAN+ and CAN- (no external terminating resistors are required, see figure below).
The following figures show the different connection options for the CANopen communication interface module:



In the case of TU517/TU518, the terminating resistors are not located inside the TU but inside the communication interface module CI581-CN. Hence, when removing the device from the TU, the bus terminating resistors are no longer connected to the bus. The bus itself will not be disconnected if a device is removed.

The grounding of the shield should take place at the control cabinet. Please refer to the AC500 System-Data "y Chapter 5.1.2 "System data AC500" on page 166.

Table 218: Assignment of the other terminals

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 2.0 | Al0+ | Positive pole of analog input signal 0 |
| 2.1 | Al1+ | Positive pole of analog input signal 1 |
| 2.2 | Al2+ | Positive pole of analog input signal 2 |
| 2.3 | Al3+ | Positive pole of analog input signal 3 |
| 2.4 | AI- | Negative pole of analog input signals 0 to 3 |
| 2.5 | AO0+ | Positive pole of analog output signal 0 |
| 2.6 | AO1+ | Positive pole of analog output signal 1 |
| 2.7 | AI- | Negative pole of analog output signals 0 and 1 |
| 2.8 | UP | Process voltage UP (24 V DC) |
| 2.9 | ZP | Process voltage ZP (0 V DC) |
| 3.0 | DI0 | Signal of the digital input DI0 |
| 3.1 | DI1 | Signal of the digital input DI1 |
| 3.2 | DI2 | Signal of the digital input DI2 |
| 3.3 | DI3 | Signal of the digital input DI3 |


| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 3.4 | DI4 | Signal of the digital input DI4 |
| 3.5 | DI5 | Signal of the digital input DI5 |
| 3.6 | DI6 | Signal of the digital input DI6 |
| 3.7 | DI7 | Signal of the digital input DI7 |
| 3.8 | UP | Process voltage UP (24 V DC) |
| 3.9 | ZP | Process voltage ZP (0 V DC) |
| 4.0 | DO0 | Signal of the digital output DO0 |
| 4.1 | DO1 | Signal of the digital output DO1 |
| 4.2 | DO2 | Signal of the digital output DO2 |
| 4.3 | DO3 | Signal of the digital output DO3 |
| 4.4 | DO4 | Signal of the digital output DO4 |
| 4.5 | DO5 | Signal of the digital output DO5 |
| 4.6 | DO6 | Signal of the digital output DO6 |
| 4.7 | DO7 | Signal of the digital output DO7 |
| 4.8 | UP3 | Process voltage UP3 (24 V DC) |
| 4.9 | ZP | Process voltage ZP (0 V DC) |

## WARNING!

## Removal/Insertion under power

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

## Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

For the open-circuit detection (wire break), each analog input channel is pulled up to "plus" by a high-resistance resistor. If nothing is connected, the maximum voltage will be read in then.

Generally, analog signals must be laid in shielded cables. The cable shields must be grounded at both sides of the cables. In order to avoid unacceptable potential differences between different parts of the installation, low resistance equipotential bonding conductors must be laid.


Fig. 170: Connection of the communication interface module Cl581-CN
The module provides several diagnosis functions $\stackrel{y}{ } \stackrel{y}{c}$ Chapter 5.2.6.3.2.9 "Diagnosis" on page 979.
For the measuring ranges that can be configured, please refer to the sections Measuring Ranges $\Leftrightarrow$ Chapter 5.2.6.3.2.11 "Measuring ranges" on page 984 and Parameterization Chapter 5.2.6.3.2.8 "Parameterization" on page 974.
The meaning of the LEDs is described in the section for the state LEDs ${ }_{y}{ }^{\mu}$ Chapter 5.2.6.3.2.10 "State $L E D$ " on page 982.

Bus length The maximum possible bus length of a CAN network depends on bit rate (transmission rate) and cable type. The sum of all bus segments must not exceed the maximum bus length

| Bit Rate (speed) | Bus Length |
| :--- | :--- |
| $1 \mathrm{Mbit} / \mathrm{s}$ | 40 m |
| $800 \mathrm{kbit} / \mathrm{s}$ | 50 m |
| $500 \mathrm{kbit} / \mathrm{s}$ | 100 m |
| $250 \mathrm{kbit} / \mathrm{s}$ | 250 m |


| Bit Rate (speed) | Bus Length |
| :--- | :--- |
| $125 \mathrm{kbit} / \mathrm{s}$ | 500 m |
| $50 \mathrm{kbit} / \mathrm{s}$ | 1000 m |

## Connection of the digital inputs



Fig. 171: Connection of the digital inputs to the module CI581-CN

## Connection of the digital outputs



Fig. 172: Connection of configurable digital outputs to the module CI581-CN

## Connection of resistance thermometers in 2-wire configuration to the analog inputs

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow to build the necessary voltage drop for the evaluation. For this, the module CI581-CN provides a constant current source which is multiplexed over the max. 4 analog input channels.


Fig. 173: Connection of resistance thermometers in 2-wire configuration to the analog inputs

| Pt100 | 2-wire configuration, 1 channel used |
| :--- | :--- |
| Pt1000 | 2-wire configuration, 1 channel used |
| Ni1000 | 2-wire configuration, 1 channel used |

For the measuring ranges that can be configured, please refer to sections Measuring Ranges » Chapter 5.2.6.3.2.11 "Measuring ranges" on page 984 and Parameterization ${ }^{\star} »$ Chapter 5.2.6.3.2.8 "Parameterization" on page 974.

The module CI581-CN performs a linearization of the resistance characteristic.
To avoid error messages, configure unused analog input channels as "unused".

## Connection of resistance thermometers in 3-wire configuration to the analog inputs

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the module CI581-CN provides a constant current source which is multiplexed over the max. 4 analog input channels.


Fig. 174: Connection of resistance thermometers in 3-wire configuration to the analog inputs
With 3 -wire configuration, 2 adjacent analog channels belong together (e. g. the channels 0 and 1). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0 ), the next higher address must be the odd address (channel 1 ).
The constant current of one channel flows through the resistance thermometer. The constant current of the other channel flows through one of the cores. The module calculates the measured value from the two voltage drops and stores it under the input with the higher channel number (e. g. I1).
In order to keep measuring errors as small as possible, it is necessary to have all the involved conductors in the same cable. All the conductors must have the same cross section.

| Pt 100 | 3-wire configuration, 2 channels used |
| :--- | :--- |
| Pt 1000 | 3-wire configuration, 2 channels used |
| Ni1000 | 3-wire configuration, 2 channels used |

For the measuring ranges that can be configured, please refer to the sections Measuring Ranges * Chapter 5.2.6.3.2.11 "Measuring ranges" on page 984 and Parameterization * Chapter 5.2.6.3.2.8 "Parameterization" on page 974.

The module $\mathrm{Cl} 581-\mathrm{CN}$ performs a linearization of the resistance characteristic.
To avoid error messages, configure unused analog input channels as "unused".

Connection of active-type analog sensors (Voltage) with galvanically isolated power supply to the analog inputs


Fig. 175: Connection of active-type analog sensors (voltage) with galvanically isolated power supply to the analog inputs

| Voltage | $0 \ldots 10 \mathrm{~V}$ | 1 channel used |
| :--- | :--- | :--- |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | 1 channel used |

For the measuring ranges that can be configured, please refer to the sections Measuring Ranges $\stackrel{y}{c}$ Chapter 5.2.6.3.2.11 "Measuring ranges" on page 984 and Parameterization $\Leftrightarrow$ Chapter 5.2.6.3.2.8 "Parameterization" on page 974.
To avoid error messages, configure unused analog input channels as "unused".

## Connection of active-type analog sensors (Current) with galvanically isolated power supply to the analog inputs



Fig. 176: Connection of active-type analog sensors (current) with galvanically isolated power supply to the analog input AIO (Proceed with the analog inputs Al1 ... AI3 in the same way)

| Current | $0 \ldots 20 \mathrm{~mA}$ | 1 channel used |
| :--- | :--- | :--- |
| Current | $4 \ldots 20 \mathrm{~mA}$ | 1 channel used |

For the measuring ranges that can be configured, please refer to the sections Measuring Ranges ${ }^{\wedge} \Rightarrow$ Chapter 5.2.6.3.2.11 "Measuring ranges" on page 984 and Parameterization $\Leftrightarrow$ Chapter 5.2.6.3.2.8 "Parameterization" on page 974.
Unused input channels can be left open-circuited, because they are of low resistance.

Connection of active-type analog sensors (Voltage) with no galvanically isolated power supply to the analog inputs


Fig. 177: Connection of active-type sensors (voltage) with no galvanically isolated power supply to the analog inputs (AO ... AI3)

## NOTICE!

## Risk of faulty measurements!

The negative pole/ground potential at the sensors must not have too large a potential difference with respect to ZP (max. $\pm 1 \mathrm{~V}$ within the full signal range).

Make sure that the potential difference never exceeds $\pm 1 \mathrm{~V}$.

| Voltage | $0 \ldots 10 \mathrm{~V}$ | 1 channel used |
| :--- | :--- | :--- |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | 1 channel used |

For the measuring ranges that can be configured, plese refer to the sections Measuring Ranges ${ }^{\#}$ Chapter 5.2.6.3.2.11 "Measuring ranges" on page 984 and Parameterization ${ }^{\#}$ Chapter 5.2.6.3.2.8 "Parameterization" on page 974.

To avoid error messages, configure unused analog input channels as "unused".

## Connection of passive-type analog sensors (Current) to the analog inputs



Fig. 178: Connection of passive-type analog sensors (current) to the analog inputs (AO ... A3)

| Current | $4 \ldots 20 \mathrm{~mA}$ | 1 channel used |
| :--- | :--- | :--- |

## CAUTION!

## Risk of overloading the analog input!

If an analog current sensor supplies more than 25 mA for more than 1 second during initialization, this input is switched off by the module (input protection).

Only use sensors with fast initialization or without current peaks higher than 25 mA . If not possible, connect a 10-volt Zener diode in parallel to I+ and I-.

Unused input channels can be left open-circuited, because they are of low resistance.

## Connection of active-type analog sensors (Voltage) to differential analog inputs

Differential inputs are very useful if analog sensors which are remotely non-isolated (e.g. the negative terminal is remotely grounded) are used.

Using differential inputs helps to considerably increase the measuring accuracy and to avoid ground loops.
With differential input configurations, two adjacent analog channels belong together (e.g. the channels 0 and 1 ). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0 ), the next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1).
The analog value is calculated by subtraction of the input value with the higher address from the input value of the lower address.

The converted analog value is available at the odd channel (higher address).

## NOTICE!

## Risk of faulty measurements!

The negative pole/ground potential at the sensors must not have too large a potential difference with respect to ZP (max. $\pm 1 \mathrm{~V}$ within the full signal range).

Make sure that the potential difference never exceeds $\pm 1 \mathrm{~V}$.


Fig. 179: Connection of active-type analog sensors (voltage) to differential analog inputs (AIO ... Al3)

| Voltage | $0 \ldots 10 \mathrm{~V}$ | with differential inputs, 2 chan- <br> nels used |
| :--- | :--- | :--- |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | with differential inputs, 2 chan- <br> nels used |

For the measuring ranges that can be configured, please refer to the sections Measuring Ranges ${ }^{\wedge}>$ Chapter 5.2.6.3.2.11 "Measuring ranges" on page 984 and Parameterization Chapter 5.2.6.3.2.8 "Parameterization" on page 974.

To avoid error messages, configure unused analog input channels as "unused".

## Use of analog inputs as digital inputs

Several (or all) analog inputs can be configured as digital inputs. The inputs are not galvanically isolated against the other analog channels.


Fig. 180: Connection of digital sensors to the analog input (AIO ... AI3)

| Digital input | 24 V | 1 channel used |
| :--- | :--- | :--- |

For the measuring ranges that can be configured, please refer to the sections Measuring Ranges $\stackrel{y}{ }{ }^{*}$ Chapter 5.2.6.3.2.11 "Measuring ranges" on page 984 and Parameterization $\left.{ }^{*}\right\rangle$ Chapter 5.2.6.3.2.8 "Parameterization" on page 974.

## Connection of analog output loads (Voltage)



Fig. 181: Connection of analog output loads (voltage) to the analog outputs (AO0 ... AO1)

| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | Load $\pm 10 \mathrm{~mA}$ max. | 1 channel used |
| :--- | :--- | :--- | :--- |

For the measuring ranges that can be configured, please refer to the sections Measuring Ranges ${ }^{*}$ Chapter 5.2.6.3.2.11 "Measuring ranges" on page 984 and Parameterization \& Chapter 5.2.6.3.2.8 "Parameterization" on page 974.
Unused analog outputs can be left open-circuited.

## Connection of analog output loads (Current)



Fig. 182: Connection of analog output loads (current) to the analog outputs (AOO ... AO1)

| Current | $0 \ldots 20 \mathrm{~mA}$ | Load $0 \ldots 500 \Omega$ | 1 channel used |
| :--- | :--- | :--- | :--- |
| Current | $4 \ldots 20 \mathrm{~mA}$ | Load $0 \ldots 500 \Omega$ | 1 channel used |

For the measuring ranges that can be configured, please refer to the sections Measuring Ranges $\stackrel{\wedge}{ } \boldsymbol{y}$ Chapter 5.2.6.3.2.11 "Measuring ranges" on page 984 and Parameterization出 Chapter 5.2.6.3.2.8 "Parameterization" on page 974.

Unused analog outputs can be left open-circuited.

## Internal data exchange

| Parameter | Value |
| :--- | :--- |
| Digital inputs (bytes) | 3 |
| Digital outputs (bytes) | 3 |
| Analog inputs (words) | 4 |
| Analog outputs (words) | 2 |
| Counter input data (words) | 4 |
| Counter output data (words) | 8 |

## Addressing

A detailed description concerning addressing can be found in the documentation of ABB Control Builder Plus Software.

The CANopen communication interface module reads the position of the rotary switches only during power-up, i. e. changes of the switch position during operation will have no effect until the next module initialization.

The range of permitted CANopen slave addresses is 1 to 127 . Setting a higher address (> 128) does not lead to an error response, but results in a special mode (DS401). In this special mode, the device creates the node address by subtracting the value 128 from the address switch's value.

## I/O configuration

The CI582-CN CANopen bus configuration is handled by CANopen master with the exception of the slave node ID (via rotary switches) and the transmision rate (automatic detection).

The digital I/O channels and the fast counter are configured via software.

## Parameterization

## Parameters of the module

| Name | Value | Internal value | Internal value, type | Default |
| :---: | :---: | :---: | :---: | :---: |
| Module ID ${ }^{1}$ ) | Internal | 0x1C84 | WORD | 0x1C84 |
| Parameter length | Internal | 54 |  | 54 |
| Error LED / Failsafe function (table error LED / Failsafe function 4y) Further information on page 974) | On | 0 | BYTE | 0 |
|  | Off by E4 | 1 |  |  |
|  | Off by E3 | 2 |  |  |
|  | On + failsafe | 16 |  |  |
|  | Off by E4 + failsafe | 17 |  |  |
|  | Off by E3 + failsafe | 18 |  |  |
| Reserved | 0 | 0 | ARRAY of 24 BYTES |  |
| Check supply | On | 0 | BYTE |  |
| (U | Off | 1 |  | 1 |
| Fast counter | 0 | 0 | BYTE | 0 |
|  | : | : |  |  |
|  | $10^{2}$ ) | 10 |  |  |

[^17]Table 219: Settings "Error LED / Failsafe function"

| Setting | Description |
| :--- | :--- |
| On | Error LED (S-ERR) lights up at errors of all error classes, failsafe <br> mode off |
| Off by E4 | Error LED (S-ERR) lights up at errors of error classes E1, E2 and E3, <br> failsafe mode off |
| Off by E3 | Error LED (S-ERR) lights up at errors of error classes E1 and E2, <br> failsafe mode off |
| On +Failsafe | Error LED (S-ERR) lights up at errors of all error classes, failsafe <br> mode on *) |
| Off by E4 + Failsafe | Error LED (S-ERR) lights up at errors of error classes E1, E2 and E3, <br> failsafe mode on *) |
| Off by E3 + Failsafe | Error LED (S-ERR) lights up at errors of error classes E1 and E2, <br> failsafe mode on *) |
| *) The parameters Behaviour analog outputs at communication error and Behaviour digital <br> outputs at communication error are only evaluated if the failsafe function is enabled. |  |

## Group parameters for the analog part

| Name | Value | Internal value | Internal value, type | Default |
| :---: | :---: | :---: | :---: | :---: |
| Analog data format | Standard Reserved | $\begin{aligned} & \hline 0 \\ & 255 \end{aligned}$ | BYTE | 0 |
| Behavior analog outputs at communication error *) | Off <br> Last value <br> Last value 5 s <br> Last value 10 s <br> Substitute value <br> Substitute value 5 s <br> Substitute value 10 s | $\begin{aligned} & 0 \\ & 1 \\ & 6 \\ & 11 \\ & 2 \\ & 7 \\ & 12 \end{aligned}$ | BYTE | 0 |

Channel parameters for the analog inputs (4x)

| Name | Value | Internal value | Internal value, <br> type | Default |
| :--- | :--- | :--- | :--- | :--- |
| Input 0, Channel <br> configuration | Operation modes <br> of analog inputs | Operation modes <br> of analog inputs | BYTE | 0 |
| Input 0, Check <br> channel | Settings channel <br> monitoring | Settings channel <br> monitoring | BYTE | 0 |
| $:$ | $:$ | $:$ | $:$ | $:$ |
| $:$ | $:$ | $:$ | $:$ | $:$ |


| Name | Value | Internal value | Internal value, <br> type | Default |
| :--- | :--- | :--- | :--- | :--- |
| Input 3, Channel <br> configuration | Operation modes <br> of analog inputs | Operation modes <br> of analog inputs | BYTE | 0 |
| Input 3, Check <br> channel | Settings channel <br> monitoring | Settings channel <br> monitoring | BYTE | 0 |

Table 220: Channel configuration - Operating modes of the analog inputs

| Internal Value | Operating Modes (individually configurable) |
| :---: | :---: |
| 0 (default) | Not used |
| 1 | 0 ... 10 V |
| 2 | Digital input |
| 3 | $0 \ldots 20 \mathrm{~mA}$ |
| 4 | $4 \ldots 20 \mathrm{~mA}$ |
| 5 | -10 V ... +10 V |
| 8 | 2-wire Pt100-50 ${ }^{\circ} \mathrm{C} \ldots+400{ }^{\circ} \mathrm{C}$ |
| 9 | 3-wire Pt100-50 ... $+400{ }^{\circ} \mathrm{C}$ *) |
| 10 | 0 ... 10 V (voltage diff.) *) |
| 11 | -10 V ... +10 V (voltage diff.) *) |
| 14 | 2-wire Pt100-50 ${ }^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ |
| 15 | 3-wire Pt100-50 ${ }^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ *) |
| 16 | 2-wire Pt1000-50 ${ }^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ |
| 17 | 3-wire Pt1000-50 ${ }^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}{ }^{*}$ ) |
| 18 | 2-wire Ni1000-50 ${ }^{\circ} \mathrm{C} \ldots+150{ }^{\circ} \mathrm{C}$ |
| 19 | 3-wire Ni1000-50 ${ }^{\circ} \mathrm{C} \ldots+150{ }^{\circ} \mathrm{C}{ }^{*}$ ) |

*) In the operating modes with 3-wire configuration or with differential inputs, two adjacent analog inputs belong together (e.g. the channels 0 and 1). In these cases, both channels are configured in the desired operating mode. The lower address must be the even address (channel 0). The next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1).

Table 221: Channel monitoring

| Internal Value | Check Channel |
| :--- | :--- |
| 0 (default) | Plausibility, wire break, short circuit |
| 3 | Not used |

## Channel parameters for the analog outputs (2x)

| Name | Value | Internal value | Internal value, <br> type | Default |
| :--- | :--- | :--- | :--- | :--- |
| Output 0, <br> Channel configu- <br> ration | Operation modes <br> of analog outputs | Operation modes <br> of analog outputs | BYTE | 0 |
| Output 0, Check <br> channel | Channel moni- <br> toring | Channel moni- <br> toring | BYTE | 0 |
| Output 0, Substi- <br> tute value | Substitute value | Substitute value | WORD | 0 |
| Output 1, <br> Channel configu- <br> ration | Operation modes <br> of analog outputs | Operation modes <br> of analog outputs | BYTE | 0 |
| Output 1, Check <br> channel | Channel moni- <br> toring | Channel moni- <br> toring | BYTE | 0 |
| Output 1, Substi- <br> tute value | Substitute value | Substitute value | WORD | 0 |

Table 222: Channel configuration - Operating modes of the analog outputs

| Internal value | Operating Modes (individually configu- <br> rable) |
| :--- | :--- |
| 0 (default) | Not used |
| 128 | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ |
| 129 | $0 \mathrm{~mA} . .20 \mathrm{~mA}$ |
| 130 | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ |

Table 223: Channel monitoring

| Internal value | Check channel |
| :--- | :--- |
| 0 | Plausibility, wire break, short circuit |
| 3 | None |

Table 224: Substitute value

| Intended Behavior of Output <br> Channel when the Control <br> System Stops | Required Setting of <br> the Module Parameter <br> "Behavior of Outputs in <br> Case of a Communication <br> Error" | Required Setting of the <br> Channel Parameter "Substi- <br> tute value" |
| :--- | :--- | :--- |
| Output OFF | Off | 0 |
| Last value infinite | Last value | 0 |
| Last value for 5 s and then <br> turn off | Last value 5 sec | 0 |
| Last value for 10 s and then <br> turn off | Last value 10 sec | 0 |
| Substitute value infinite | Substitute value | Depending on configuration |


| Intended Behavior of Output <br> Channel when the Control <br> System Stops | Required Setting of <br> the Module Parameter <br> Behavior of Outputs in <br> Case of a Communication <br> Error" | Required Setting of the <br> Channel Parameter "Substi- <br> tute value" |
| :--- | :--- | :--- |
| Substitute value for 5 s and <br> then turn off | Substitute value 5 sec | Depending on configuration |
| Substitute value for 10 s and <br> then turn off | Substitute value 10 sec | Depending on configuration |

## Group parameters for the digital part

$\left.\begin{array}{|l|l|l|l|l|}\hline \text { Name } & \text { Value } & \text { Internal value } & \begin{array}{l}\text { Internal value, } \\ \text { type }\end{array} & \text { Default } \\ \hline \text { Input delay } & 0.1 \mathrm{~ms} & 0 & \text { BYTE } \\ 1 \mathrm{~ms} \\ 8 \mathrm{~ms} \\ 32 \mathrm{~ms}\end{array}\right)$
${ }^{1}$ ) The parameter Behavior digital outputs at communcation error is only analyzed if the failsafe mode is ON .
${ }^{2}$ ) The state "externally voltage detected" appears if the output of a channel DCO ... DC7 is to be switched on while an external voltage is connected «\% Chapter 5.2.6.3.2.4 "Connections" on page 957. In this case, the start-up is disabled as long as the external voltage is connected. The monitoring of this state and the resulting diagnosis message can be disabled by setting the parameters to "OFF".

| Byte Number | Description | Possible Values |
| :---: | :---: | :---: |
| 1 | Diagnosis byte, slot number | 31 = CI581-CN (e. g. error at integrated 8 DI / 8 DO) <br> 1 = 1st connected S500 I/O module <br> ... <br> $10=10$ th connected S500 I/O module |
| 2 | Diagnosis byte, module number | According to the I/O bus specification passed on by modules to the fieldbus master |
| 3 | Diagnosis byte, channel | According to the I/O bus specification passed on by modules to the fieldbus master |
| 4 | Diagnosis byte, error code | According to the I/O bus specification Bit 7 and bit 6, coded error class $\begin{aligned} & 0=E 1 \\ & 1=E 2 \\ & 2=E 3 \\ & 3=E 4 \end{aligned}$ <br> Bit 0 to bit 5 , coded error description |
| 5 | Diagnosis byte, flags | According to the I/O bus specification <br> Bit 7: 1 = coming error <br> Bit 6: 1 = leaving error |

In cases of short circuit or overload, the digital outputs are turned off. The module performs reactivation automatically. Thus, an acknowledgement of the errors is not necessary. The error message is stored via the LED.

| E1..E4 | d1 | d2 | d3 | d4 | $\begin{array}{\|l\|} \hline \begin{array}{l} \text { Identi- } \\ \text { fier } \end{array} \\ 000 \text {.. } 063 \end{array}$ | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err | $\begin{array}{\|l} \text { PS501 } \\ \text { PLC } \\ \text { Browser } \end{array}$ |  |
| $\begin{array}{\|l} \hline \text { Byte } 4 \\ \text { Bit } \\ 6 \ldots 7 \end{array}$ | - | Byte 1 | Byte 2 | Byte 3 | $\begin{aligned} & \text { Byte 4 } \\ & \text { Bit } \\ & 0 \text {... } 5 \end{aligned}$ | CANope n diagnosis block |  |
| Class | Interface | Device | Module | Channel | Error- <br> Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) | ${ }^{4}$ ) |  |  |  |
| Module errors |  |  |  |  |  |  |  |
| 3 | - | 31 | 31 | 31 | 19 | Checksum error in the I/O module | Replace I/O module |
| 3 | - | 31 | 31 | 31 | 3 | Timeout in the I/O module |  |
| 3 | - | 31 | 31 | 31 | 40 | Different hard-/firmware versions in the module |  |


| E1..E4 | d1 | d2 | d3 | d4 | Identifier $000 \text {.. } 063$ | AC500- ${ }^{\text {A }}$ Display ${ }^{\text {c- Display }}$ | <- Display in |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err | $\begin{array}{\|l\|} \hline \text { PS501 } \\ \text { PLC } \\ \text { Browser } \end{array}$ |  |
| Byte 4 Bit 6... 7 | - | Byte 1 | Byte 2 | Byte 3 | Byte 4 Bit 0 ... 5 | CANope n diagnosis block |  |
| Class | Interface | Device | Module | Channel | Error-Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) | ${ }^{4}$ ) |  |  |  |
| 3 | - | 31 | 31 | 31 | 43 | Internal error in the module |  |
| 3 | - | 31 | 31 | 31 | 36 | Internal data exchange failure |  |
| 3 | - | 31 | 31 | 31 | 9 | Overflow diagnosis buffer | Restart |
| 3 | - | 31 | 31 | 31 | 26 | Parameter error | Check Master |
| 3 | - | 31 | 31 | 31 | 11 | Process voltage UP too low | Check process supply voltage |
| 3 | - | 31 | 31 | 31 | 45 | Process voltage UP gone | Check process supply voltage |
| 3 | - | 31/1... 10 | 31 | 31 | 17 | No communication with I/O module | Replace I/O module |
| 3 | - | 1... 10 | 31 | 31 | 32 | Wrong I/O module type on socket | Replace I/O module / check configuration |
| 4 | - | 1... 10 | 31 | 31 | 31 | At least one module does not support failsafe function | Check modules and parameterization |
| 4 | - | 31 | 31 | 31 | 46 | Voltage feedback on activated digital outputs ${ }^{4}$ ) | Check terminals |
| 4 | - | 31/1... 10 | 31 | 31 | 34 | No response during initialization of the I/O module | Replace I/O module |
| 4 | - | 31 | 31 | 31 | 11 | Process voltage UP3 too low | Check process supply voltage |


| E1..E4 | d1 | d2 | d3 | d4 | Identifier $000 \text {.. } 063$ | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err | $\begin{aligned} & \text { PS501 } \\ & \text { PLC } \\ & \text { Browser } \end{aligned}$ |  |
| $\begin{aligned} & \text { Byte } 4 \\ & \text { Bit } \\ & 6 \text {... } 7 \end{aligned}$ | - | Byte 1 | Byte 2 | Byte 3 | $\begin{array}{\|l} \hline \text { Byte } 4 \\ \text { Bit } \\ 0 \text {... } 5 \end{array}$ | CANope n diagnosis block |  |
| Class | Interface | Device | Module | Channel | Error- <br> Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) | ${ }^{4}$ ) |  |  |  |
| 4 | - | 31 | 31 | 31 | 45 | Process voltage UP3 gone | Check process supply voltage |
| 4 | - | 31 | 31 | 31 | 10 | Voltage overflow on outputs (above UP3 level) ${ }^{5}$ ) | Check terminals/ check process supply voltage |
| Channel error digital |  |  |  |  |  |  |  |
| 4 | - | 31 | 2 | 0... 7 | 46 | Voltage feedback on deactivated digital output ${ }^{6}$ ) | Check terminals |
| 4 | - | 31 | 2 | 0... 7 | 47 | Short circuit at digital output ${ }^{7}$ ) | Check terminals |
| Channel error analog |  |  |  |  |  |  |  |
| 4 | - | 31 | 1 | $0 . .3$ | 48 | Analog value overflow or broken wire at an analog input | Check value or check terminals |
| 4 | - | 31 | 1 | $0 . .3$ | 7 | Analog value underflow at an analog input | Check value |
| 4 | - | 31 | 1 | $0 . .3$ | 47 | Short circuit at an analog input | Check terminals |
| 4 | - | 31 | 3 | $0 . .1$ | 4 | Analog value overflow at an analog output | Check output value |
| 4 | - | 31 | 3 | $0 . .1$ | 7 | Analog value underflow at an analog output | Check output value |

Remarks:

| ${ }^{1}$ ) | In AC500, the following interface identifier applies: <br> "-" = Diagnosis via bus-specific function blocks; 0 ... 4 or $10=$ position of the communication module; $14=1 / \mathrm{O}$ bus; $31=$ module itself <br> The identifier is not contained in the CI541-DP diagnosis block. |
| :---: | :---: |
| ${ }^{2}$ ) | With "Device" the following allocation applies: $31=$ module itself; $1 \ldots 10=$ decentralized communication interface module |
| ${ }^{3}$ ) | With "Module" the following allocation applies: 31 = module itself <br> Channel error: module type ( $1=\mathrm{Al}, 2=\mathrm{DO}, 3=\mathrm{AO}$ ) |
| ${ }^{4}$ ) | This message appears if external voltages at one or more terminals DO0 ... DO7 cause other digital outputs to be fed by that voltage ${ }^{*} \gg$ Chapter 5.2.6.3.2.4 "Connections" on page 957. All outputs of the digital output groups will be turned off for 5 seconds. The diagnosis message appears for the whole output group. |
| ${ }^{5}$ ) | The voltage on digital outputs DO0 ... DO7 has overrun the process supply voltage UP3 ${ }^{*}$ " Chapter 5.2.6.3.2.4 "Connections" on page 957. Diagnosis message appears for the whole module. |
| ${ }^{6}$ ) | This message appears if the output of a channel DOO ... DO7 is to be switched on while an external voltage is connected. In this case, start-up is disabled while the external voltage is connected. Otherwise, this could produce reverse voltage flowing from this output to other digital outputs. This diagnosis message appears for each channel. |
| ${ }^{7}$ ) | Short circuit: After a short circuit has been detected, the output is deactivated for 100 ms seconds. Subsequently, a new start-up will be executed. This diagnosis message appears for each channel. |

## State LEDs

The state LEDs are located at the front of module. There are 2 different groups:

- The 5 system LEDs (PWR, CN-RUN, CN-ERR, S-ERR and I/O bus) show the operation states of the module and display possible errors.
- The 27 process LEDs (UP, UP3, inputs, outputs, CH-ERR1 to CH-ERR3) show the process supply voltage and the states of the inputs and outputs and display possible errors.

States of the 5 system LEDs
$\left.\begin{array}{|l|l|l|l|l|}\hline \text { LED } & \text { Color } & \text { OFF } & \text { ON } & \text { Flashing } \\ \hline \text { PWR/RUN } & \text { Green } & \begin{array}{l}\text { Process supply } \\ \text { voltage missing }\end{array} & \begin{array}{l}\text { Internal supply } \\ \text { voltage OK, } \\ \text { module ready for } \\ \text { communication } \\ \text { with I/O controller }\end{array} & \begin{array}{l}\text { Start-up / pre- } \\ \text { paring communi- } \\ \text { cation }\end{array} \\ \hline \text { CN-RUN } & \text { Green } & ---- & --- & \begin{array}{l}\text { Device config- } \\ \text { ured, CANopen } \\ \text { bus in OPERRA- } \\ \text { TIONAL state } \\ \text { and cyclic data } \\ \text { exchange run- } \\ \text { ning }\end{array}\end{array} \begin{array}{l}\text { Flashing: } \\ \text { CANopen bus in } \\ \text { PRE-OPERA- } \\ \text { TIONAL state } \\ \text { and slave is } \\ \text { being configured } \\ \text { Single flash: } \\ \text { CANopen bus in } \\ \text { STOPPED state. } \\ \text { Flickering: Auto- } \\ \text { detect is active }\end{array}\right\}$

| LED | Color | OFF | ON | Flashing |
| :--- | :--- | :--- | :--- | :--- |
| CN-ERR | Red | No system error | CANopen Bus is <br> OFF | Flashing: Config- <br> uration error <br> Single flash: error <br> counter overflow <br> due to too many <br> error frames <br> Double flash: A <br> node-guard or a <br> heartbeat event <br> occurred <br> Flickering: Auto- <br> detect is active |
| S-ERR | Red | No error | Internal error | -- |
| I/O bus | Green | No decentralized <br> I/O modules con- <br> nected or com- <br> munication error | Decentralized I/O <br> modules con- <br> nected and <br> operational | --- |

States of the 27 process LEDs:

| LED | Color | OFF | ON | Flashing |
| :--- | :--- | :--- | :--- | :--- |
| AIO ... AI3 | Yellow | Input is OFF | Input is ON <br> (brightness <br> depends on the <br> value of the <br> analog signal) | -- |
| AO0 ... AO1 | Yellow | Output is OFF | Output is ON <br> (brightness <br> depends on the <br> value of the <br> analog signal) | -- |
| DIO ... DI7 | Yellow | Input is OFF | Input is ON (the <br> input voltage is <br> even displayed if <br> the supply <br> voltage is OFF) | -- |
| DO0 ... DO7 | Yellow | Output is OFF | Output is ON | -- |
| UP | Green | Process supply <br> voltage missing | Process supply <br> voltage OK and <br> initialization fin- <br> ished | -- |
| UP3 | Green | Process supply <br> voltage missing | Process supply <br> voltage OK | -- |
| CH-ERR1 to CH- <br> ERR3 | Red | No error or <br> process supply <br> voltage missing | Internal error | Error on one <br> channel of the <br> corresponding <br> group |

## Measuring ranges

Input ranges voltage, current and digital input

| Range | 0 V ... +10 | $-10 \mathrm{~V} . .$ | $0 \mathrm{~mA} \ldots$ | $4 \mathrm{~mA} . . .20$ | Digital | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  | Decimal | Hex. |
| Overflow | > 11.7589 | > 11.7589 | > 23.5178 | > 22.8142 |  | 32767 | 7FFF |
| Measured value too high | $\begin{aligned} & 11.7589 \\ & : \\ & 10.0004 \end{aligned}$ |  | 23.5178 <br> 20.0007 |  |  | 32511 $:$ 27649 | $\begin{aligned} & \text { 7EFF } \\ & : \\ & 6 \mathrm{C} 01 \end{aligned}$ |
| Normal range <br> Normal range or | $\begin{aligned} & 10.0000 \\ & : \\ & 0.0004 \end{aligned}$ | $\begin{aligned} & 10.0000 \\ & : \\ & 0.0004 \end{aligned}$ | $\begin{aligned} & 20.0000 \\ & : \\ & 0.0007 \end{aligned}$ | $\begin{aligned} & 20.0000 \\ & : \\ & 4.0006 \end{aligned}$ | On | $27648$ | $\begin{aligned} & 6 \mathrm{C} 00 \\ & : \\ & 0001 \end{aligned}$ |
| measured | 0.0000 | 0.0000 | 0 | 4 | Off | 0 | 0000 |
| value too low | $\begin{array}{l\|} \hline-0.0004 \\ -1.7593 \end{array}$ | $\begin{aligned} & -0.0004 \\ & : \\ & : \\ & -10,0000 \end{aligned}$ |  | $\begin{aligned} & 3.9994 \\ & 1.1858 \end{aligned}$ |  | $\begin{aligned} & \hline-1 \\ & -4864 \\ & : \\ & -27648 \end{aligned}$ | $\begin{aligned} & \text { FFFF } \\ & \text { ED00 } \\ & : \\ & 9400 \end{aligned}$ |
| Measured value too low |  | -10.0004 <br> -11.7589 |  |  |  | $\begin{aligned} & -27649 \\ & : \\ & -32512 \\ & \hline \end{aligned}$ | 93FF $:$ 8100 |
| Underflow | < 1.7593 | <-11.7589 | < 0.0000 | < 1.1858 |  | -32768 | 8000 |

The represented resolution corresponds to 16 bits.

Input ranges resistance temperature detector

| Range | Pt100 / Pt1000 | Ni1000 | Digital value |  |
| :---: | :---: | :---: | :---: | :---: |
|  |  |  | Decimal | Hex. |
| Overflow | $>+450.0{ }^{\circ} \mathrm{C}$ | > $+160.0{ }^{\circ} \mathrm{C}$ | 32767 | 7FFF |
| Measured value too high | $\begin{aligned} & +450.0^{\circ} \mathrm{C} \\ & : \\ & +400.1^{\circ} \mathrm{C} \end{aligned}$ |  | $\begin{array}{\|l\|} \hline 4500 \\ \vdots \\ 4001 \\ \hline \end{array}$ | $\begin{aligned} & 1194 \\ & : \\ & \text { OFA1 } \end{aligned}$ |
|  |  | $\begin{aligned} & +160.0^{\circ} \mathrm{C} \\ & : \\ & +150.1^{\circ} \mathrm{C} \end{aligned}$ | $\begin{array}{\|l\|} \hline 1600 \\ : \\ 1501 \\ \hline \end{array}$ | $\begin{aligned} & 0640 \\ & : \\ & \text { 05DD } \end{aligned}$ |
|  |  |  | $\begin{aligned} & 800 \\ & \vdots \\ & 701 \end{aligned}$ | $\begin{aligned} & 0320 \\ & \vdots \\ & \text { 02BD } \end{aligned}$ |


| Range | $\begin{array}{\|l\|} \hline \mathrm{Pt} 100 / \mathrm{Pt} 1000 \\ -50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C} \end{array}$ | $\begin{aligned} & \mathrm{Ni} 1000 \\ & -50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C} \end{aligned}$ | Digital value |  |
| :---: | :---: | :---: | :---: | :---: |
|  |  |  | Decimal | Hex. |
| Normal range | $\begin{aligned} & +400.0^{\circ} \mathrm{C} \\ & : \\ & : \\ & : \\ & +0.1^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & +150.0^{\circ} \mathrm{C} \\ & : \\ & +0.1^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \hline 4000 \\ & 1500 \\ & 700 \\ & : \\ & 1 \end{aligned}$ | $\begin{aligned} & \hline 0 \mathrm{FAO} \\ & 05 \mathrm{DC} \\ & 02 \mathrm{BC} \\ & : \\ & 0001 \end{aligned}$ |
|  | $0.0{ }^{\circ} \mathrm{C}$ | $0.0{ }^{\circ} \mathrm{C}$ | 0 | 0000 |
|  | $\begin{aligned} & -0.1^{\circ} \mathrm{C} \\ & : \\ & -50.0^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & -0.1^{\circ} \mathrm{C} \\ & : \\ & -50,0^{\circ} \mathrm{C} \end{aligned}$ | \|-1 -500 | $\begin{aligned} & \text { FFFF } \\ & \text { : } \\ & \text { FEOC } \end{aligned}$ |
| Measured value too low | $\begin{aligned} & -50.1^{\circ} \mathrm{C} \\ & : \\ & -60.0^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & -50.1^{\circ} \mathrm{C} \\ & : \\ & -60.0^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \hline-501 \\ & : \\ & -600 \end{aligned}$ | $\begin{aligned} & \text { FEOB } \\ & : \\ & \text { FDA8 } \end{aligned}$ |
| Underflow | $<-60.0{ }^{\circ} \mathrm{C}$ | $<-60.0{ }^{\circ} \mathrm{C}$ | -32768 | 8000 |

## Output ranges voltage and current

| Range | -10 V ...+10 V | $\begin{aligned} & 0 \mathrm{~mA} \ldots 20 \\ & \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 4 \mathrm{~mA} \ldots 20 \\ & \mathrm{~mA} \end{aligned}$ | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  | Decimal | Hex. |
| Overflow | 0 V | 0 mA | 0 mA | > 32511 | > 7EFF |
| Measured value too high | $\begin{aligned} & 11.7589 \mathrm{~V} \\ & : \\ & 10.0004 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 23.5178 \mathrm{~mA} \\ & : \\ & 20.0007 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 22.8142 \mathrm{~mA} \\ & : \\ & 20.0006 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 32511 \\ & : \\ & 27649 \end{aligned}$ | $\begin{aligned} & 7 \mathrm{EFF} \\ & : \\ & 6 \mathrm{C} 01 \end{aligned}$ |
| Normal range | $\begin{aligned} & 10.0000 \mathrm{~V} \\ & : \\ & 0.0004 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 20.0000 \mathrm{~mA} \\ & : \\ & 0,0007 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 20.0000 \mathrm{~mA} \\ & : \\ & 4.0006 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 27648 \\ & : \\ & 1 \end{aligned}$ | $\begin{aligned} & \text { 6C00 } \\ & : \\ & 0001 \end{aligned}$ |
|  | 0.0000 V | 0.0000 mA | 4.0000 mA | 0 | 0000 |
|  | $\begin{aligned} & -0.0004 \mathrm{~V} \\ & : \\ & -10.0000 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 0 \mathrm{~mA} \\ & : \\ & 0 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 3.9994 \mathrm{~mA} \\ & 0 \mathrm{~mA} \\ & 0 \mathrm{~mA} \end{aligned}$ | $\begin{array}{\|l} \hline-1 \\ -6912 \\ -27648 \end{array}$ | $\begin{aligned} & \text { FFFF } \\ & \text { E500 } \\ & 9400 \end{aligned}$ |
| Measured value too low | $\begin{aligned} & -10.0004 \mathrm{~V} \\ & : \\ & -11.7589 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 0 \mathrm{~mA} \\ & : \\ & 0 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 0 \mathrm{~mA} \\ & : \\ & 0 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & -27649 \\ & : \\ & -32512 \end{aligned}$ | $\begin{aligned} & 93 F F \\ & : \\ & 8100 \end{aligned}$ |
| Underflow | 0 V | 0 mA | 0 mA | <-32512 | < 8100 |

The represented resolution corresponds to 16 bits.

## Technical data

## Technical data of the module

The system data of AC500 and S500 are applicable to the standard version Chapter 5.1.2 "System data AC500" on page 166.
The system data of AC500-XC are applicable to the XC version $\Leftrightarrow$ Chapter 5.1.3 "System data AC500-XC" on page 169.

Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

## Multiple overloads

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an internal smart high-side switch.

| Parameter | Value |
| :---: | :---: |
| Interface | CAN |
| Protocol | CANopen |
| Power supply | From the process supply voltage UP |
| Supply of the electronic circuitry of the I/O modules attached | Through the I/O bus interface (I/O bus) |
| Rotary switches | For setting the CANopen Node ID for configuration purposes (00h to FFh) |
| LED displays | For system displays, signal states, errors and power supply |
| External supply voltage | Via terminals ZP, UP and UP3 (process supply voltage 24 V DC) |
| Transmission rates | 10 / 20 / 50 / 125 / 250 / 500 / 800 kbit/s 1 Mbit/s Auto transmission rate detection is supported |
| Bus connection | Depending on used terminal unit TU510: 9-pin D-sub connector TU518: 10-pin terminal block |
| Processor | Hilscher NETX 100 |
| Expandability | CI58x can only be used on onboard CAN interface and without any I/O expansion module « Table 215 "CANopen" on page 952. |
| State display | Module state: PWR/RUN, CN-RUN, CN-ERR, E-ERR, I/O bus |
| Adjusting elements | 2 rotary switches for generation of the node address |
| Ambient temperature | System data AC500 « Chapter 5.1.2 "System data AC500" on page 166 <br> System data AC500 XC $\Leftrightarrow$ Chapter 5.1.3 "System data AC500-XC" on page 169 |
| Current consumption | UP: 0.2 A UP3: 0.06 A + 0.5 A max. per output |
| Weight (without terminal unit) | Ca. 125 g |
| Process supply voltages UP/UP3 |  |
| Rated value | 24 V DC (for inputs and outputs) |


| Parameter | Value |
| :---: | :---: |
| Max. load for the terminals | 10 A |
| Protection against reversed voltage | Yes |
| Rated protection fuse on UP/UP3 | 10 A fast |
| Galvanic isolation | CANopen interface against the rest of the module |
| Inrush current from UP (at power up) | On request |
| Current consumption via UP (normal operation) | 0.2 A |
| Current consumption via UP3 | 0.06 A + 0.5 A max. per output |
| Connections | Terminals 2.8 and 3.8 for +24 V (UP) <br> Terminal 4.8 for +24 V (UP3) <br> Terminals 2.9, 3.9 and 4.9 for $0 \mathrm{~V}(\mathrm{ZP})$ |
| Max. power dissipation within the module | 6 W |
| Reference potential for all digital inputs and outputs | Negative pole of the supply voltage, signal name ZP |
| Setting of the CANopen Node ID identifier | With 2 rotary switches at the front side of the module |
| Mounting position | Horizontal <br> Or vertical with derating (output load reduced to $50 \%$ at $+40{ }^{\circ} \mathrm{C}$ per group) |
| Cooling | The natural convection cooling must not be hindered by cable ducts or other parts in the control cabinet. |
| Effect of incorrect input terminal connection | Wrong or no signal detected, no damage up to 35 V |
| Required terminal unit | TU509, TU510, TU517 or TU518 <br> \& Chapter 5.2.7.4 "TU517 and TU518 for communication interface modules" on page 1278 |

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and continuous overvoltage up to 30 V DC.

## Technical data of the digital inputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 8 |
| Distribution of the channels into groups | 1 group of 8 channels |
| Terminals of the channels DIO ... DI7 | Terminals $3.0 \ldots 3.7$ |
| Reference potential for all inputs | Terminals $2.9 \ldots 4.9$ (negative pole of the <br> supply voltage, signal name ZP) |
| Indication of the input signals | 1 yellow LED per channel, the LED is ON <br> when the input signal is high (signal 1) |
| Input type (according EN 61131-2) | Type 1 |


| Parameter | Value |
| :--- | :--- |
| Input delay (0->1 or 1->0) | Typ. 0.1 ms, configurable from <br> $0.1 \mathrm{~ms} \ldots 32 \mathrm{~ms}$ |
| Input signal voltage | 24 V DC |
| Signal 0 | $-3 \mathrm{~V} \ldots+5 \mathrm{~V}$ |
| Undefined signal | $>+5 \mathrm{~V} \ldots<+15 \mathrm{~V}$ |
| Signal 1 | $+15 \mathrm{~V} \ldots+30 \mathrm{~V}$ |
| Ripple with signal 0 | Within $-3 \mathrm{~V} \ldots+5 \mathrm{~V}$ |
| Ripple with signal 1 | Within $+15 \mathrm{~V} \ldots+30 \mathrm{~V}$ |
| Input current per channel |  |
| Input voltage +24 V | Typ. 5 mA |
| Input voltage +5 V | $>1 \mathrm{~mA}$ |
| Input voltage +15 V | $>2 \mathrm{~mA}$ |
| Input voltage +30 V | $<8 \mathrm{~mA}$ |
| Max. cable length |  |
| Shielded | 1000 m |
| Unshielded | 600 m |

## Technical data of the digital outputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 8 |
| Distribution of the channels into groups | 1 group of 8 channels |
| Terminals of the channels DO0 to DO7 | Terminals 4.0 to 4.7 |
| Reference potential for all outputs | Terminals $2.9 \ldots 4.9$ (negative pole of the <br> supply voltage, signal name ZP) |
| Common power supply voltage | For all outputs terminal 4.8 (positive pole of <br> the supply voltage, signal name UP3) |
| Output voltage for signal 1 | UP3 (-0.8 V) |
| Output delay (0->1 or 1->0) | On request |
| Output current | 500 mA at UP3 $=24 \mathrm{~V}$ |
| Rated value per channel | 4 A |
| Max. value (all channels together) | $<0.5 \mathrm{~mA}$ |
| Leakage current with signal 0 | 10 A fast |
| Fuse for UP3 | Via internal varistors (see figure below this <br> table) |
| Demagnetization with inductive DC load |  |
| Output switching frequency | On request |
| With resistive load | Max. 0.5 Hz |
| With inductive loads | 11 Hz max. at 5 W max. |
| With lamp loads | Yes |
| Short-circuit-proof / overload-proof |  |


| Parameter | Value |
| :--- | :--- |
| Overload message (I > 0.7 A) | Yes, after ca. 100 ms |
| Output current limitation | Yes, automatic reactivation after short cir- <br> cuit/overload |
| Resistance to feedback against 24 V signals | Yes (software-controlled supervision) |
| Max. cable length |  |
| Shielded | 1000 m |
| Unshielded |  |

Fig. 183: Circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off

1 Digital output
2 Varistors for demagnetization when inductive loads are turned off

## Technical data of the analog inputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 4 |
| Distribution of channels into groups | 1 group with 4 channels |
| Connection if channels $\mathrm{AlO}+$ to $\mathrm{Al3+}$ | Terminals 2.0 to2.3 |
| Reference potential for $\mathrm{AlO}+$ to $\mathrm{Al3+}$ | Terminal 2.4 (AI-) for voltage and RTD measurement <br> Terminal 2.9, 3.9 and 4.9 for current measurement |
| Input type |  |
| Unipolar | Voltage 0... 10 V, current or Pt100/Pt1000/ Ni1000 |
| Bipolar | Voltage -10...+10 V |
| Galvanic isolation | Against CANopen Bus |
| Configurability | 0... 10 V, -10...+10 V, 0/4... $20 \mathrm{~mA}, \mathrm{Pt} 100 / 1000$, Ni1000 (each input can be configured individually) |
| Channel input resistance | Voltage: > $100 \mathrm{k} \Omega$ <br> Current: ca. $330 \Omega$ |
| Time constant of the input filter | Voltage: $100 \mu \mathrm{~s}$ Current: $100 \mu \mathrm{~s}$ |
| Indication of the input signals | 1 LED per channel (brightness depends on the value of the analog signal) |


| Parameter | Value |
| :---: | :---: |
| Conversion cycle | 1 ms (for 4 inputs +2 outputs); with RTDs Pt/ $\mathrm{Ni} . . \mathrm{I}$ s |
| Resolution | Range 0... $10 \mathrm{~V}: 12$ bits <br> Range $-10 \ldots+10 \mathrm{~V}$ : 12 bits including sign <br> Range 0... $20 \mathrm{~mA}: 12$ bits <br> Range 4... $20 \mathrm{~mA}: 12$ bits <br> Range RTD (Pt100, PT1000, Ni1000): $+0.1^{\circ} \mathrm{C}$ |
| Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range | Typ. 0.5 \%, max. 1 \% |
| Relationship between input signal and hex code | ③) Chapter 5.2.6.3.2.11.2 "Input ranges resistance temperature detector" on page 984 |
| Unused inputs | Are configured as "unused" (default value) |
| Overvoltage protection | Yes |

## Technical data of the analog inputs if used as digital inputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | Max. 4 |
| Distribution of channels into groups | 1 group of 4 channels |
| Connections of the channels AI0+ to Al3+ | Terminals 2.0 to 2.3 |
| Reference potential for the inputs | Terminals 2.9, 3.9 and 4.9 (ZP) |
| Indication of the input signals | 1 LED per channel |
| Input signal voltage | 24 VDC |
| Signal 0 | -30 V... +5 V |
| Undefined signal | +5V...+15 V |
| Signal 1 | +15 V...+30 V |
| Input current per channel |  |
| Input voltage +24 V | Typ. 7 mA |
| Input voltage +5 V | Typ. 1.4 mA |
| Input voltage +15 V | Typ. 3.7 mA |
| Input voltage +30 V | < 9 mA |
| Input resistance | Ca. $3.5 \mathrm{k} \Omega$ |

## Technical data of the analog outputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 2 |
| Distribution of channels into groups | 1 group for 2 channels |
| Connection of the channels AO0+...AO1+ | Terminals 1.5...1.6 |


| Parameter | Value |
| :---: | :---: |
| Reference potential for $\mathrm{AO} 0+$ to $\mathrm{AO} 1+$ | Terminal 2.7 (AO-) for voltage output Terminal 2.9, 3.9 and 4.9 for current output |
| Output type |  |
| Unipolar | Current |
| Bipolar | Voltage |
| Galvanic isolation | Against internal supply and other modules |
| Configurability | $-10 \ldots+10 \mathrm{~V}, 0 \ldots 20 \mathrm{~mA}, 4 \ldots 20 \mathrm{~mA}$ (each output can be configured individually) |
| Output resistance (load), as current output | 0... $500 \Omega$ |
| Output loadability, as voltage output | $\pm 10$ mA max. |
| Indication of the output signals | 1 LED per channel (brightness depends on the value of the analog signal) |
| Resolution | 12 bits including sign |
| Settling time for full range change (resistive load, output signal within specified tolerance) | Typ. 5 ms |
| Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range | Typ. 0.5 \%, max. 1 \% |
| Relationship between input signal and hex code | ③ Chapter 5.2.6.3.2.11.3 "Output ranges voltage and current" on page 985 |
| Unused outputs | Are configured as "unused" (default value) and can be left open-circuited |

## Technical data of the fast counter

| Parameter | Value |
| :--- | :--- |
| Used inputs | Terminal 3.0 (DI0), 3.1 (DI1) |
| Used outputs | Terminal 4.0 (DO0) |
| Counting frequency | Depending on operation mode: <br> Mode 1-6: max. 200 kHz <br> Mode 7: max. 50 kHz <br> Mode 9: max. 35 kHz <br> Mode 10: max. 20 kHz |
| Detailed description | Fast Counter « Chapter 5.2.4.3.2.9 "Fast <br> counter" on page 606 |
| Operating modes | Operating modes 乡 Chapter 5.2.4.3.2.9 "Fast <br> counter" on page 606 |

## Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm

The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 228 100 <br> R0001 | CI581-CN, CANopen <br> communication interface module with <br> 8 DI, 8 DO, 4 AI and 2 AO | Active |
| 1SAP 428 100 <br> R0001 | CI581-CN-XC, CANopen <br> communication interface module with <br> 8 DI, 8 DO, 4 AI and 2 AO, XC version | Active |

*) Modules in lifecycle Classic are available from stock but not recommended
for planning and commissioning of new installations.

### 5.2.6.3.3 CI582-CN

Features

- 8 digital inputs 24 V DC
- 8 digital outputs $24 \mathrm{~V} \mathrm{DC}, 0.5 \mathrm{~A}$ max.
- 8 configurable digital inputs/outputs $24 \mathrm{~V} D C, 0.5 \mathrm{~A}$ max.
- Module-wise galvanically isolated
- Fast counter
- XC version for use in extreme ambient conditions available


1 I/O bus
2 Allocation between terminal number and signal name
38 yellow LEDs to display the signal states of the configurable digital inputs/outputs (DC0 ... DC7)
48 yellow LEDs to display the signal states of the digital inputs (DI8 ... DI15)
58 yellow LEDs to display the signal states of the digital outputs (DO8 ... DO15)
62 green LEDs to display the supply voltage UP and UP3
73 red LEDs to display errors (CH-ERR1, CH-ERR2, CH-ERR3)
85 System LEDs: PWR/RUN, CN-RUN, CN-ERR, S-ERR, I/O-Bus
9 Label
102 rotary switches for setting the CANopen node ID
1110 terminals to connect the CANopen bus signals
12 Terminal unit
13 DIN rail


## Intended purpose

The CANopen communication interface module $\mathrm{Cl} 582-\mathrm{CN}$ is used as decentralized I/O module in CANopen networks. Depending on the terminal unit used, the network connection is performed either via a female 9-pin D-sub connector or via 10 terminals (screw or spring terminals) which are integrated in the terminal unit. The communication interface module contains 24 I/O channels with the following properties:

- 8 digital configurable inputs/outputs in 1 group (1.0 ... 1.7)
- 8 digital inputs 24 V DC in 1 group (2.0 ... 2.7)
- 8 digital outputs 24 V DC in 1 group (3.0 ... 3.7)

The inputs/outputs are galvanically isolated from the CANopen network. There is no potential separation between the channels. The configuration of the analog inputs/outputs is performed by software.
For use in extreme ambient conditions (e.g. wider temperature and humidity range), a special XC version of the device is available.

## Functionality

| Parameter | Value |
| :---: | :---: |
| Interface | CAN |
| Protocol | CANopen |
| Power supply | From the process supply voltage UP |
| Supply of the electronic circuitry of the I/O modules attached | Through the I/O bus interface (I/O bus) |
| Rotary switches | For setting the CANopen Node ID for configuration purposes (00h to FFh) |
| LED displays | For system displays, signal states, errors and power supply |
| External supply voltage | Via terminals ZP, UP and UP3 (process supply voltage 24 V DC) |
| Transmission rates | 10 / 20 / 50 / 125 / $250 / 500 / 800$ kbit/s 1 Mbit/s Auto transmission rate detection is supported |
| Bus connection | Depending on used terminal unit TU510: 9-pin D-sub connector TU518: 10-pin terminal block |
| Processor | Hilscher NETX 100 |
| Expandability | CI58x can only be used on onboard CAN interface and without any I/O expansion module « Table 215 "CANopen" on page 952. |
| State display | Module state: PWR/RUN, CN-RUN, CN-ERR, E-ERR, I/O bus |
| Adjusting elements | 2 rotary switches for generation of the node address |
| Ambient temperature | System data AC500 ${ }^{\text {y }}$ Chapter 5.1.2 "System data AC500" on page 166 <br> System data AC500 XC $\stackrel{y}{c}$ Chapter 5.1.3 "System data AC500-XC" on page 169 |
| Current consumption | UP: 0.2 A UP3: 0.06 A + 0.5 A max. per output |
| Weight (without terminal unit) | Ca. 125 g |


| Parameter | Value |
| :---: | :---: |
| Process supply voltages UP/UP3 |  |
| Rated value | 24 V DC (for inputs and outputs) |
| Max. load for the terminals | 10 A |
| Protection against reversed voltage | Yes |
| Rated protection fuse on UP/UP3 | 10 A fast |
| Galvanic isolation | CANopen interface against the rest of the module |
| Inrush current from UP (at power up) | On request |
| Current consumption via UP (normal operation) | 0.2 A |
| Current consumption via UP3 | 0.06 A + 0.5 A max. per output |
| Connections | Terminals 2.8 and 3.8 for +24 V (UP) <br> Terminal 4.8 for +24 V (UP3) <br> Terminals 2.9, 3.9 and 4.9 for $0 \mathrm{~V}(\mathrm{ZP})$ |
| Max. power dissipation within the module | 6 W |
| Reference potential for all digital inputs and outputs | Negative pole of the supply voltage, signal name ZP |
| Setting of the CANopen Node ID identifier | With 2 rotary switches at the front side of the module |
| Mounting position | Horizontal <br> Or vertical with derating (output load reduced to $50 \%$ at $+40^{\circ} \mathrm{C}$ per group) |
| Cooling | The natural convection cooling must not be hindered by cable ducts or other parts in the control cabinet. |
| Effect of incorrect input terminal connection | Wrong or no signal detected, no damage up to 35 V |
| Required terminal unit | TU509, TU510, TU517 or TU518 <br> * Chapter 5.2.7.4 "TU517 and TU518 for communication interface modules" on page 1278 |

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and continuous overvoltage up to 30 V DC.

CI582-CN: Input/ Output characteristics

| Parameter | Value |
| :--- | :--- |
| Inputs and outputs | 8 digital inputs (24 V DC) |
|  | 8 digital transistor outputs (24 V DC, 0.5 A |
|  | max.) |
|  | 8 configurable digital inputs/outputs (24 V DC, |
|  | 0.5 A max.) |

## Connections

## General

The CANopen communication interface module is plugged on the I/O terminal units TU517 « Chapter 5.2.7.4 "TU517 and TU518 for communication interface modules" on page 1278 or TU518 $\Leftrightarrow$ Chapter 5.2.7.4 "TU517 and TU518 for communication interface modules" on page 1278 and accordingly TU509 or TU510. Properly position the module and press until it locks in place.

The connection of the I/O channels is established using the 30 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

The terminals $2.8,3.8,2.9,3.9$ and 4.9 are electrically interconnected within the terminal unit and always have the same assignment, irrespective of the inserted module:

Terminals 2.8 and 3.8: process supply voltage UP $=+24 \mathrm{~V}$ DC
Terminal 4.8: process supply voltage UP3 $=+24 \mathrm{~V}$ DC
Terminals 2.9, 3.9 and 4.9: process supply voltage $\mathrm{ZP}=0 \mathrm{~V}$

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

With a separate UP3 power supply, the digital outputs can be switched off externally. This way, an emergency-off functionality can be realized.

## Possibilities of connection

Mounting on ter- The assignment of the 9-pin female D-sub for the CANopen signals minal units TU509 or TU510

|  | 1 | --- | Reserved |
| :---: | :---: | :---: | :---: |
|  | 2 | CAN- | Inverted signal of the CAN bus |
|  | 3 | CAN_GND | Ground potential of the CAN bus |
|  | 4 | --- | Reserved |
|  | 5 | --- | Reserved |
|  | 6 | --- | Reserved |
|  | 7 | CAN+ | Non-inverted signal of the CAN bus |
|  | 8 | --- | Reserved |
|  | 9 | --- | Reserved |
|  | Shield | Cable shield | Functional earth |

Bus terminating The ends of the data lines have to be terminated with a $120 \Omega$ bus terminating resistor. The bus resistors


Fig. 184: CANopen interface, bus terminating resistors connected to the line ends

| 1 | CAN_GND |
| :--- | :--- |
| 2 | CAN_L |
| 3 | Shield |
| 4 | CAN_H |
| 5 | Data line, shielded twisted pair |
| 6 | COMBICON connection, CANopen interface |



Fig. 185: DeviceNet interface, bus terminating resistors connected to the line ends

| 6 | DeviceNet power supply |
| :--- | :--- |
| 7 | COMBICON connection, DeviceNet interface |
| 8 | Data lines, twisted pair cables |
| 9 | red |


| 10 | black |
| :--- | :--- |
| 11 | white |
| 12 | blue |
| 13 | bare |

The grounding of the shield should take place at the switchgear $«$ Chapter
5.1.2 "System data AC500" on page 166.

Mounting on ter- Table 225: Assignment of the terminals minal units TU517 or TU518

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 1.0 | CAN+ | Non-inverted signal of the CAN bus |
| 1.1 | CAN+ | Non-inverted signal of the CAN bus |
| 1.2 | CAN- | Inverted signal of the CAN bus |
| 1.3 | Term + | Inverted signal of the CAN bus |
| 1.4 | Term- | CAN bus termination for CAN+ (for bus termination, <br> Term+ must be connected with CAN+) |
| 1.5 | CAN bus termination for CAN+ (connecting alterna- <br> tive for terminal 1.4) |  |
| 1.6 | CAN-GND | CAN bus termination for CAN- (for bus termination, <br> Term- must be connected with CAN-) |
| 1.7 | CAN-GND | CAN bus termination for CAN- (connecting alterna- <br> tive for terminal 1.6) |
| 1.8 | Ground potential of the CAN bus |  |
| 1.9 | Ground potential of the CAN bus |  |

At the line ends of a bus segment, terminating resistors must be connected. If TU517 or TU518 is used, the bus terminating resistors can be enabled by connecting the terminals Term+ and Term- to the data lines CAN+ and CAN- (no external terminating resistors are required, see figure below).
The following figures show the different connection options for the CANopen communication interface module:


In the case of TU517/TU518, the terminating resistors are not located inside the TU but inside the communication interface module CI581-CN. Hence, when removing the device from the TU, the bus terminating resistors are no longer connected to the bus. The bus itself will not be disconnected if a device is removed.

The grounding of the shield should take place at the control cabinet. Please refer to the AC500 System-Data $\stackrel{\text { ® }}{ }$ Chapter 5.1.2 "System data AC500" on page 166.

Table 226: Assignment of the other terminals

| Terminal | Signal | Description |
| :---: | :---: | :---: |
| 2.0 | DC0 | Signal of the configurable digital input/output DC0 |
| 2.1 | DC1 | Signal of the configurable digital input/output DC1 |
| 2.2 | DC2 | Signal of the configurable digital input/output DC2 |
| 2.3 | DC3 | Signal of the configurable digital input/output DC3 |
| 2.4 | DC4 | Signal of the configurable digital input/output DC4 |
| 2.5 | DC5 | Signal of the configurable digital input/output DC5 |
| 2.6 | DC6 | Signal of the configurable digital input/output DC6 |
| 2.7 | DC7 | Signal of the configurable digital input/output DC7 |
| 2.8 | UP | Process voltage UP (24 V DC) |
| 2.9 | ZP | Process voltage ZP (0 V DC) |
| 3.0 | DI8 | Signal of the digital input DI8 |
| 3.1 | DI9 | Signal of the digital input DI9 |
| 3.2 | DI10 | Signal of the digital input DI10 |
| 3.3 | DI11 | Signal of the digital input DI11 |
| 3.4 | DI12 | Signal of the digital input DI12 |
| 3.5 | DI13 | Signal of the digital input DI13 |
| 3.6 | DI14 | Signal of the digital input DI14 |
| 3.7 | DI15 | Signal of the digital input DI15 |
| 3.8 | UP | Process voltage UP (24 V DC) |
| 3.9 | ZP | Process voltage ZP (0 V DC) |
| 4.0 | DO8 | Signal of the digital output DO8 |
| 4.1 | DO9 | Signal of the digital output DO9 |
| 4.2 | DO10 | Signal of the digital output DO10 |
| 4.3 | DO11 | Signal of the digital output DO11 |
| 4.4 | DO12 | Signal of the digital output DO12 |
| 4.5 | DO13 | Signal of the digital output DO13 |
| 4.6 | DO14 | Signal of the digital output DO14 |
| 4.7 | DO15 | Signal of the digital output DO15 |
| 4.8 | UP3 | Process voltage UP3 (24 V DC) |
| 4.9 | ZP | Process voltage ZP (0 V DC) |

## WARNING!

## Removal/Insertion under power

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.
Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.
The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

Risk of damaging the PLC modules!
Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.


Fig. 186: Connection of the communication interface module Cl582-CN
For a description of the meaning of the LEDs, please refer to the section for the state LEDs * Chapter 5.2.6.3.3.10 "State LEDs" on page 1010.

Bus length The maximum possible bus length of a CAN network depends on bit rate (transmission rate) and cable type. The sum of all bus segments must not exceed the maximum bus length

| Bit Rate (speed) | Bus Length |
| :--- | :--- |
| $1 \mathrm{Mbit} / \mathrm{s}$ | 40 m |
| $800 \mathrm{kbit} / \mathrm{s}$ | 50 m |
| $500 \mathrm{kbit} / \mathrm{s}$ | 100 m |
| $250 \mathrm{kbit} / \mathrm{s}$ | 250 m |
| $125 \mathrm{kbit} / \mathrm{s}$ | 500 m |
| $50 \mathrm{kbit} / \mathrm{s}$ | 1000 m |

## Connection of the digital inputs



Fig. 187: Connection of the digital inputs (DI8 ... DI15) to the module C/582-CN

## Connection of the digital outputs



Fig. 188: Connection of configurable digital outputs (DO8 ... DO15) to the module CI582-CN

## Connection of the configurable digital inputs/outputs



Fig. 189: Connection of configurable digital inputs/outputs (DCO ... DC7) to the module CI582CN

## Internal data exchange

| Parameter | Value |
| :--- | :--- |
| Digital inputs (bytes) | 5 |
| Digital outputs (bytes) | 5 |
| Counter input data (words) | 4 |
| Counter output data (words) | 8 |

## Addressing

A detailed description concerning addressing can be found in the documentation of ABB Control Builder Plus Software.

The CANopen communication interface module reads the position of the rotary switches only during power-up, i. e. changes of the switch position during operation will have no effect until the next module initialization.
The range of permitted CANopen slave addresses is 1 to 127. Setting a higher address (> 128) does not lead to an error response, but results in a special mode (DS401). In this special mode, the device creates the node address by subtracting the value 128 from the address switch's value.

## I/O configuration

The CI582-CN CANopen bus configuration is handled by CANopen master with the exception of the slave node ID (via rotary switches) and the transmision rate (automatic detection).
The digital I/O channels and the fast counter are configured via software.

## Parameterization

## Parameters of the module

| Name | Value | Internal value | Internal value, type | Default |
| :---: | :---: | :---: | :---: | :---: |
| Module ID ${ }^{1}$ ) | Internal | 0x1C89 | WORD | 0x1C89 |
| Parameter length | Internal | 38 | BYTE | 38 |
| Error LED / failsafe function table error LED / failsafe function Table 227 "Err or LED / Failsafe function" on page 1005) | On | 0 | BYTE | 0 |
|  | Off by E4 | 1 |  |  |
|  | Off by E3 | 2 |  |  |
|  | On + failsafe | 16 |  |  |
|  | Off by E4 + failsafe | 17 |  |  |
|  | Off by E3 + failsafe | 18 |  |  |
| Reserved | 0 | 0 | ARRAY of 24 BYTES |  |
| Check supply | On | 0 | BYTE |  |
|  | Off | 1 |  | 1 |


| Name | Value | Internal value | Internal value, <br> type | Default |
| :--- | :--- | :--- | :--- | :--- |
| Fast counter | 0 | 0 | BYTE | 0 |
|  | $:$ | $:$ |  |  |
|  | $\left.10^{2}\right)$ | 10 |  |  |

${ }^{1}$ ) With a faulty ID, the module reports a "parameter error" and does not perform cyclic process data transmission.
${ }^{2}$ ) For a description of the counter operating modes, please refer to the 'Fast Counter' section ${ }^{2}$ Chapter 5.2.4.3.2.9 "Fast counter" on page 606.

Table 227: Error LED / Failsafe function

| Setting | Description |
| :--- | :--- |
| On | Error LED (S-ERR) lights up at errors of all <br> error classes, failsafe mode off |
| Off by E4 | Error LED (S-ERR) lights up at errors of error <br> classes E1, E2 and E3, failsafe mode off |
| Off by E3 | Error LED (S-ERR) lights up at errors of error <br> classes E1 and E2, failsafe mode off |
| On + Failsafe | Error LED (S-ERR) lights up at errors of all <br> error classes, failsafe mode on *) |
| Off by E4 + Failsafe | Error LED (S-ERR) lights up at errors of error <br> classes E1, E2 and E3, failsafe mode on *) |
| Off by E3 + Failsafe | Error LED (S-ERR) lights up at errors of error <br> classes E1 and E2, failsafe mode on *) |
| *) The parameter Behavior DO at comm. error is only analyzed if the failsafe mode is ON. |  |

## Group parameters for the digital part

$\left.\begin{array}{|l|l|l|l|l|}\hline \text { Name } & \text { Value } & \text { Internal value } & \begin{array}{l}\text { Internal value, } \\ \text { type }\end{array} & \text { Default } \\ \hline \text { Input delay } & 0.1 \mathrm{~ms} & 0 & \text { BYTE } & 0.1 \mathrm{~ms} \\ & 1 \mathrm{~ms} \\ 8 \mathrm{~ms} \\ 32 \mathrm{~ms}\end{array} \quad \begin{array}{l}2 \\ 3\end{array}\right)$

| Name | Value | Internal value | Internal value, <br> type | Default |
| :--- | :--- | :--- | :--- | :--- |
| Substitute value <br> at output | $0 \ldots 65535$ | $0000 \mathrm{~h} \ldots$ FFFFh | WORD | 0 |
| Preventive <br> voltage feedback <br> monitoring for <br> DC0 ... DC7 ${ }^{2}$ ) | Off <br> On | 1 <br> 0 | BYTE | Off <br> $0 \times 00$ |
| Detect voltage <br> overflow at out- <br> puts ${ }^{3}$ ) | Off <br> On | 0 | BYTE | Off <br> $0 \times 00$ |

Remarks:

| $\left.{ }^{1}\right)$ | The parameter Behavior DO at comm. error is applied to DC and DO <br> channels and only analyzed if the failsafe mode is ON. |
| :--- | :--- |
| $\left.{ }^{2}\right)^{2}$ | The state "externally voltage detected" appears if the output of a channel <br> DC0 ... DC7 is to be switched on while an external voltage is connected. <br> In this case, start-up is disabled while the externally voltage is con- <br> nected. The monitoring of this state and the resulting diagnosis message <br> can be disabled by setting the parameters to "OFF". |
| $\left.{ }^{3}\right)^{2}$ | The error state "voltage overflow at outputs" appears if external voltage <br> at digital outputs DC0 ... DC7 and DO0 ... DO7 has exceeded the <br> process supply voltage UP3 Chapter 5.2.6.3.3.4 "Connections" <br> on page 996. The according diagnosis message "Voltage overflow on <br> outputs " can be disabled by setting the parameters to "OFF". This <br> parameter should only be disabled in exceptional cases as voltage over- <br> flow may produce reverse voltage. |

## Diagnosis

| Byte Number | Description | Possible Values |
| :--- | :--- | :--- |
| 1 | Diagnosis byte, slot number | $31=\mathrm{CI} 582-\mathrm{CN}$ (e. g. error at integrated 8 DI / <br> $8 \mathrm{DO})$ <br> $1=1$ st connected S500 I/O module <br> $\ldots$ |
| 2 | Diagnosis byte, module <br> number | According to the I/O bus specification passed <br> on by modules to the fieldbus master |
| 3 | Diagnosis byte, channel | According to the I/O bus specification passed <br> on by modules to the fieldbus master |


| Byte Number | Description | Possible Values |
| :--- | :--- | :--- |
| 4 | Diagnosis byte, error code | According to the I/O bus specification |
|  |  | Bit 7 and bit 6, coded error class <br> $0=\mathrm{E} 1$ <br>  |
|  |  | $1=\mathrm{E} 2$ |
|  | $2=\mathrm{E} 3$ |  |
|  |  | $3=\mathrm{E} 4$ |
|  | Bit 0 to Bit 5, coded error description |  |
| 5 | According to the I/O bus specification |  |
|  |  | Bit 7: 1 = coming error |
| Bit 6: 1 = leaving error |  |  |

In cases of short circuit or overload, the digital outputs are turned off. The module performs reactivation automatically. Thus, an acknowledgement of the errors is not necessary. The error message is stored via the LED.

| E1..E4 | d1 | d2 | d3 | d4 | $\begin{array}{\|l\|} \hline \begin{array}{l} \text { Identi- } \\ \text { fier } \end{array} \\ 000 \text {.. } 063 \end{array}$ | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err |  |  |
| $\begin{array}{\|l} \hline \text { Byte } 4 \\ \text { Bit } \\ 6 \text {... } 7 \end{array}$ | - | Byte 1 | Byte 2 | Byte 3 | $\begin{aligned} & \text { Byte } 4 \\ & \text { Bit } 0 \\ & \ldots 5 \end{aligned}$ |  |  |
| Class | Interface | Device | Module | Channel | Error <br> identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | $\left.{ }^{3}\right)$ | ${ }^{4}$ ) |  |  |  |
| Module errors |  |  |  |  |  |  |  |
| 3 | - | 31 | 31 | 31 | 19 | Checksum error in the I/O module | Replace I/O module |
| 3 | - | 31 | 31 | 31 | 3 | Timeout in the I/O module |  |
| 3 | - | 31 | 31 | 31 | 40 | Different hard-/firmware versions in the module |  |
| 3 | - | 31 | 31 | 31 | 43 | Internal error in the module |  |
| 3 | - | 31 | 31 | 31 | 36 | Internal data exchange failure |  |
| 3 | - | 31 | 31 | 31 | 9 | Overflow diagnosis buffer | Restart |
| 3 | - | 31 | 31 | 31 | 26 | Parameter error | Check Master |


| E1..E4 | d1 | d2 | d3 | d4 | Identifier $000 \text {.. } 063$ | AC500 display | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err | $\begin{array}{\|l} \text { PS501 } \\ \text { PLC } \\ \text { Browser } \end{array}$ |  |  |
| $\begin{array}{\|l} \hline \text { Byte } 4 \\ \text { Bit } \\ 6 \text {... } 7 \end{array}$ | - | Byte 1 | Byte 2 | Byte 3 | $\begin{aligned} & \hline \text { Byte } 4 \\ & \text { Bit } 0 \\ & \ldots 5 \end{aligned}$ | CANope n diagnosis block |  |  |
| Class | Interface | Device | Module | Channel | Error <br> identifier | Error message |  | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) | ${ }^{4}$ ) |  |  |  |  |
| 3 | - | 31 | 31 | 31 | 11 | Process voltage UP too low |  | Check process supply voltage |
| 3 | - | 31 | 31 | 31 | 45 | Process voltage UP gone |  | Check process supply voltage |
| 3 | - | 31/1... 10 | 31 | 31 | 17 | No communication with I/O module |  | Replace I/O module |
| 3 | - | 1... 10 | 31 | 31 | 32 | Wrong I/O module type on socket |  | Replace I/O module / check configuration |
| 4 | - | 1... 10 | 31 | 31 | 31 | At least one module does not support failsafe function |  | Check modules and parameterization |
| 4 | - | 31 | 31 | 31 | 45 | Process voltage UP3 too low |  | Check process voltage |
| 4 | - | 31 | 31 | 31 | 46 | Voltage feedback on activated digital outputs ${ }^{4}$ ) |  | Check terminals |
| 4 | - | 31/1... 10 | 31 | 31 | 34 | No response during initialization of the I/O module |  | Replace I/O module |
| 4 | - | 31 | 31 | 31 | 11 | Process voltage UP3 too low |  | Check process supply voltage |
| 4 | - | 31 | 31 | 31 | 45 | Process voltage UP3 gone |  | Check process supply voltage |


| E1..E4 | d1 | d2 | d3 | d4 | $\begin{array}{\|l\|} \hline \begin{array}{l} \text { Identi- } \\ \text { fier } \end{array} \\ 000 \text {.. } 063 \end{array}$ | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err |  |  |
| $\begin{array}{\|l} \hline \text { Byte } 4 \\ \text { Bit } \\ 6 \text {... } 7 \end{array}$ | - | Byte 1 | Byte 2 | Byte 3 | $\begin{array}{\|l} \hline \text { Byte } 4 \\ \text { Bit } 0 \\ \text {... } 5 \end{array}$ |  |  |
| Class | Interface | Device | Module | Channel | Error <br> identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) | ${ }^{4}$ ) |  |  |  |
| 4 | - | 31 | 31 | 31 | 10 | Voltage overflow on outputs (above UP3 level) ${ }^{5}$ ) | Check terminals/ check process supply voltage |
| Channel error digital |  |  |  |  |  |  |  |
| 4 | - | 31 | 2 | 8... 15 | 46 | Externally voltage detected at digital output DO0 ... DO7 ${ }^{6}$ ) | Check terminals |
| 4 | - | 31 | 4 | 0... 7 | 46 | Externally voltage detected at digital output DC0 ... DC7 ${ }^{6}$ ) | Check terminals |
| 4 | - | 31 | 4 | 0... 7 | 47 | Short circuit at digital output DC0 ... DC77) | Check terminals |
| 4 | - | 31 | 2 | 8... 15 | 47 | Short circuit at digital output DO0 ... DO77) | Check terminals |

Remarks:

| $\left.{ }^{1}\right)$ | In AC500, the following interface identifier applies: <br> "-" = Diagnosis via bus-specific function blocks; $0 \ldots 4$ or 10 = position of the <br> communication module; 14 = I/O bus; 31 = module itself <br> The identifier is not contained in the CI542-DP diagnosis block. |
| :--- | :--- |
| $\left.{ }^{2}\right)$ | With "Device" the following allocation applies: $31=$ module itself, $1 . .10=$ <br> expansion module |
| $\left.{ }^{3}\right)$ | With "Module" the following allocation applies depending on the master: <br> Module error: $31=$ module itself <br> Channel error: module type ( $=\mathrm{AI}, 2=\mathrm{DO}, 3=\mathrm{AO})$ |


| ${ }^{4}$ ) | This message appears if external voltages at one or more terminals <br> DC0 ... DC7 or DO0 ... DO7 cause other digital outputs to be supplied by that <br> voltage \& Chapter 5.2.6.3.3.4 "Connections" on page 996. All outputs of the <br> digital output groups will be turned off for 5 seconds. The diagnosis message <br> appears for the whole output group. |
| :--- | :--- |
| ${ }^{5}$ ) | The voltage at digital outputs DCO ... DC7 and DOO ... DO7 has exceeded <br> the process supply voltage UP3 \& Chapter 5.2.6.3.3.4 "Connections" <br> on page 996. A diagnosis message appears for the whole module. |
| ${ }^{6}$ ) | This message appears if the output of a channel DC0 ... DC7 or DOO ... DO7 <br> should be switched on while an external voltage is connected. In this case <br> the start-up is disabled while the external voltage is connected. Otherwise, this <br> could produce reverse voltage flowing from this output to other digital outputs. <br> This diagnosis message appears for each channel. |
| ${ }^{7}$ ) | Short circuit: After a short circuit has been detected, the output is deactivated <br> for 100ms. Subsequently, a new start-up will be executed. This diagnosis mes- <br> sage appears for each channel. |

## State LEDs

The LEDs are located at the front of the module. There are 2 different groups:

- The 5 system LEDs (PWR, CN-RUN, CN-ERR, S-ERR and I/O bus) show the operation states of the module and display possible errors.
- The 29 process LEDs (UP, UP3, inputs, outputs, CH-ERR1 to CH-ERR3) show the process supply voltage and the states of the inputs and outputs and display possible errors.

States of the 5 system LEDs

| LED | Color | OFF | ON | Flashing |
| :--- | :--- | :--- | :--- | :--- |
| PWR/RUN | Green | Process supply <br> voltage missing | Internal supply <br> voltage OK, <br> module ready for <br> communication <br> with I/O controller | Start-up / pre- <br> paring communi- <br> cation |
|  |  |  | ---1 | --- |
|  | Yellow | --- | Device config- <br> ured, CANopen <br> bus in OPERRA- <br> TIONAL state <br> and cyclic data <br> exchange run- <br> ning | Flashing: <br> CANopen bus in <br> PRE-OPERAA- <br> TIONAL state <br> and slave is <br> being configured <br> Single flash: <br> CANopen bus in <br> STOPPED state. |
|  | Green | --- | Flickering: Auto- <br> detect is active |  |


| LED | Color | OFF | ON | Flashing |
| :--- | :--- | :--- | :--- | :--- |
| CN-ERR | Red | No system error | CANopen Bus is <br> OFF | Flashing: Config- <br> uration error <br> Single flash: error <br> counter overflow <br> due to too many <br> error frames <br> Double flash: A <br> node-guard or a <br> heartbeat event <br> occurred <br> Flickering: Auto- <br> detect is active |
| S-ERR | Red | No error | Internal error | -- |
| I/O bus | Green | No decentralized <br> I/O modules con- <br> nected or com- <br> munication error | Decentralized I/O <br> modules con- <br> nected and <br> operational | -- |

States of the 29 process LEDs

| LED | Color | OFF | ON | Flashing |
| :--- | :--- | :--- | :--- | :--- |
| DC0 ... DC7 | Yellow | Input/output is OFF | Input/output is ON | -- |
| D18 ...DI15 | Yellow | Input is OFF | Input is ON (the input <br> voltage is even dis- <br> played if the supply <br> voltage is OFF) | -- |
| DO8 ... DO15 | Yellow | Output is OFF | Output is ON | -- |
| UP | Green | Process supply <br> voltage missing | Process supply <br> voltage OK and initi- <br> alization finished | -- |
| UP3 | Green | Process supply <br> voltage missing | Process supply <br> voltage OK | -- |
| CH-ERR1 to <br> CH-ERR3 | Red | No error or process <br> supply voltage <br> missing | Internal error | Error on one channel <br> of the corresponding <br> group |

## Technical data

## Technical data of the module

The system data of AC500 and S500 are applicable to the standard version ${ }^{\mu}$ Chapter 5.1.2 "System data AC500" on page 166
The system data of AC500-XC are applicable to the XC version ${ }^{\Perp}$ Chapter 5.1.3 "System data AC500-XC" on page 169.

Only additional details are therefore documented below.
The technical data are also applicable to the XC version.


## Multiple overloads

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an internal smart high-side switch.

| Parameter | Value |
| :---: | :---: |
| Interface | CAN |
| Protocol | CANopen |
| Power supply | From the process supply voltage UP |
| Supply of the electronic circuitry of the I/O modules attached | Through the I/O bus interface (I/O bus) |
| Rotary switches | For setting the CANopen Node ID for configuration purposes (00h to FFh) |
| LED displays | For system displays, signal states, errors and power supply |
| External supply voltage | Via terminals ZP, UP and UP3 (process supply voltage 24 V DC) |
| Transmission rates | $10 / 20 / 50 / 125 / 250 / 500 / 800 \mathrm{kbit} / \mathrm{s} 1$ Mbit/s Auto transmission rate detection is supported |
| Bus connection | Depending on used terminal unit TU510: 9-pin D-sub connector TU518: 10-pin terminal block |
| Processor | Hilscher NETX 100 |
| Expandability | CI58x can only be used on onboard CAN interface and without any I/O expansion module ⓨ Table 215 "CANopen" on page 952. |
| State display | Module state: PWR/RUN, CN-RUN, CN-ERR, E-ERR, I/O bus |
| Adjusting elements | 2 rotary switches for generation of the node address |
| Ambient temperature | System data AC500 ${ }^{\wedge}$ Chapter 5.1.2 "System data AC500" on page 166 <br> System data AC500 XC $\Leftrightarrow$ Chapter 5.1.3 "System data AC500-XC" on page 169 |
| Current consumption | UP: 0.2 A UP3: 0.06 A + 0.5 A max. per output |
| Weight (without terminal unit) | Ca. 125 g |
| Process supply voltages UP/UP3 |  |
| Rated value | 24 V DC (for inputs and outputs) |
| Max. load for the terminals | 10 A |
| Protection against reversed voltage | Yes |
| Rated protection fuse on UP/UP3 | 10 A fast |
| Galvanic isolation | CANopen interface against the rest of the module |
| Inrush current from UP (at power up) | On request |
| Current consumption via UP (normal operation) | 0.2 A |
| Current consumption via UP3 | 0.06 A + 0.5 A max. per output |
| Connections | Terminals 2.8 and 3.8 for +24 V (UP) <br> Terminal 4.8 for +24 V (UP3) <br> Terminals 2.9, 3.9 and 4.9 for $0 \mathrm{~V}(\mathrm{ZP})$ |
| Max. power dissipation within the module | 6 W |


| Parameter | Value |
| :--- | :--- |
| Reference potential for all digital inputs and <br> outputs | Negative pole of the supply voltage, signal <br> name ZP |
| Setting of the CANopen Node ID identifier | With 2 rotary switches at the front side of the <br> module |
| Mounting position | Horizontal <br> Or vertical with derating (output load reduced to <br> 50 \% at $+40^{\circ} \mathrm{C}$ per group) |
| Cooling | The natural convection cooling must not be hin- <br> dered by cable ducts or other parts in the con- <br> trol cabinet. |
| Effect of incorrect input terminal connection | Wrong or no signal detected, no damage up to <br> 35 V |
| Required terminal unit | TU509, TU510, TU517 or TU518 <br> Y Chapter 5.2.7.4 "TU517 and TU518 for com- <br> munication interface modules" on page 1278 |

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and continuous overvoltage up to 30 V DC.

## Technical data of the digital inputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 8 |
| Distribution of the channels into groups | 1 group of 8 channels |
| Terminals of the channels DI0 ... DI7 | Terminals $3.0 \ldots 3.7$ |
| Reference potential for all inputs | Terminals $2.9 \ldots 4.9$ (negative pole of the <br> supply voltage, signal name ZP) |
| Indication of the input signals | 1 yellow LED per channel, the LED is ON <br> when the input signal is high (signal 1) |
| Input type (according EN 61131-2) | Type 1 |
| Input delay (0->1 or 1->0) | Typ. 0.1 ms, configurable from <br> $0.1 \mathrm{~ms} . .32 \mathrm{~ms}$ |
| Input signal voltage | 24 V DC |
| Signal 0 | $-3 \mathrm{~V} \ldots+5 \mathrm{~V}$ |
| Undefined signal | $>+5 \mathrm{~V} \ldots<+15 \mathrm{~V}$ |
| Signal 1 | $+15 \mathrm{~V} \ldots+30 \mathrm{~V}$ |
| Ripple with signal 0 | Within $-3 \mathrm{~V} \ldots+5 \mathrm{~V}$ |
| Ripple with signal 1 | Within $+15 \mathrm{~V} \ldots+30 \mathrm{~V}$ |
| Input current per channel | Typ. 5 mA |
| Input voltage +24 V | $>1 \mathrm{~mA}$ |
| Input voltage +5 V | $>2 \mathrm{~mA}$ |
| Input voltage +15 V | $<8 \mathrm{~mA}$ |
| Input voltage +30 V |  |


| Parameter | Value |
| :--- | :--- |
| Max. cable length |  |
| Shielded | 1000 m |
| Unshielded | 600 m |

## Technical data of the digital outputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 8 |
| Distribution of the channels into groups | 1 group of 8 channels |
| Terminals of the channels DO0 to DO7 | Terminals 4.0 to 4.7 |
| Reference potential for all outputs | Terminals 2.9 ... 4.9 (negative pole of the supply voltage, signal name ZP) |
| Common power supply voltage | For all outputs terminal 4.8 (positive pole of the supply voltage, signal name UP3) |
| Output voltage for signal 1 | UP3 (-0.8 V) |
| Output delay (0->1 or 1->0) | On request |
| Output current |  |
| Rated value per channel | 500 mA at UP3 $=24 \mathrm{~V}$ |
| Max. value (all channels together) | 4 A |
| Leakage current with signal 0 | $<0.5 \mathrm{~mA}$ |
| Fuse for UP3 | 10 A fast |
| Demagnetization with inductive DC load | Via internal varistors (see figure below this table) |
| Output switching frequency |  |
| With resistive load | On request |
| With inductive loads | Max. 0.5 Hz |
| With lamp loads | 11 Hz max. at 5 W max. |
| Short-circuit-proof / overload-proof | Yes |
| Overload message ( l 0 0.7 A) | Yes, after ca. 100 ms |
| Output current limitation | Yes, automatic reactivation after short circuit/overload |
| Resistance to feedback against 24 V signals | Yes (software-controlled supervision) |
| Max. cable length |  |
| Shielded | 1000 m |
| Unshielded | 600 m |



Fig. 190: Circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off

1 Digital output
2 Varistors for demagnetization when inductive loads are turned off

## Technical data of the configurable digital inputs/outputs

Each of the configurable I/O channels is defined as input or output by the user program. This is done by interrogating or allocating the corresponding channel.

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 8 inputs/outputs (with transistors) |
| Distribution of the channels into groups | 1 group for 8 channels |
| If the channels are used as inputs |  |
| Channels DC0...DC07 | Terminals 2.0...2.7 |
| If the channels are used as outputs | Terminals 2.0...2.7 |
| Channels DC0...DC07 | 1 yellow LED per channel, the LED is ON <br> when the input/output signal is high (signal 1) |
| Indication of the input/output signals | From the CANopen network |
| Galvanic isolation |  |

Technical data of the digital inputs/outputs if used as inputs

Please refer to the Technical Data of the Digital Inputs $\Leftrightarrow$ Chapter 5.2.6.3.3.11 "Technical data" on page 1011. Deviation:
Terminals of the channels DC0 to DC7: Terminals 2.0 to 2.7
Due to the direct connection to the output, the demagnetizing varistor is also effective at the input. This is why the difference between UPx and the input signal must not exceed the clamp voltage of the varistor. The varistor limits the clamp voltage to approx. 36 V . Consequently, the input voltage must range from -12 V to +30 V when $\mathrm{UPx}=24 \mathrm{~V}$ and from -6 V to +30 V when $U P x=30 \mathrm{~V}$.

Please refer to the Technical Data of the Digital Outputs ${ }^{\circledR}>$ Chapter 5.2.6.3.3.11 "Technical data" on page 1011. Deviation:
Terminals of the channels DC0 to DC7: Terminals 2.0 to 2.7
The following drawing shows the circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off.


Fig. 191: Digital input/output (circuit diagram)

| 1 | Digital input/output |
| :--- | :--- |
| 2 | For demagnetization when inductive loads are turned off |

## Technical data of the fast counter

| Parameter | Value |
| :--- | :--- |
| Used inputs | Terminal 3.0 (DI8), 3.1 (DI9) |
| Used outputs | Terminal 4.0 (DO8) |
| Counting frequency | Depending on operation mode: <br> Mode 1-6: max. 200 kHz <br> Mode 7: max. 50 kHz <br> Mode 9: max. 35 kHz <br> Mode 10: max. 20 kHz |
| Detailed description | Fast Counter « Chapter 5.2.4.3.2.9 "Fast <br> counter" on page 606 |
| Operating modes | Operating modes ङ Chapter 5.2.4.3.2.9 "Fast <br> counter" on page 606 |

## Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm

The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 228 200 R0001 | CI582-CN, CANopen <br> communication interface module with <br> 8 DI, 8 DO and 8 DC | Active |
| 1SAP 428 200 R0001 | CI582-CN-XC, CANopen <br> communication interface module with <br> 8 DI, 8 DO and 8 DC, XC version | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.6.4 EtherCAT

### 5.2.6.4.1 CI511-ETHCAT

## Features

- 4 analog inputs (resolution 12 bits including sign)
- 2 analog outputs (resolution 12 bits including sign)
- 8 digital inputs 24 V DC
- 8 digital outputs 24 V DC, 0.5 A max.
- Cam switch functionality (see also Extended Cam Switch Library)
- Extended Cam switch functionality *) (see also Extended Cam Switch Library)
- Module-wise galvanically isolated - Expandability with up to 10 S500 I/O Modules *)
*) Applicable for device index C0 and above.


1 I/O bus
2 Allocation between terminal number and signal name
36 yellow LEDs to display the signal states of the analog inputs/outputs (AIO ... AI3, AO0 ... AO1)
48 yellow LEDs to display the signal states of the digital inputs (DIO ... DI7)
58 yellow LEDs to display the signal states of the digital outputs (DO0 ... DO7)
62 green LEDs to display the supply voltage UP and UP3
73 red LEDs to display errors (CH-ERR1, CH-ERR2, CH-ERR3)
85 system LEDs: PWR/RUN, NET, DC, S-ERR, I/O-Bus
92 rotary switches (reserved for future extensions)
10 Label

11 Ethernet interfaces (ETH1, ETH2) on the terminal unit
12 Terminal unit
13 DIN rail

## Intended purpose

The EtherCAT communication interface module CI511-ETHCAT is used as decentralized I/O module in EtherCAT networks. The network connection is performed via 2 RJ45 connectors which are integrated in the terminal unit. The communication interface module contains 22 I/O channels with the following properties:

- 4 analog inputs (1.0 ... 1.3)
- 2 analog outputs (1.5 ... 1.6)
- 8 digital inputs 24 V DC in 1 group (2.0 ... 2.7)
- 8 digital outputs 24 V DC in 1 group (3.0 ... 3.7)
- Cam switch functionality

The inputs/outputs are galvanically isolated from the Ethernet network. There is no potential separation between the channels. The configuration of the analog inputs/outputs is performed by software.

## Functionality

| Parameter | Value |
| :--- | :--- |
| Interface | Ethernet |
| Protocol | EtherCAT |
| Power supply | From the process supply voltage UP |
| Supply of the electronic circuitry of the I/O <br> expansion modules attached | Through the I/O bus interface (I/O bus) |
| Rotary switches | Not used; reserved for future extensions |
| Analog inputs | 4 (configurable via software) |
| Analog outputs | 2 (configurable via software) |
| Digital inputs | 8 (24 V DC; delay time configurable via soft- <br> ware) |
| Digital outputs | 8 (24 V DC, 0.5 A max.) |
| LED displays | For system displays, signal states, errors and <br> power supply |
| External supply voltage | Via terminals ZP, UP and UP3 (process supply <br> voltage 24 V DC) |
| Effect of incorrect input terminal connection | Wrong or no signal detected, no damage up to <br> 35 V |
| Required terminal unit | TU507 or TU508 « Chapter 5.2.7.2 "TU507- <br> ETH and TU508-ETH for Ethernet communica- <br> tion interface modules" on page 1270 |

## Connections

## General

The Ethernet communication interface module CI511-ETHCAT is plugged on the I/O terminal unit TU507-ETH or TU508-ETH. Properly seat the module and press until it locks in place. The terminal unit is mounted on a DIN rail or with 2 screws plus the additional accessory for wall mounting (TA526).

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

The connection of the I/O channels is carried out using the 30 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

The terminals 1.8 and 2.8 as well as $1.9,2.9$ and 3.9 are electrically interconnected within the terminal unit and have always the same assignment, independent of the inserted module:
Terminals 1.8 and 2.8: Process supply voltage UP $=+24 \mathrm{~V}$ DC
Terminal 3.8: Process supply voltage UP3 $=+24 \mathrm{~V}$ DC
Terminals 1.9, 2.9 and 3.9: Process supply voltage $\mathrm{ZP}=0 \mathrm{~V}$

With a separate UP3 power supply, the digital outputs can be switched off externally. This way, an emergency-off functionality can be realized.

The assignment of the other terminals:

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| $1.0 \ldots 1.3$ | AIO ... AI3 | Positive pole of the 4 analog <br> inputs |
| 1.4 | AI- | Negative pole of the analog <br> inputs |
| $1.5 \ldots 1.6$ | AO0 ... AO1 | Positive pole of the 2 analog <br> outputs |
| 1.7 | AO- | Negative pole of the analog <br> outputs |
| $2.0 \ldots 2.7$ | DI0 $\ldots$ DI7 | 8 digital inputs |
| $3.0 \ldots 3.7$ | DO0 ... DO7 | 8 digital outputs |

## WARNING!

## Removal/Insertion under power

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.
Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.
Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.
The devices must not be opened when in operation. The same applies to the network interfaces.

## CAUTION!

There is no galvanic isolation between the analog circuitry and ZP/UP. Therefore, the analog sensors must be galvanically isolated in order to avoid loops via the ground potential or the supply voltage.

## CAUTION!

Because of their common reference potential, analog current inputs cannot be circuited in series, neither within the module nor with channels of other modules.

For the open-circuit detection (wire break), each channel is pulled up to "plus" by a high-resistance resistor. If nothing is connected, the maximum voltage will be read in then.

Analog signals are always laid in shielded cables. The cable shields are grounded at both ends of the cables. In order to avoid unacceptable potential differences between different parts of the installation, low resistance equipotential bonding conductors must be laid.

For simple applications (low disturbances, no high requirement on precision), the shielding can also be omitted.


Fig. 192: Connection of the communication interface module CI511-ETHCAT
14 analog inputs, configurable for $0 \ldots 10 \mathrm{~V},-10 \ldots+10 \mathrm{~V}, 0 / 4 \ldots 20 \mathrm{~mA}, \mathrm{Pt100/Pt} 1000$, Ni1000 and digital signals
22 analog outputs, configurable for -10 ... +10 V, 0/4 ... 20 mA
38 digital inputs 24 V DC
48 digital outputs 24 V DC, 0.5 A max.

In case of voltage feedback, 2 cases are distinguished:

1. The outputs are already active

The output group will be switched off. A diagnosis message will appear. After 5 seconds, the module tries automatic reactivation.
2. The outputs are not active

Only the output with voltage feedback will not be set to active. A diagnosis message will appear.

## NOTICE!

## Risk of faulty measurements!

The negative pole/ground potential at the sensors must not have too large a potential difference with respect to ZP (max. $\pm 1 \mathrm{~V}$ within the full signal range).
Make sure that the potential difference never exceeds $\pm 1 \mathrm{~V}$.

## CAUTION!

The process supply voltage must be included within the grounding concept of the plant (e. g. grounding of the negative pole).

The module provide several diagnosis functions $\Leftrightarrow$ Chapter 5.2.6.4.1.9 "Diagnosis" on page 1039.
The measuring ranges are described in the section Measuring Ranges ${ }^{\mu} \Rightarrow$ Chapter 5.2.6.4.1.8 "Parameterization" on page 1033 \& Chapter 5.2.6.4.1.11 "Measuring ranges" on page 1042.

The function of the LEDs is described in the section State LEDs $\Leftrightarrow>$ Chapter 5.2.6.4.1.9 "Diagnosis" on page 1039.

## Connection of resistance thermometers in 2-wire configuration

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the module CI511-ETHCAT provides a constant current source which is multiplexed over the max. 4 analog input channels.


Fig. 193: Connection of resistance thermometers in 2-wire configuration
1 Pt100 (2-wire), Pt1000 (2-wire), Ni1000 (2-wire); 1 analog sensor requires 1 channel

| Pt100 | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 2-wire configuration, 1 <br> channel used |
| :--- | :--- | :--- |
| $\operatorname{Pt1000}$ | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 2-wire configuration, 1 <br> channel used |
| Ni1000 | $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ | 2-wire configuration, 1 <br> channel used |

The measuring ranges are described in the section Measuring Ranges ${ }^{\mu} \Rightarrow$ Chapter 5 .2.6.4.1.8 "Parameterization" on page 1033 ${ }^{〔}$ Chapter 5.2.6.4.1.11 "Measuring ranges" on page 1042.

The module CI511-ETHCAT performs a linearization of the resistance characteristic.
In order to avoid error messages from unused analog input channels, it is useful to configure them as "unused".

## Connection of resistance thermometers in 3-wire configuration

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the module CI511-ETHCAT provides a constant current source which is multiplexed over the max. 4 analog input channels.


Fig. 194: Connection of resistance thermometers in 3-wire configuration
1 Pt100 (3-wire), Pt1000 (3-wire), Ni1000 (3-wire); 1 analog sensor requires 2 channels
2 Twisted pair within the cable
3 Return line: The return line is only needed once if measuring points are adjacent to each other. This saves wiring costs.

With 3-wire configuration, two adjacent analog channels belong together (e. g. the channels 0 and 1). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0 ), the next higher address must be the odd address (channel 1).

The constant current of one channel flows through the resistance thermometer. The constant current of the other channel flows through one of the cores. The module calculates the measured value from the two voltage drops and stores it under the input with the higher channel number (e. g. I1).
In order to keep measuring errors as small as possible, it is necessary, to have all the involved conductors in the same cable. All the conductors must have the same cross section.

| Pt100 | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 3-wire configuration, 2 chan- <br> nels used |
| :--- | :--- | :--- |
| Pt1000 | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 3-wire configuration, 2 chan- <br> nels used |
| Ni1000 | $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ | 3-wire configuration, 2 chan- <br> nels used |

The measuring ranges are described in the section Measuring Ranges ${ }^{\mu} \Rightarrow$ Chapter 5.2.6.4.1.8 "Parameterization" on page 1033 出 Chapter 5.2.6.4.1.11 "Measuring ranges" on page 1042.
The module CI511-ETHCAT performs a linearization of the resistance characteristic.
In order to avoid error messages from unused analog input channels, it is useful to configure them as "unused".

## Connection of active-type analog sensors (Voltage) with galvanically isolated power supply



Fig. 195: Connection of active-type analog sensors (voltage) with galvanically isolated power supply

11 analog sensor requires 1 channel
2 By connecting to AI-, the galvanically isolated voltage source of the sensor is referred to ZP
3 Galvanically isolated power supply for the analog sensor

| Voltage | $0 \ldots 10 \mathrm{~V}$ | 1 channel used |
| :--- | :--- | :--- |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | 1 channel used |

The measuring ranges are described in the section Measuring Ranges ${ }_{\mu}^{\mu}$ Chapter $5 \cdot 2 \cdot 6 \cdot 4.1 .8$ "Parameterization" on page 1033 \& $\stackrel{y}{ }$ Chapter 5.2.6.4.1.11 "Measuring ranges" on page 1042.

In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as "unused".

## Connection of active-type analog sensors (Current) with galvanically isolated power supply



Fig. 196: Connection of active-type analog sensors (current) with galvanically isolated power supply

11 analog sensor requires 1 channel
2 Galvanically isolated power supply for the analog sensor

| Current | $0 \ldots 20 \mathrm{~mA}$ | 1 channel used |
| :--- | :--- | :--- |
| Current | $4 \ldots 20 \mathrm{~mA}$ | 1 channel used |

The measuring ranges are described in the section Measuring Ranges $\Leftrightarrow$ Chapter 5 2.2.6.4.1.8 "Parameterization" on page 1033 \& Chapter 5.2.6.4.1.11 "Measuring ranges" on page 1042. Unused input channels can be left open-circuited, because they are of low resistance.

## Connection of active-type analog sensors (Voltage) with no galvanically isolated power supply



Fig. 197: Connection of active-type sensors (voltage) with no galvanically isolated power supply
11 analog sensor requires 1 channel
2 Power supply not galvanically isolated
3 The connection between the negative pole of the sensor and ZP has to be performed
4 Long cable

## NOTICE!

## Risk of faulty measurements!

The negative pole/ground potential at the sensors must not have too large a potential difference with respect to ZP (max. $\pm 1 \mathrm{~V}$ within the full signal range).
Make sure that the potential difference never exceeds $\pm 1 \mathrm{~V}$.

| Voltage | $0 \ldots 10 \mathrm{~V}$ | 1 channel used |
| :--- | :--- | :--- |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ ) | 1 channel used |

*) if the sensor can provide this signal range
The measuring ranges are described in the section Measuring Ranges $\Longleftrightarrow$ Chapter $5 \cdot 2 \cdot 6 \cdot 4.1 .8$ "Parameterization" on page 1033 出 Chapter 5.2.6.4.1.11 "Measuring ranges" on page 1042.
In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as "unused".

## Connection of passive-type analog sensors (Current)



Fig. 198: Connection of passive-type analog sensors (current)
11 analog sensor requires 1 channel

| Current | $4 \ldots 20 \mathrm{~mA}$ | 1 channel used |
| :--- | :--- | :--- |

The measuring ranges are described in the section Measuring Ranges ${ }^{\mu}$ Chapter 5 .2.6.4.1.8 "Parameterization" on page $1033 \Leftrightarrow$ Chapter 5.2.6.4.1.11 "Measuring ranges" on page 1042.

## CAUTION!

If, during initialization, an analog current sensor supplies more than 25 mA for more than 1 second into an analog input, this input is switched off by the module (input protection). In such cases, it is recommended, to protect the analog input by a 10 -volt Zener diode (in parallel to I+ and ZP). But, in general, it is a better solution to prefer sensors with fast initialization or without current peaks higher than 25 mA .

Unused input channels can be left open-circuited, because they are of low resistance.

## Connection of active-type analog sensors (Voltage) to differential inputs

Differential inputs are very useful, if analog sensors are used which are remotely non-isolated (e.g. the minus terminal is remotely grounded).

The evaluation using differential inputs helps to considerably increase the measuring accuracy and to avoid ground loops.
With differential input configurations, two adjacent analog channels belong together (e.g. the channels 0 and 1 ). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0 ), the next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1).

The analog value is calculated by subtraction of the input value with the higher address from the input value of the lower address.

The converted analog value is available at the odd channel (higher address).
Important: The ground potential at the sensors must not have a too big potential difference with respect to ZP (max. $\pm 1 \mathrm{~V}$ within the full signal range). Otherwise problems can occur concerning the common-mode input voltages of the involved analog inputs


Fig. 199: Connection of active-type analog sensors (voltage) to differential inputs
11 analog sensor requires 2 channels
2 Galvanically isolated power supply for the analog sensor
3 Grounding at the sensor
$40 \vee \ldots 10 \vee /-10 \vee \ldots+10 \mathrm{~V}$ connected to differential inputs

| Voltage | $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ | with differential inputs, 2 chan- <br> nels used |
| :--- | :--- | :--- |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | with differential inputs, 2 chan- <br> nels used |

The measuring ranges are described in the section Measuring Ranges $\Rightarrow$ Chapter 5.2.6.4.1.8 "Parameterization" on page 1033 $\Rightarrow$ Chapter 5.2.6.4.1.11 "Measuring ranges" on page 1042.
In order to avoid error messages or long processing times, it is useful to configure unused analog input channels as "unused".

## Use of analog inputs as digital inputs

Several (or all) analog inputs can be configured as digital input. The inputs are not galvanically isolated against the other analog channels.


Fig. 200: Use of analog inputs as digital inputs
11 digital signal requires 1 channel

| Digital input | 24 V | 1 channel used |
| :--- | :--- | :--- |

The measuring ranges are described in the section Measuring Ranges $\Rightarrow$ Chapter 5.2.6.4.1.8 "Parameterization" on page 1033 « Chapter 5.2.6.4.1.11 "Measuring ranges" on page 1042.

## Connection of analog output loads (Voltage, current)



Fig. 201: Connection of analog output loads (voltage, current)
11 analog load requires 1 channel

| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | Load $\pm 10 \mathrm{~mA}$ max. | 1 channel used |
| :--- | :--- | :--- | :--- |
| Current | $0 \ldots 20 \mathrm{~mA}$ | Load $0 \Omega \ldots 500 \Omega$ | 1 channel used |
| Current | $4 \ldots 20 \mathrm{~mA}$ | Load $0 \Omega \ldots 500 \Omega$ | 1 channel used |

 "Parameterization" on page 1033 y Chapter 5.2.6.4.1.11 "Measuring ranges" on page 1042.
Unused analog outputs can be left open-circuited.

## Assignment of the Ethernet ports

The terminal unit for the communication interface module provides two Ethernet interfaces with the following pin assignment. The pin assignment is used for the EtherCAT master (communication module CM5xy-ETHCAT) as well.

| Pin assignment | Interface | Pin | Signal | Description |
| :---: | :---: | :---: | :---: | :---: |
|  |  | 1 | TxD+ | Transmit data + |
|  |  | 2 | TxD- | Transmit data - |
|  |  | 3 | RxD+ | Receive data + |
|  |  | 4 | NC | Not connected |
|  |  | 5 | NC | Not connected |
|  |  | 6 | RxD- | Receive data - |
|  |  | 7 | NC | Not connected |
|  |  | 8 | NC | Not connected |
|  |  | Shield | Cable shield | Functional earth |

In corrosive environment, please protect unused connectors using the TA535 accessory.
Not supplied with this device.
*3 Further information about wiring and cable types

The EtherCAT network differentiates between input-connectors (IN) and outputconnectors (OUT):
At the EtherCAT slaves (communication interface modules), the ETH1-connector is IN and the ETH2-connector is OUT.
At the EtherCAT master (communication module), the ETHCAT1 connector has to be used. The ETHCAT2 connector is reserved for future extensions.

## Internal data exchange

| Parameter | Value |
| :--- | :--- |
| Digital inputs (bytes) | 1 |
| Digital outputs (bytes) | 1 |
| Analog inputs (words) | 4 |
| Analog outputs (words) | 2 |

## Addressing

The Ethernet bus module CI511-ETHCAT does not consider the position of the rotary switches at the front side of the module. The function of the rotary switches is reserved for future expansions.

## I/O configuration

In order to be able to use the CI51X-ETHCAT with device index C0 or above properly, please download the corresponding device description (.xml-)files from http://www.abb.com/plc and install them to the device repository of your Automation Builder. This will allow you to use up to 10 Expandable S500 I/O modules as well as the Extended Cam Switch Library with your CI51X-ETHCAT device.

The CI511-ETHCAT does not store configuration data itself.
The analog I/O channels are configured via software.

## Parameterization

## Module parameter

| Name | Value | Internal value | Internal value, <br> type | Default |
| :--- | :--- | :--- | :--- | :--- |
| Module ID | Internal | 48155 | WORD | 48155 |
| Parameter length | Internal | 28 | BYTE | 28 |
| Error LED / Fail- <br> safe function ${ }^{1}$ ) | On <br> Off by E4 <br> Off by E3 On + <br> failsafe Off by E4 <br> + failsafe Off by <br> E3 + failsafe | 16 <br> 17 | 17 <br> 19 | 0 |
| Check Supply | Off <br> On | 0 | 1 | BYTE |

Table 228: Error LED / Failsafe function ${ }^{1}$ )

| Setting | Description |
| :--- | :--- |
| On | Error LED lights up at errors of all error classes, Failsafemode off |
| Off by E4 | Error LED lights up at errors of error classes E1, E2 and E3, Failsa- <br> femode off |
| Off by E3 | Error LED lights up at errors of error classes E1 and E2 auf, Failsa- <br> femode off |
| On + failsafe | Error LED lights up at errors of all error classes, Failsafemode on *) |
| Off by E4 + failsafe | Error LED lights up at errors of error classes E1, E2 and E3, Failsa- <br> femode on *) |
| Off by E3 + failsafe | Error LED lights up at errors of error classes E1 and E2, Failsafe- <br> mode on *) |

[^18]
## Group parameters of the cam switch

| Name | Value | Internal value | Internal value, type | Default |
| :---: | :---: | :---: | :---: | :---: |
| numOfUsedCams ${ }^{1}$ ) | $\begin{array}{\|l\|} \hline 0 \ldots 32 \\ 128 \ldots 160 \end{array}$ | $\begin{aligned} & \hline 0 \ldots 32 \\ & 218 \ldots 160 \end{aligned}$ | WORD | 0 |
| resolution ${ }^{2}$ ) | $\begin{aligned} & 0 \ldots 2 \\ & -1 \end{aligned}$ | $\begin{aligned} & 0 \ldots 2 \\ & -1 \end{aligned}$ | DWORD | 36000 |
| zeroShift ${ }^{3}$ ) | $\begin{aligned} & 0 \ldots 2 \\ & -1 \end{aligned}$ | $\begin{aligned} & 0 \ldots 2 \\ & -1 \end{aligned}$ | DWORD | 0 |
| EncoderBitResolution ${ }^{4}$ ) | $8 \ldots 32$ | 8 ... 32 | WORD | 18 |
| Reserve | - | - | WORD | - |

${ }^{1}$ ) The parameter numOfUsedCams defines the interrupt cycle time (Therefore, it takes effect to the accuracy of the track) and the behavior of the module if the DC information is lost.

| Parameter setting <br> for numOfUsed- <br> Cams | Number of cams <br> used | Interrupt cycle time | Behavior if DC infor- <br> mation is lost |
| :--- | :--- | :--- | :--- |
| 0 | 0 | $50 \mu \mathrm{~s}$ | Module changes <br> to "safe-operational" <br> state; the outputs are <br> activated trough the <br> user program |
| $1 \ldots 8$ | $1 \ldots 8$ | $80 \mu \mathrm{~s}$ | $100 \mu \mathrm{~s}$ |
| $9 \ldots 16$ | $9 \ldots 16$ | $200 \mu \mathrm{~s}$ | Module keeps in <br> "operational" state; <br> the outputs are acti- <br> vated trough the user <br> program |
| $17 \ldots 32$ | $17 \ldots 32$ | $50 \mu \mathrm{~s}$ | Module keeps in <br> "operational" state; <br> the cam switch out- |
| 128 | 0 | $80 \mu \mathrm{~s}$ | puts are activated <br> according to an inter- <br> polated timing infor- <br> mation |
| $129 \ldots 136$ | $1 \ldots 8$ | $100 \mu \mathrm{~s}$ | $200 \mu \mathrm{~s}$ |
| $137 \ldots 144$ | $9 \ldots 16$ | $17 \ldots 32$ |  |
| $145 \ldots 170$ |  |  |  |

${ }^{2}$ ) The parameter resolution defines the angle resolution of the track. The value gives the number of increments related to $360^{\circ}$; e. g. the value 36,000 corresponds to an angle resolution of $0.01^{\circ}$.
${ }^{3}$ ) The parameter zeroShift defines the zero shift. With it the encoder can be adjusted to the mounting position. The value of zeroShift is set in encoder-increments. It is not assigned to the parameter resolution of the cam switch.
${ }^{4}$ ) The parameter EncoderBitResolution defines the resolution of the used encoder (in bits), e. g. with the default setting 18 bits the encoder has 196,608 divisions.

Channel parameters for the cam switch (max. 32x)

| Name | Value | Internal value | Internal value, <br> type | Default |
| :--- | :--- | :--- | :--- | :--- |
| camToTrack0 *) | Digital Output <br> $0 \ldots 7$, none | $0 \ldots 7$, FF | BYTE | FF |
| $:$ | $:$ | $:$ | $:$ | $:$ |
| camToTrack31 | Digital Output <br> $0 \ldots 7$, none | $0 \ldots 7$, FF | BYTE | FF |

*) The value of the parameter camToTrack\# defines which DO (digital output) is assigned to the track. camToTrack0 $=3$ for example means that track 0 is assigned to the digital output 3 . If the value FFh is set to a track, no digital output is assigned to it.

| Name | Value | Referred FB from extended Cam Switch Library ${ }^{2}$ ) | Internal value | Internal value, type | Default |
| :---: | :---: | :---: | :---: | :---: | :---: |
| camType[0] ${ }^{1}$ ) ... | Common <br> Pulsed <br> Timed <br> Comfort <br> Cam shift <br> Binary shift <br> Multiturn cam <br> Time timed <br> Reference <br> Multiturn timed | MCX_CamSwitchSimple_c <br> MCX_CamSwitchSimple_dc <br> MCX_PulseSwitch_dc <br> MCX_CamSwitchTimed_dc <br> MCX_CamSwitchCom- <br> fort_dc <br> MCX_CamShift_dc <br> MCX_BinaryShift_dc <br> MCX_CamSwitchMulti_dc <br> MCX_SwitchTimeTimed_dc <br> MCX_BinaryReference_dc <br> MCX_CamSwitchMulti- <br> Timed_dc | 0 <br> 1 <br> 2 <br> 3 <br> 4 <br> 5 <br> 6 <br> 7 <br> 8 <br> 9 | BYTE | 0 |

${ }^{1}$ ) camType additionally to camToTrack identifies the type of each cam switch and enables the use of a specific function block from the Extended Cam Switch Library.
${ }^{2}$ ) camType parameters and the Extended Camswitch Library are only available for CI511ETHCAT and CI512-ETHCAT with device index C0 and above.

## Group parameters for the analog part

| Name | Value | Internal value | Internal value, <br> type | Default |
| :--- | :--- | :--- | :--- | :--- |
| Analog data <br> format | Standard | 0 | BYTE | 0 |
| Behaviour AO at <br> comm. error *) | Off <br> Last value <br> Last value 5 s <br> Last value 10 s <br> Substitute value <br> Substitute value <br> 5 s <br> Substitute value | 11 <br> 10 | 12 | BYTE |

${ }^{*}$ ) The parameter Behaviour AO at comm. error is only analyzed if the Failsafe-mode is ON.

Channel parameters for the analog inputs (4x)

| Name | Value | Internal value | Internal value, <br> type | Default |
| :--- | :--- | :--- | :--- | :--- |
| Input 0, channel <br> configuration | see $^{1}$ ) | see $^{1}$ ) | BYTE | 0 |
| Input 0, check <br> channel | see $^{2}$ ) | see $^{2}$ ) | BYTE | 0 |
| $:$ | $:$ | $:$ | $:$ | $:$ |
| $:$ | $:$ | see ${ }^{1}$ ) | BYTE | 0 |
| Input 3, channel <br> configuration | see $^{1}$ ) | see $^{2}$ ) | BYTE | 0 |
| Input 3, channel <br> configuration | see $^{2}$ ) |  |  | $:$ |

Channel configuration ${ }^{1}$ )

| Internal value | Operating modes of the analog inputs, individually configurable |
| :---: | :---: |
| 0 (default) | Not used |
| 1 | 0 ... 10 V |
| 2 | Digital input |
| 3 | $0 \ldots 20 \mathrm{~mA}$ |
| 4 | $4 \ldots 20 \mathrm{~mA}$ |
| 5 | -10 V ... +10 V |
| 8 | 2-wire Pt100-50 ${ }^{\circ} \mathrm{C} \ldots+400{ }^{\circ} \mathrm{C}$ |
| 9 | 3-wire Pt100-50 ${ }^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ *) |
| 10 | 0 V ... 10 V (voltage diff.) ${ }^{\text {* }}$ ) |
| 11 | -10 V ... +10 V (voltage diff.) *) |
| 14 | 2-wire Pt100-50 ${ }^{\circ} \mathrm{C} \ldots+70{ }^{\circ} \mathrm{C}$ |
| 15 | 3-wire Pt100-50 ${ }^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ *) |


| Internal value | Operating modes of the analog inputs, individually configurable |
| :--- | :--- |
| 16 | 2-wire Pt1000 $-50{ }^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ |
| 17 | 3-wire Pt1000 $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}{ }^{*}$ ) |
| 18 | 2-wire Ni1000 $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ |
| 19 | 3-wire Ni1000 $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}{ }^{*}$ ) |
|  | *) In the operating modes with 3-wire configuration or with differential inputs, <br> two adjacent analog inputs belong together (e.g. the channels 0 and 1). In <br> these cases, both channels are configured in the desired operating mode. <br> The lower address must be the even address (channel 0). The next higher <br> address must be the odd address (channel 1). The converted analog value is <br> available at the higher address (channel 1). |

Table 229: Channel monitoring ${ }^{2}$ )

| Internal Value | Check channel |
| :--- | :--- |
| 0 | Plausibility, wire break, short circuit |
| 3 | not used |

## Channel parameters for the analog outputs (2x)

| Name | Value | Internal value | Internal value, <br> type | Default |
| :--- | :--- | :--- | :--- | :--- |
| Output 0, <br> channel configu- <br> ration | see $^{3}$ ) | see $^{3}$ ) | BYTE | 0 |
| Output 0, check <br> channel | see $^{4}$ ) | see $^{4}$ ) | BYTE | 0 |
| Output 0, substi- <br> tute value | see $^{5}$ ) | see $^{5}$ ) | WORD | 0 |
| Output 1, <br> channel configu- <br> ration | see $^{3}$ ) | see $^{3}$ ) | BYTE | 0 |
| Output 1, check <br> channel | see $^{4}$ ) | see $^{4}$ ) | BYTE | 0 |
| Output 1, substi- <br> tute value | see $^{5}$ ) | see $^{5}$ ) | WORD | 0 |

Table 230: Channel configuration ${ }^{3}$ )

| Internal value | Operating modes of the analog outputs, individually configu- <br> rable |
| :--- | :--- |
| 0 | Not used (default) |
| 128 | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ |
| 129 | $0 \ldots 20 \mathrm{~mA}$ |
| 130 | $4 \ldots 20 \mathrm{~mA}$ |

Table 231: Channel monitoring ${ }^{4}$ )

| Internal value | Check channel |
| :--- | :--- |
| 0 | Plausibility, wire break, short circuit |
| 3 | None |

Table 232: Substitute value ${ }^{5}$ )

| Intended behavior of <br> output channel when the <br> control system stops | Required setting of the module <br> parameter "Behaviour of outputs <br> in case of a communication <br> error" | Required setting of <br> the channel parameter <br> "Substitute value" |
| :--- | :--- | :--- |
| Output OFF | Off | 0 |
| Last value infinite | Last value | 0 |
| Last value for 5 s | Last value 5 s | 0 |
| Last value for 10 s | Last value 10 s | 0 |
| Substitute value infinite | Substitute value | Depending on configura- <br> tion |
| Substitute value for 5 s | Substitute value 5 s | Depending on configura- <br> tion |
| Substitute value for 10 s | Substitute value 10 s | Depending on configura- <br> tion |

## Group parameters for the digital part

| Name | Value | Internal value | Internal value, type | Default |
| :---: | :---: | :---: | :---: | :---: |
| Input delay | $\begin{aligned} & 0.01 \mathrm{~ms} \\ & 1 \mathrm{~ms} \\ & 8 \mathrm{~ms} \\ & 32 \mathrm{~ms} \end{aligned}$ | $\begin{array}{\|l\|} \hline 0 \\ 1 \\ 2 \\ 3 \end{array}$ | BYTE | $\begin{aligned} & 0.01 \mathrm{~ms} \\ & 0 \times 00 \end{aligned}$ |
| Detect short circuits at outputs | $\begin{aligned} & \text { Off } \\ & \text { On } \end{aligned}$ | $\begin{array}{\|l\|} \hline 0 \\ 1 \end{array}$ | BYTE | $\begin{aligned} & \text { On } \\ & 0 \times 01 \end{aligned}$ |
| Behaviour DO at comm. error *) | Off <br> Last value <br> Last value 5 sec <br> Last value 10 sec <br> Substitute value <br> Substitute 5 sec <br> Substitute 10 sec | $\begin{aligned} & 0 \\ & 1 \\ & 6 \\ & 11 \\ & 11 \\ & 2 \\ & 7 \\ & 12 \end{aligned}$ | BYTE | $\begin{array}{\|l} \hline \text { Off } \\ 0 \times 00 \end{array}$ |
| Substitute value at output | 0 ... 255 | 00h ... FFh | BYTE | $\begin{aligned} & \hline 0 \\ & 0 \times 0000 \end{aligned}$ |

*) The parameter behaviourDOatCommunicationFault is only analyzed if the Failsafe-mode is ON.

## Diagnosis

In cases of short circuit or overload, the digital outputs are turned off. The modules performs reactivation automatically. Thus an acknowledgement of the errors is not necessary. The error message is stored via the LED.


| E1 ... E4 | d1 | d2 | d3 | d4 | Identifier $000 \ldots$ | AC500- <br> Display$\|<-$ Display | <- Display in |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err | $\begin{aligned} & \text { PS501 } \\ & \text { PLC } \\ & \text { Browser } \end{aligned}$ |  |
| $\begin{array}{\|l} \hline \text { Byte } 6 \\ \text { Bit } \\ 6 \text {... } 7 \end{array}$ | - | Byte 3 | Byte 4 | Byte 5 | $\begin{aligned} & \hline \text { Byte } 6 \\ & \text { Bit } \\ & 0 \text {... } 5 \end{aligned}$ | ETHCAT <br> Diagnosis block |  |
| Class | Interface | Device | Module | Channel | Error identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) | ${ }^{4}$ ) |  |  |  |
| 4 | - | 31 | 31 | 31 | 46 | Voltage feedback on activated digital outputs $\left.{ }^{4}\right)$ | Check terminals |
| Channel error digital |  |  |  |  |  |  |  |
| 4 | - | 31 | 2 | $0 . .7$ | 46 | Voltage feedback on deactivated digital output ${ }^{5}$ ) | Check terminals |
| 4 | - | 31 | 2 | $0 . .7$ | 47 | Short circuit at digital output | Check terminals |
| Channel error analog |  |  |  |  |  |  |  |
| 4 | - | 31 | 1 | $0 . .3$ | 48 | Analog value overflow or broken wire at an analog input | Check value or check terminals |
| 4 | - | 31 | 1 | $0 . .3$ | 7 | Analog value underflow at an analog input | Check value |
| 4 | - | 31 | 1 | $0 . .3$ | 47 | Short circuit at an analog input | Check terminals |
| 4 | - | 31 | 3 | $0 . .1$ | 48 | Analog value overflow at an analog output | Check output value |
| 4 | - | 31 | 3 | $0 . .1$ | 7 | Analog value underflow at an analog output | Check output value |

Remarks:

| $\left.{ }^{1}\right)$ | In AC500 the following interface identifier applies: <br> "-" = Diagnosis via bus-specific function blocks; $0 \ldots 4$ or $10=$ Position of the <br> Communication Module; $14=1 / O$ bus; $31=$ Module itself <br> The identifier is not contained in the CI511-ETHCAT diagnosis block. |
| :--- | :--- |
| ${ }^{2}$ ) | With "Device" the following allocation applies: <br> $31=$ Module itself or ADR = Hardware address (e. g. of the DC551) |


| ${ }^{3}$ ) | With "Module" the following allocation applies dependent of the master: <br> $31=$ Module itself (Module error) or Module type (1=AI, 2=DO, 3=AO; channel <br> error) |
| :--- | :--- |
| ${ }^{4}$ ) | Diagnosis message appears for the whole output group and not per channel. <br> The message occurs if the output channel is already active. |
| $\left.{ }^{5}\right)$ | Diagnosis message appears per channel. The message occurs if the output <br> channel is not active. |

## State LEDs

The LEDs are located at the front of module. There are 2 different groups:

- The 5 system LEDs (PWR, NET, DC, S-ERR and I/O-Bus) show the operation state of the module and display possible errors.
- The 27 process LEDs (UP, UP3, inputs, outputs, CH-ERR1 to CH-ERR3) show the process supply voltage and the states of the inputs and outputs and display possible errors.

Table 233: States of the 5 system LEDs

| LED | Color | Off | On | Flashing | 1x Flash | 2x Flash |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| PWR/RUN | Green | Error in the internal supply voltage or process voltage missing | Internal supply voltage OK | Module is not configured | -- | -- |
|  | Yellow | -- | -- | -- | -- | -- |
| NET | Green | Init | Operational | Pre-operational | Safe-operational | -- |
|  | Red | No error | PDI <br> Watchdog <br> Timeout | Invalid Configuration | Unsolicited State Change | Application time out |
| DC *) | Green | Distributed Clock not active | Distributed Clock active | -- | -- | -- |
|  | Red | -- | -- | -- | -- | -- |
| S-ERR | Red | No error | Internal error | -- | -- | -- |
| I/O-Bus | Green | No communication interface modules connected or communication error | --- | --- | -- | -- |
| ETH1 | Green | No EtherCAT connection | Link OK <br> No data transfer | Link OK <br> Data transfer OK | -- | -- |
|  | Yellow | -- | -- | -- | -- | -- |


| LED | Color | Off | On | Flashing | 1x Flash | 2x Flash |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| ETH2 | Green | No <br> EtherCAT <br> connection | Link OK <br> No data <br> transfer | Link OK <br> Data <br> transfer OK | -- | -- |
|  | Yellow | -- | -- | -- | -- | -- |

*) The state of this LED is only significant if the cam switch functionality is enabled

Table 234: States of the 27 process LEDs

| LED | Color | OFF | ON | Flashing |
| :--- | :--- | :--- | :--- | :--- |
| AIO ... AI3 | Yellow | Input is OFF | Input is ON <br> (brightness <br> depends on the <br> value of the <br> analog signal) | -- |
| AO0 ... AO1 | Yellow | Output is OFF | Output is ON <br> (brightness <br> depends on the <br> value of the <br> analog signal) | -- |
| DI0 ... DI7 | Yellow | Input is OFF | Input is ON (the <br> input voltage is <br> even displayed if <br> the supply <br> voltage is OFF) | -- |
| DO0 ... DO7 | Yellow | Green | Output is OFF <br> UPress supply <br> voltage missing | Output is ON <br> Process supply <br> voltage OK and <br> initialization fin- <br> ished |
| UP3 | Green | Process supply <br> voltage missing | Process supply <br> voltage OK | --- |
| CH-ERR1 to CH- <br> ERR3 | Red | No error or <br> process supply <br> voltage missing | Internal error | Error on one <br> channel of the <br> corresponding <br> group |

## Measuring ranges

Input ranges voltage, current and digital input

| Range | $0 \mathrm{~V} . .+10$ | -10 V ... | $0 \text { mA ... }$ | $4 \mathrm{~mA} \ldots 20$ | Digital | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  | Decimal | Hex. |
| Overflow | > 11.7589 | > 11.7589 | > 23.5178 | > 22.8142 |  | 32767 | 7FFF |
| Measured value too high | $\begin{aligned} & 11.7589 \\ & : \\ & 10.0004 \end{aligned}$ | $\begin{aligned} & 11.7589 \\ & : \\ & 10.0004 \end{aligned}$ | $\begin{aligned} & 23.5178 \\ & : \\ & 20.0007 \end{aligned}$ | $\begin{aligned} & 22.8142 \\ & : \\ & 20.0006 \end{aligned}$ |  | $\begin{aligned} & 32511 \\ & : \\ & 27649 \end{aligned}$ | 7EFF 6C01 |


| Range | $0 \text { V ... +10 }$ | -10 V ... | $0 \mathrm{~mA} . .$ | $4 \mathrm{~mA} . .20$ | Digital | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  | Decimal | Hex. |
| Normal range <br> Normal range or measured value too low | $\begin{aligned} & 10.0000 \\ & : \\ & 0.0004 \end{aligned}$ | $\begin{aligned} & 10.0000 \\ & : \\ & 0.0004 \end{aligned}$ | $\begin{aligned} & 20.0000 \\ & : \\ & 0.0007 \end{aligned}$ | $\begin{aligned} & 20.0000 \\ & : \\ & 4.0006 \end{aligned}$ | On | $\begin{aligned} & 27648 \\ & : \\ & 1 \end{aligned}$ | $\begin{aligned} & 6 \mathrm{C} 00 \\ & : \\ & 0001 \end{aligned}$ |
|  | 0.0000 | 0.0000 | 0 | 4 | Off | 0 | 0000 |
|  | $\begin{array}{\|l\|} \hline-0.0004 \\ -1.7593 \end{array}$ | $\begin{aligned} & -0.0004 \\ & : \\ & : \\ & -10,0000 \end{aligned}$ |  | $\begin{aligned} & 3.9994 \\ & 1.1858 \end{aligned}$ |  | -1 \|-4864 |-27648 | $\begin{aligned} & \text { FFFF } \\ & \text { ED00 } \\ & : \\ & 9400 \end{aligned}$ |
| Measured value too low |  | $-10.0004$ -11.7589 |  |  |  | $\begin{aligned} & -27649 \\ & : \\ & -32512 \end{aligned}$ | $\begin{aligned} & \text { 93FF } \\ & : \\ & 8100 \end{aligned}$ |
| Underflow | < 1.7593 | <-11.7589 | < 0.0000 | < 1.1858 |  | -32768 | 8000 |

The represented resolution corresponds to 16 bits.

Input ranges resistance temperature detector

| Range | Pt100 / Pt1000 | Ni1000 | Digital value |  |
| :---: | :---: | :---: | :---: | :---: |
|  |  |  | Decimal | Hex. |
| Overflow | $>+450.0^{\circ} \mathrm{C}$ | $>+160.0{ }^{\circ} \mathrm{C}$ | 32767 | 7FFF |
| Measured value too high | $\begin{aligned} & +450.0^{\circ} \mathrm{C} \\ & : \\ & +400.1^{\circ} \mathrm{C} \end{aligned}$ |  | $\begin{aligned} & 4500 \\ & : \\ & 4001 \end{aligned}$ | $\begin{aligned} & 1194 \\ & : \\ & \text { OFA1 } \end{aligned}$ |
|  |  | $\begin{aligned} & +160.0^{\circ} \mathrm{C} \\ & : \\ & +150.1^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & 1600 \\ & : \\ & 1501 \end{aligned}$ | $\begin{array}{\|l\|} \hline 0640 \\ : \\ \text { 05DD } \\ \hline \end{array}$ |
|  |  |  | $\begin{aligned} & 800 \\ & : \\ & 701 \end{aligned}$ | $\begin{aligned} & 0320 \\ & : \\ & \text { 02BD } \end{aligned}$ |
| Normal range | $\begin{aligned} & +400.0^{\circ} \mathrm{C} \\ & : \\ & : \\ & \vdots \\ & +0.1^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & +150.0^{\circ} \mathrm{C} \\ & : \\ & : \\ & +0.1^{\circ} \mathrm{C} \end{aligned}$ | 4000 1500 700 $:$ 1 | $\begin{aligned} & \hline \text { OFAO } \\ & \text { 05DC } \\ & \text { 02BC } \\ & : \\ & 0001 \end{aligned}$ |
|  | $0.0{ }^{\circ} \mathrm{C}$ | $0.0{ }^{\circ} \mathrm{C}$ | 0 | 0000 |
|  | $\begin{aligned} & -0.1^{\circ} \mathrm{C} \\ & : \\ & -50.0^{\circ} \mathrm{C} \end{aligned}$ | $-0.1^{\circ} \mathrm{C}$ <br> $-50,0^{\circ} \mathrm{C}$ | $\begin{aligned} & \hline-1 \\ & : \\ & -500 \end{aligned}$ | $\begin{aligned} & \text { FFFF } \\ & : \\ & \text { FEOC } \end{aligned}$ |


| Range | $\begin{aligned} & \text { Pt100 / Pt1000 } \\ & -50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \text { Ni1000 } \\ & -50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C} \end{aligned}$ | Digital value |  |
| :---: | :---: | :---: | :---: | :---: |
|  |  |  | Decimal | Hex. |
| Measured value | $-50.1{ }^{\circ} \mathrm{C}$ | $-50.1{ }^{\circ} \mathrm{C}$ | -501 | FEOB |
|  |  |  |  |  |
|  | $-60.0{ }^{\circ} \mathrm{C}$ | $-60.0{ }^{\circ} \mathrm{C}$ | -600 | FDA8 |
| Underflow | <-60.0 ${ }^{\circ} \mathrm{C}$ | <-60.0 ${ }^{\circ} \mathrm{C}$ | -32768 | 8000 |

## Output ranges voltage and current

| Range | -10 V ... +10 V | $0 \mathrm{~mA} . . .20 \mathrm{~mA}$ | $\sqrt{4 \mathrm{~mA} \ldots 20}$ | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  | Decimal | Hex. |
| Overflow | 0 V | 0 mA | 0 mA | > 32511 | > 7EFF |
| Measured value too high | $\begin{aligned} & 11.7589 \mathrm{~V} \\ & : \\ & 10.0004 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 23.5178 \mathrm{~mA} \\ & : \\ & 20.0007 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 22.8142 \mathrm{~mA} \\ & : \\ & 20.0006 \mathrm{~mA} \end{aligned}$ | $\begin{array}{\|l} \hline 32511 \\ : \\ 27649 \\ \hline \end{array}$ | $\begin{aligned} & \text { 7EFF } \\ & : \\ & 6 \mathrm{C} 01 \end{aligned}$ |
| Normal range | $\begin{aligned} & \hline 10.0000 \mathrm{~V} \\ & : \\ & 0.0004 \mathrm{~V} \\ & \hline \end{aligned}$ | $\begin{aligned} & 20.0000 \mathrm{~mA} \\ & : \\ & 0,0007 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 20.0000 \mathrm{~mA} \\ & : \\ & 4.0006 \mathrm{~mA} \end{aligned}$ | 27648 | $\begin{aligned} & \hline 6 \mathrm{COO} \\ & : \\ & 0001 \end{aligned}$ |
|  | 0.0000 V | 0.0000 mA | 4.0000 mA | 0 | 0000 |
|  | $\begin{aligned} & -0.0004 \mathrm{~V} \\ & : \\ & -10.0000 \mathrm{~V} \end{aligned}$ |  | $\begin{aligned} & 3.9994 \mathrm{~mA} \\ & 0 \mathrm{~mA} \\ & 0 \mathrm{~mA} \end{aligned}$ | $\begin{array}{\|l} \hline-1 \\ -6912 \\ -27648 \end{array}$ | $\begin{aligned} & \text { FFFF } \\ & \text { E500 } \\ & 9400 \end{aligned}$ |
| Measured value too low | $\begin{aligned} & -10.0004 \mathrm{~V} \\ & : \\ & -11.7589 \mathrm{~V} \end{aligned}$ |  |  | $\begin{aligned} & -27649 \\ & \vdots \\ & -32512 \end{aligned}$ | $\begin{aligned} & \text { 93FF } \\ & : \\ & 8100 \end{aligned}$ |
| Underflow | 0 V | 0 mA | 0 mA | <-32512 | < 8100 |

The represented resolution corresponds to 16 bits.

## Technical data

## Technical data of the module

The system data of AC500 and S500 are applicable to the standard version Chapter 5.1.2 "System data AC500" on page 166.
The system data of AC500-XC are applicable to the XC version Chapter 5.1.3 "System data AC500-XC" on page 169.
Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter | Value |
| :---: | :---: |
| Bus connection | $2 \times \mathrm{RJ45}$ |
| Technology | Hilscher NETX 100 |
| Transfer rate | 10/100 Mbit/s (full-duplex) |
| Transfer method | According to Ethernet II, IEEE 802.3 |
| Ethernet | 100 base-TX, internal switch, $2 \times$ RJ45 socket |
| Expandability (S500 I/O modules) | Up to 10 S500 I/O modules (Index C0 and above), not available (Index below C0) |
| Indicators | 5 LEDs for state indication |
| Adjusting elements | 2 rotary switches (used for future topology extensions) |
| Quantity of input/output data | CI512-ETHCAT: 10 bytes input and 14 bytes output <br> CI511-ETHCAT: 18 bytes input and 18 bytes output |
| Limit of data for input and output | 144 byte |
| Acyclic services | SDO (1500 bytes max.) <br> Emergency ECAT_SLV_DIAG |
| Protective functions (according to CODESYS) | Protected against: <br> - short circuit <br> - reverse supply <br> - overvoltage <br> - reverse polarity <br> Galvanic isolation to network |


| Parameter | Value |
| :---: | :---: |
| Process supply voltage UP/UP3 |  |
| Rated value | 24 V DC (for inputs and outputs) |
| Max. load for the terminals | 10 A |
| Protection against reversed voltage | Yes |
| Rated protection fuse on UP/UP3 | 10 A fast |
| Galvanic isolation | Ethernet interface against the rest of the module |
| Inrush current from UP (at power up) | On request |
| Current consumption via UP (normal operation) | 0.2 A |
| Current consumption via UP3 | 0.06 A + 0.5 A max. per output |
| Connections | Terminals 1.8 and 2.8 for +24 V (UP) <br> Terminal 3.8 for +24 V (UP3) <br> Terminals 1.9, 2.9 and 3.9 for 0 V (ZP) |
| Max. power dissipation within the module | 6 W |
| Number of digital inputs | 8 |
| Number of digital outputs | 8 |
| Number of analog inputs | 4 |


| Parameter | Value |
| :--- | :--- |
| Number of analog outputs | 2 |
| Reference potential for all digital inputs and <br> outputs | Negative pole of the supply voltage, signal <br> name ZP |
| Diagnosis | See Diagnosis and Displays «\% Chapter <br> $5.2 .6 .4 .1 .9 ~ " D i a g n o s i s " ~ o n ~ p a g e ~ 1039 ~$ |
| Operation and error displays | 32 LEDs (totally) |
| Weight (without terminal unit) | ca. 125 g |
| Mounting position | Horizontal <br> Or vertical with derating (output load reduced to <br> $50 \%$ at $+40^{\circ} \mathrm{C}$ per group) |
| Cooling | The natural convection cooling must not be hin- <br> dered by cable ducts or other parts in the con- <br> trol cabinet. |

## NOTICE!

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and temporary overvoltage up to 30 V DC.

## Multiple overloads

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an internal smart high-side switch.

## Technical data of the digital inputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 8 |
| Distribution of the channels into groups | 1 group of 8 channels |
| Terminals of the channels DI0 ... DI7 | Terminals $2.0 \ldots 2.7$ |
| Reference potential for all inputs | Terminals $1.9 \ldots 3.9$ (Negative pole of the <br> supply voltage, signal name ZP) |
| Indication of the input signals | 1 yellow LED per channel, the LED is ON when <br> the input signal is high (signal 1) |
| Input type (according EN 61131-2) | Type 1 |
| Input delay (0->1 or 1->0) | Typ. 0.1 ms, configurable from $0.1 \mathrm{~ms} \ldots 32 \mathrm{~ms}$ |
| Input signal voltage | 24 V DC |
|  | $0-$ Signal |
| Undefined Signal | $-3 \mathrm{~V} \ldots+5 \mathrm{~V}$ |
|  | 1-Signal |
| Ripple with signal 0 | $+5 \mathrm{~V} \ldots<+15 \mathrm{~V}$ |
| Ripple with signal 1 | Within $-3 \mathrm{~V} \ldots+5 \mathrm{~V}$ |
| Input current per channel | Within $+15 \mathrm{~V} \ldots+30 \mathrm{~V}$ |
|  | Input voltage +24 V |


| Parameter |  | Value |
| :--- | :--- | :--- |
|  | Input voltage +5 V | $>1 \mathrm{~mA}$ |
|  | Input voltage +15 V | $>2 \mathrm{~mA}$ |
|  | Input voltage +30 V | $<8 \mathrm{~mA}$ |
| Max. cable length |  |  |
|  | Shielded | 1000 m |
|  | Unshielded | 600 m |

## Technical data of the digital outputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 8 |
| Distribution of the channels into groups | 1 group of 8 channels |
| Terminals of the channels DO0 ... DO7 | Terminals 3.0 ... 3.7 |
| Reference potential for all outputs | Terminals 1.9 ... 3.9 (Negative pole of the supply voltage, signal name ZP) |
| Common power supply voltage | For all outputs terminal 3.8 (positive pole of the supply voltage, signal name UP3) |
| Output voltage for signal 1 | UP3 (-0.8 V) |
| Output delay (0->1 or 1->0) | On request |
| Output current |  |
| Rated value per channel | 500 mA at UP3 $=24 \mathrm{~V}$ |
| Max. value (all channels together) | 4 A |
| Leakage current with signal 0 | $<0.5 \mathrm{~mA}$ |
| Fuse for UP3 | 10 A fast |
| Demagnetization with inductive DC load | Via internal varistors (see figure below this table) |
| Output switching frequency |  |
| With resistive load | On request |
| With inductive loads | Max. 0.5 Hz |
| With lamp loads | 11 Hz max. at 5 W max. |
| Short-circuit-proof / overload-proof | Yes |
| Overload message ( $1>0.7$ A) | Yes, after ca. 100 ms |
| Output current limitation | Yes, automatic reactivation after short circuit/overload |
| Resistance to feedback against 24 V signals | Yes (software-controlled supervision) |
| Max. cable length |  |
| Shielded | 1000 m |
| Unshielded | 600 m |

The following drawing shows the circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off.


Fig. 202: Digital input/output (circuit diagram)
1 Digital output
2 Varistors for demagnetization when inductive loads are turned off

## Technical data of the analog inputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 4 |
| Distribution of channels into groups | 1 group with 4 channels |
| Connection if channels $\mathrm{AlO}+\ldots \mathrm{Al} 3+$ | Terminals 1.0 ... 1.3 |
| Reference potential for $\mathrm{AlO}+\ldots \mathrm{Al} 3+$ | Terminal 1.4 (AI-) for voltage and RTD measurement <br> Terminals 1.9, 2.9 and 3.9 for current measurement |
| Input type |  |
| Unipolar | Voltage 0 V ... 10 V, current or Pt100/Pt1000/ Ni1000 |
| Bipolar | Voltage -10 V ... +10 V |
| Galvanic isolation | Against Ethernet network |
| Configurability | 0 V ... $10 \mathrm{~V},-10 \mathrm{~V} \ldots+10 \mathrm{~V}, 0 / 4 \mathrm{~mA} . . .20 \mathrm{~mA}$, Pt100/1000, Ni1000 (each input can be configured individually) |
| Channel input resistance | Voltage: > $100 \mathrm{k} \Omega$ <br> Current: ca. $330 \Omega$ |
| Time constant of the input filter | Voltage: $100 \mu \mathrm{~s}$ Current: $100 \mu \mathrm{~s}$ |
| Indication of the input signals | 1 LED per channel (brightness depends on the value of the analog signal) |
| Conversion cycle | 1 ms (for 4 inputs + 2 outputs); with RTDs Pt/ Ni... 1 s |
| Resolution | Range 0 ... 10 V : 12 bits <br> Range $-10 \ldots+10 \mathrm{~V}$ : 12 bits including sign <br> Range 0 ... 20 mA : 12 bits <br> Range 4 ... 20 mA : 12 bits <br> Range RTD (Pt100, PT1000, Ni1000): $+0.1^{\circ} \mathrm{C}$ |
| Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range | Typ. 0.5 \%, max. 1 \% |


| Parameter | Value |
| :--- | :--- |
| Relationship between input signal and hex <br> code | (y) Chapter 5.2.6.4.1.11.2 "Input ranges resist- <br> ance temperature detector" on page 1043 |
| Unused inputs | Are configured as "unused" (default value) |
| Overvoltage protection | Yes |

## Technical data of the analog inputs, if used as digital inputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | Max. 4 |
| Distribution of channels into groups | 1 group of 4 channels |
| Connections of the channels $\mathrm{AlO+} \ldots \mathrm{Al} 3+$ | Terminals $1.0 \ldots 1.3$ |
| Reference potential for the inputs | Terminals $1.9,2.9$ and $3.9(\mathrm{ZP})$ |
| Indication of the input signals | 1 LED per channel |
| Input signal voltage | 24 V DC |
|  | Signal 0 |
|  | Undefined signal |
|  | Signal 1 |
| Input current per channel | $+5 \mathrm{~V} \ldots+13 \mathrm{~V}$ |
|  | Input voltage +24 V |
|  | Input voltage +5 V |
| Input voltage +15 V | Typ. 7 mA |
|  | Input voltage +30 V |
| Input resistance | Typ. 1.4 mA |

## Technical data of the analog outputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 2 |
| Distribution of channels into groups | 1 group for 2 channels |
| Connection of the channels AO0+...AO1+ | Terminals $1.5 \ldots 1.6$ |
| Reference potential for AO0+ ... AO1+ | Terminal 1.7 (AO-) for voltage outputTerminals <br> $1.9,2.9$ and 3.9 (ZP) for current output |
| Output type |  |
|  | Unipolar |
| Bipolar | Vorrent |
| Galvanic isolation | Against Ethernet network |
| Configurability | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}, 0 \mathrm{~mA} \ldots 20 \mathrm{~mA}, 4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ |
| (each output can be configured individually) |  |
| Output resistance (load), <br> as current output | $0 \Omega \ldots 500 \Omega$ |


| Parameter | Value |
| :--- | :--- |
| Output loadability, <br> as voltage output | $\pm 10 \mathrm{~mA}$ max. |
| Indication of the output signals | 1 LED per channel (brightness depends on the <br> value of the analog signal) |
| Resolution | 12 bits including sign |
| Settling time for full range change (resistive <br> load, output signal within specified tolerance) | Typ. 5 ms |
| Conversion error of the analog values <br> caused by non-linearity, adjustment error <br> at factory and resolution within the normal <br> range | Typ. 0.5 \%, max. 1 \% |
| Relationship between input signal and hex <br> code | Table Output Ranges Voltage and Current <br> \& Chapter 5.2.6.4.1.11.3 "Output ranges <br> voltage and current" on page 1044 |
| Unused outputs | Are configured as unused (default value) and <br> can be left open-circuited |

## Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm


## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 220 900 R0001 | CI511-ETHCAT, EtherCAT communi- <br> cation interface module, 8 DI, 8 DO, <br> 4 Al and 2 AO | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.6.4.2 CI512-ETHCAT

Features

- 8 digital inputs 24 V DC
- 8 digital outputs $24 \mathrm{~V} D, 0.5 \mathrm{~A}$ max.
- 8 configurable digital inputs/outputs $24 \mathrm{~V} \mathrm{DC}, 0.5 \mathrm{~A}$ max.
- Cam switch functionality (see also Extended Cam Switch Library)
- Extended Cam switch functionality *)
(see also Extended Cam Switch Library)
- Module-wise galvanically isolated
- Expandability with up to 10 S 500 I/O modules *)
*) Applicable for device index C0 and above.


1 I/O bus
2 Allocation between terminal number and signal name
38 yellow LEDs to display the signal states of the digital configurable inputs/outputs (DC0 ... DC7)
48 yellow LEDs to display the signal states of the digital inputs (DIO ... DI7)
58 yellow LEDs to display the signal states of the digital outputs (DO0 ... DO7)
62 green LEDs to display the supply voltage UP and UP3
73 red LEDs to display errors (CH-ERR1, CH-ERR2, CH-ERR3)
85 System LEDs: PWR/RUN, NET, DC, S-ERR, I/O-Bus
92 rotary switches (reserved for future extensions)
10 Label
11 Ethernet interfaces (ETH1, ETH2) on the terminal unit
12 Terminal unit
13 DIN rail

## Intended purpose

The EtherCAT communication interface module CI512-ETHCAT is used as decentralized I/O module in EtherCAT networks. The network connection is performed via 2 RJ45 connectors which are integrated in the terminal unit. The communication interface module contains 24 I/O channels with the following properties:

- 8 digital configurable inputs/outputs in 1 group (1.0 ... 1.7)
- 8 digital inputs 24 V DC in 1 group (2.0 ... 2.7)
- 8 digital outputs 24 V DC in 1 group (3.0 ... 3.7)
- Cam switch functionality

The inputs/outputs are galvanically isolated from the Ethernet network. There is no potential separation between the channels. The configuration of the configurable digital inputs/outputs is performed by software.

## Functionality

| Parameter | Value |
| :---: | :---: |
| Interface | Ethernet |
| Protocol | EtherCAT |
| Power supply | From the process supply voltage UP |
| Supply of the electronic circuitry of the I/O expansion modules attached | Through the I/O bus interface (I/O bus) |
| Rotary switches | Not used; reserved for future extensions |
| Configurable digital inputs/outputs | 8 (configurable via software) |
| Digital inputs | 8 (24 V DC; delay time configurable via software) |
| Digital outputs | 8 (24 V DC, 0.5 A max.) |
| LED displays | For system displays, signal states, errors and power supply |
| External supply voltage | Via terminals ZP, UP and UP3 (process supply voltage 24 V DC) |
| Effect of incorrect input terminal connection | Wrong or no signal detected, no damage up to 35 V |
| Required terminal unit | TU507 or TU508 ${ }^{\aleph}$ Chapter 5.2.7.2 "TU507ETH and TU508-ETH for Ethernet communication interface modules" on page 1270 |

## Connections

The Ethernet communication interface module CI512-ETHCAT is plugged on the I/O terminal unit TU507-ETH or TU508-ETH. Properly seat the module and press until it locks in place. The terminal unit is mounted on a DIN rail or with 2 screws plus the additional accessory for wall mounting (TA526).

The connection of the I/O channels is carried out using the 30 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

The terminals 1.8 and 2.8 as well as $1.9,2.9$ and 3.9 are electrically interconnected within the terminal unit and have always the same assignment, independent of the inserted module:
Terminals 1.8 and 2.8: Process supply voltage UP $=+24 \mathrm{~V}$ DC
Terminal 3.8: Process supply voltage UP3 = +24 V DC
Terminals 1.9, 2.9 and 3.9: Process supply voltage $\mathrm{ZP}=0 \mathrm{~V}$

With a separate UP3 power supply, the digital outputs can be switched off externally. This way, an emergency-off functionality can be realized.

The assignment of the other terminals:

| Terminals | Signal | Description |
| :--- | :--- | :--- |
| $1.0 \ldots 1.7$ | DC0 $\ldots$ DC7 | 8 digital inputs/outputs (con- <br> figurable via software) |
| $2.0 \ldots 2.7$ | DI0 ... DI7 | 8 digital inputs (delay time <br> configurable via software) |
| $3.0 \ldots 3.7$ | DO0 ... DO7 | 8 digital outputs |

## WARNING!

## Removal/Insertion under power

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.
Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.

The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

## Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

DC 01.0
DC 11.1


DC 41.4

DC 51.5

DC 61.6

DC 71.7
DC 71.7



Fig. 203: Connection of the communication interface module CI512-ETHCAT
18 digital configurable inputs/outputs 24 V DC
28 digital inputs 24 V DC
38 digital outputs 24 V DC

In case of voltage feedback, 2 cases are distinguished:

1. The outputs are already active

The output group will be switched off. A diagnosis message will appear. After 5 seconds, the module tries automatic reactivation.
2. The outputs are not active

Only the output with voltage feedback will not be set to active. A diagnosis message will appear.

## CAUTION!

The process supply voltage must be included within the grounding concept of the plant (e. g. grounding of the negative pole).

The module provides several diagnosis functions ${ }^{*} \geqslant$ Chapter 5.2.6.4.2.10 "Diagnosis" on page 1060.

## Assignment of the Ethernet ports

The terminal unit for the communication interface module provides two Ethernet interfaces with the following pin assignment. The pin assignment is used for the EtherCAT master (communication module CM5xy-ETHCAT) as well.

| Pin assignment | Interface | Pin | Signal | Description |
| :---: | :---: | :---: | :---: | :---: |
|  |  | 1 | TxD+ | Transmit data + |
|  |  | 2 | TxD- | Transmit data - |
|  |  | 3 | RxD+ | Receive data + |
|  |  | 4 | NC | Not connected |
|  |  | 5 | NC | Not connected |
|  |  | 6 | RxD- | Receive data - |
|  |  | 7 | NC | Not connected |
|  |  | 8 | NC | Not connected |
|  |  | Shield | Cable shield | Functional earth |

In corrosive environment, please protect unused connectors using the TA535 accessory.
Not supplied with this device.
*3 Further information about wiring and cable types

The EtherCAT network differentiates between input-connectors (IN) and outputconnectors (OUT):
At the EtherCAT slaves (communication interface modules), the ETH1-connector is IN and the ETH2-connector is OUT.
At the EtherCAT master (communication module), the ETHCAT1 connector has to be used. The ETHCAT2 connector is reserved for future extensions.

## Internal data exchange

| Parameter | Value |
| :--- | :--- |
| Digital inputs (bytes) | 1 |
| Digital outputs (bytes) | 1 |
| Configurable digital inputs/outputs (bytes) | $1+1$ |

## Addressing

The Ethernet communication interface module $\mathrm{Cl} 512-\mathrm{ETHCAT}$ does not consider the position of the rotary switches at the front side of the module. The function of the rotary switches is reserved for future expansions.

## I/O configuration

In order to be able to use the CI51X-ETHCAT with device index C0 or above properly, please download the corresponding device description (.xml-)files from http://www.abb.com/plc and install them to the device repository of your Automation Builder. This will allow you to use up to 10 Expandable S500 I/O modules as well as the Extended Cam Switch Library with your CI51X-ETHCAT device.

The CI512-ETHCAT does not store configuration data itself.
The analog I/O channels are configured via software.

## Parameterization

## Module parameter

| Name | Value | Internal value | Internal value, <br> type | Default |
| :--- | :--- | :--- | :--- | :--- |
| Module ID | Internal | 49435 | WORD | 49435 |
| Parameter length | Internal | 10 | BYTE | 10 |
| Error LED / Fail- <br> safe function ${ }^{1}$ ) | On <br> Off by E4 <br> Off by E3 On + <br> failsafe Off by E4 <br> + failsafe Off by <br> E3 + failsafe | 16 <br> 17 | 17 <br> 19 | 0 |
| Check Supply | Off | 0 | 1 | BYTE |

Table 235: Error LED / Failsafe function ${ }^{1}$ )

| Setting | Description |
| :--- | :--- |
| On | Error LED lights up at errors of all error classes, Failsafe mode off |
| Off by E4 | Error LED lights up at errors of error classes E1, E2 and E3, Failsafe <br> mode off |
| Off by E3 | Error LED lights up at errors of error classes E1 and E2 auf, Failsafe <br> mode off |
| On + failsafe | Error LED lights up at errors of all error classes, Failsafe mode on *) |
| Off by E4 + failsafe | Error LED lights up at errors of error classes E1, E2 and E3, Failsafe <br> mode on *) |
| Off by E3 + failsafe | Error LED lights up at errors of error classes E1 and E2, Failsafe mode <br> on *) |

[^19]
## Group parameters of the cam switch

| Name | Value | Internal value | Internal value, type | Default |
| :---: | :---: | :---: | :---: | :---: |
| numOfUsedCams ${ }^{1}$ ) | $\begin{aligned} & \hline 0 \ldots 32 \\ & 128 \ldots 160 \end{aligned}$ | $\begin{aligned} & \hline 0 \ldots 32 \\ & 218 \ldots 160 \end{aligned}$ | WORD | 0 |
| resolution ${ }^{2}$ ) | $\begin{aligned} & 0 \ldots 2 \\ & -1 \end{aligned}$ | $\begin{aligned} & 0 \ldots 2 \\ & -1 \end{aligned}$ | DWORD | 36000 |
| zeroShift ${ }^{3}$ ) | $\begin{aligned} & 0 \ldots 2 \\ & -1 \end{aligned}$ | $\begin{aligned} & 0 \ldots 2 \\ & -1 \end{aligned}$ | DWORD | 0 |
| EncoderBitResolution ${ }^{4}$ ) | $8 \ldots 32$ | 8 ... 32 | WORD | 18 |
| Reserve | - | - | WORD | - |

## Remarks:

${ }^{1}$ ) The parameter numOfUsedCams defines the interrupt cycle time (Therefore, it takes effect to the accuracy of the track) and the behavior of the module if the DC information is lost.

| Parameter setting <br> for numOfUsed- <br> Cams | Number of cams <br> used | Interrupt cycle time | Behavior if DC infor- <br> mation is lost |
| :--- | :--- | :--- | :--- |
| 0 | 0 | $50 \mu \mathrm{~s}$ | Module changes <br> to "safe-operational" <br> state; the outputs are <br> activated trough the <br> user program |
| $1 \ldots 8$ | $1 \ldots 8$ | $80 \mu \mathrm{~s}$ | $100 \mu \mathrm{~s}$ |
| $9 \ldots 16$ | $9 \ldots 16$ | $200 \mu \mathrm{~s}$ | Module keeps in <br> "operational" state; <br> the outputs are acti- <br> vated trough the user <br> program |
| $17 \ldots 32$ | $17 \ldots 32$ | $50 \mu \mathrm{~s}$ | Module keeps in <br> "operational" state; |
| 128 | 0 | $80 \mu \mathrm{~s}$ | the cam switch out- <br> puts are activated <br> according to an inter- <br> polated timing infor- <br> mation |
| $129 \ldots 136$ | $1 \ldots 8$ | $200 \mu \mathrm{~s}$ | s |
| $137 \ldots 144$ | $9 \ldots 16$ | $17 \ldots 32$ |  |
| $145 \ldots 170$ |  |  |  |

${ }^{2}$ ) The parameter resolution defines the angle resolution of the track. The value gives the number of increments related to $360^{\circ}$; e. g. the value 36,000 corresponds to an angle resolution of $0.01^{\circ}$.
${ }^{3}$ ) The parameter zeroShift defines the zero shift. With it the encoder can be adjusted to the mounting position. The value of zeroShift is set in encoder-increments. It is not assigned to the parameter resolution of the cam switch.
${ }^{4}$ ) The parameter EncoderBitResolution defines the resolution of the used encoder (in bits), e. g. with the default setting 18 bits the encoder has 196,608 divisions.

Channel parameters for the cam switch (max. 32x)

| Name | Value | Internal value | Internal value, <br> type | Default |
| :--- | :--- | :--- | :--- | :--- |
| camToTrack0 ${ }^{1}$ ) | Digital Output <br> $0 \ldots 15$, none | $0 \ldots 15$, FF | BYTE | FF |
| $:$ | $:$ | $:$ | $:$ | $:$ |
| camToTrack31 | Digital Output <br> $0 \ldots 15$, none | $0 \ldots 15$, FF | BYTE | FF |

${ }^{1}$ ) The value of the parameter camToTrack\# defines which DO (digital output) is assigned to the track. camToTrack0 $=3$ for example means that track 0 is assigned to the digital output 3 . If the value FFh is set to a track, no digital output is assigned to it.

| Name | Value | Referred FB from extended Cam Switch Library ${ }^{2}$ ) | Internal value | Internal value, type |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| cam- <br> Type[0] <br> ${ }^{1}$ ) <br> ... | Common <br> Pulsed <br> Timed <br> Comfort <br> Cam shift <br> Binary shift <br> Multiturn cam <br> Time timed <br> Reference <br> Multiturn timed | MCX_CamSwitchSimple_c <br> MCX_CamSwitchSimple_dc <br> MCX_PulseSwitch_dc <br> MCX_CamSwitchTimed_dc <br> MCX_CamSwitchComfort_dc <br> MCX_CamShift_dc <br> MCX_BinaryShift_dc <br> MCX_CamSwitchMulti_dc <br> MCX_SwitchTimeTimed_dc <br> MCX_BinaryReference_dc <br> MCX_CamSwitchMulti- <br> Timed dc | $\begin{array}{\|l} \hline 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 4 \\ 5 \\ 6 \\ 7 \\ 7 \\ 8 \\ 9 \end{array}$ | BYTE | 0 |

${ }^{1}$ ) camType additionally to camToTrack identifies the type of each cam switch and enables the use of a specific function block from the Extended Cam Switch Library.
${ }^{2}$ ) camType parameters and the Extended Camswitch Library are only available for $\mathrm{CI} 511-$ ETHCAT and CI512-ETHCAT with device index C0 and above.

## Group parameters for the digital part

| Name | Value | Internal value | Internal value, <br> type | Default |
| :--- | :--- | :--- | :--- | :--- |
| Input delay | 0.01 ms | 0 | BYTE | 0.01 ms |
|  | 1 ms | 1 | 2 | $0 x 00$ |
|  | 8 ms | 3 | ms | 0 |


| Name | Value | Internal value | Internal value, type | Default |
| :---: | :---: | :---: | :---: | :---: |
| Behaviour DO at comm. error *) | Off <br> Last value <br> Last value 5 sec <br> Last value 10 sec <br> Substitute value <br> Substitute value <br> 5 sec <br> Substitute value 10 sec | $\begin{aligned} & 0 \\ & 0 \\ & 1 \\ & 6 \\ & 11 \\ & 2 \\ & 2 \\ & 7 \\ & 12 \end{aligned}$ | BYTE | $\begin{array}{\|l\|} \hline \text { Off } \\ 0 \times 00 \end{array}$ |
| Substitute values DO | 0 ... 65535 | 0000h ... FFFFh | WORD | $\begin{aligned} & 0 \\ & 0 \times 0000 \end{aligned}$ |
| *) The parameter behaviourDOatCommunicationFault is only analyzed if the Failsafe-mode is ON. |  |  |  |  |

## Diagnosis

In cases of short circuit or overload, the digital outputs are turned off. The modules performs reactivation automatically. Thus an acknowledgement of the errors is not necessary. The error message is stored via the LED.

| E1 ... E4 | d1 | d2 | d3 | d4 | Identifier $000 \text {... } 063$ | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err |  |  |
| Byte 6 <br> Bit 6 ... 7 | - | Byte 3 | Byte 4 | Byte 5 | Byte 6 <br> Bit 0 ... 5 |  |  |
| Class | Interface | Device | Module | Channel | Error identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) |  |  |  |  |
| Module error |  |  |  |  |  |  |  |
| 3 | - | 31 | 31 | 31 | 43 | Internal error in the module | Replace I/O module |
| 3 | - | 31 | 31 | 31 | 20 | Slave-to-Slave malfunction | Check configuration |
| 3 | - | 31 | 31 | 31 | 41 | Distributed Clock malfunction | Check configuration |
| 3 | - | 31 | 31 | 31 | 26 | Parameter error | Check master |
| 3 | - | 31 | 31 | 31 | 11 | Process voltage UP too low | Check process supply voltage |


| E1 ... E4 | d1 | d2 | d3 | d4 | $\begin{aligned} & \text { Identifier } \\ & 000 \ldots 063 \end{aligned}$ | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err |  |  |
| Byte 6 <br> Bit 6 ... 7 | - | Byte 3 | Byte 4 | Byte 5 | Byte 6 <br> Bit 0 ... 5 |  |  |
| Class | Interface | Device | Module | Channel | Error identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | $\left.{ }^{3}\right)$ |  |  |  |  |
| 4 | - | 31 | 31 | 31 | 45 | Process voltage UP3 too low | Check process voltage |
| 4 | - | 31 | 31 | 31 | 34 | No response during initialization of the I/O module | Replace I/O module |
| 4 | - | 31 | 31 | 31 | 46 | Voltage feedback on activated digital outputs ${ }^{4}$ ) | Check terminals |
| Channel error digital |  |  |  |  |  |  |  |
| 4 | - | 31 | 2 | $0 \ldots 15$ | 46 | Voltage feedback on deactivated digital output <br> ${ }^{5}$ ) | Check terminals |
| 4 | - | 31 | 4 | $0 \ldots 7$ | 47 | Short circuit at digital output | Check terminals |
| 4 | - | 31 | 2 | $8 \ldots 15$ | 47 | Short circuit at digital output | Check terminals |

Remarks:

| ${ }^{1}$ ) | In AC500 the following interface identifier applies: <br> "-" = Diagnosis via bus-specific function blocks; 0 ... 4 or 10 = Position of the <br> Communication Module; 14 = I/O bus; 31 = Module itself <br> The identifier is not contained in the CI512-ETHCAT diagnosis block. |
| :--- | :--- |
| $\left.{ }^{2}\right)$ | With "Device" the following allocation applies: <br> $31=$ Module itself or ADR = Hardware address (e. g. of the DC551) |
| $\left.{ }^{3}\right)$ | With "Module" the following allocation applies dependent of the master: <br> $31=$ Module itself (Module error) or Module type (1=AI, 2=DO, 3=AO; channel <br> error) |
| $\left.{ }^{4}\right)$ | Diagnosis message appears for the whole output group and not per channel. <br> The message occurs if the output channel is already active. |
| 5 | Diagnosis message appears per channel. The message occurs if the output <br> channel is not active. |

## State LEDs

The LEDs are located at the front of module. There are 2 different groups:

- The 5 system LEDs (PWR, NET, DC, S-ERR and I/O-Bus) show the operation state of the module and display possible errors.
- The 29 process LEDs (UP, UP3, inputs, outputs, CH-ERR1 to CH-ERR3) show the process supply voltage and the states of the inputs and outputs and display possible errors.

Table 236: States of the 5 system LEDs

| LED | Color | Off | On | Flashing | 1x flash | 2x flash |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| PWR/RUN | Green | Error in the internal supply voltage or process voltage missing | Internal supply voltage OK | Module is not configured | -- | -- |
|  | Yellow | -- | -- | -- | -- | -- |
| NET | Green | Init | Operational | Pre-operational | Safe-operational | -- |
|  | Red | No error | PDI <br> Watchdog Timeout | Invalid Configuration | Unsolicited State Change | Application time out |
| DC *) | Green | Distributed Clock not active | Distributed Clock active | -- | -- | -- |
|  | Red | -- | -- | -- | -- | -- |
| S-ERR | Red | No error | Internal error | -- | -- | -- |
| I/O-Bus | Green | No communication interface modules connected or communication error | --- | --- | -- | -- |
| ETH1 | Green | No EtherCAT connection | Link OK No data transfer | Link OK <br> Data transfer OK | -- | -- |
|  | Yellow | -- | -- | -- | -- | -- |
| ETH2 | Green | No EtherCAT connection | Link OK No data transfer | Link OK <br> Data transfer OK | -- | -- |
|  | Yellow | -- | -- | -- | -- | -- |
| ${ }^{*}$ ) The state of this LED is only significant if the camswitch functionality is enabled |  |  |  |  |  |  |

Table 237: States of the 29 process LEDs

| LED | Color | OFF | ON | Flashing |
| :--- | :--- | :--- | :--- | :--- |
| DC0 ... DC7 | Yellow | Input/Output is <br> OFF | Input/Output is <br> ON | -- |
| DI8 ... DI15 | Yellow | Input is OFF | Input is ON (the <br> input voltage is <br> even displayed if <br> the supply <br> voltage is OFF) | -- <br> DO8 ... DO15 Yellow |
| UP | Green | Output is OFF <br> voltage missing | Output is ON | Process supply <br> voltage OK and <br> initialization fin- <br> ished |
| UP3 | Green | Process supply <br> voltage missing | Process supply <br> voltage OK | -- |
| CH-ERR1 to CH- <br> ERR3 | Red | No error or <br> process supply <br> voltage missing | Internal error | Error on one <br> channel of the <br> corresponding <br> group |

## Technical data

## Technical data of the module

The system data of AC500 and S500 are applicable to the standard version ${ }^{\mu}$ Chapter 5.1.2 "System data AC500" on page 166.
The system data of AC500-XC are applicable to the XC version ${ }^{\Perp}$ Chapter 5.1.3 "System data AC500-XC" on page 169.

Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter | Value |
| :--- | :--- |
| Bus connection | 2 x RJ45 |
| Technology | Hilscher NETX 100 |
| Transfer rate | $10 / 100$ Mbit/s (full-duplex) |
| Transfer method | According to Ethernet II, IEEE 802.3 |
| Ethernet | 100 base-TX, internal switch, 2x RJ45 socket |
| Expandability (S500 I/O modules) | Up to 10 S500 I/O modules (Index C0 and <br> above), not available (Index below C0) |
| Indicators | 5 LEDs for state indication |
| Adjusting elements | 2 rotary switches (used for future topology <br> extensions) |
| Quantity of input/output data | CI512-ETHCAT: 10 bytes input and 14 bytes <br> output <br> CI511-ETHCAT: 18 bytes input and 18 bytes <br> output |
| Limit of data for input and output | 144 byte |


| Parameter | Value |
| :---: | :---: |
| Acyclic services | SDO (1500 bytes max.) <br> Emergency ECAT_SLV_DIAG |
| Protective functions (according to CODESYS) | Protected against: <br> - short circuit <br> - reverse supply <br> - overvoltage <br> - reverse polarity <br> Galvanic isolation to network |


| Parameter | Value |
| :---: | :---: |
| Process supply voltages UP/UP3 |  |
| Rated value | 24 V DC (for inputs and outputs) |
| Max. load for the terminals | 10 A |
| Protection against reversed voltage | Yes |
| Rated protection fuse on UP/UP3 | 10 A fast |
| Galvanic isolation | Ethernet interface against the rest of the module |
| Inrush current from UP (at power up) | On request |
| Current consumption via UP (normal operation) | 0.15 A |
| Current consumption via UP3 | 0.06 A + 0.5 A max. per output |
| Connections | Terminals 1.8 and 2.8 for +24 V (UP) <br> Terminal 3.8 for +24 V (UP3) <br> Terminals 1.9, 2.9 and 3.9 for 0 V (ZP) |
| Max. power dissipation within the module | 6 W |
| Number of digital inputs | 8 |
| Number of digital outputs | 8 |
| Number of configurable digital inputs/outputs | 8 |
| Reference potential for all digital inputs and outputs | Negative pole of the supply voltage, signal name ZP |
| Diagnosis | See Diagnosis and Displays $\stackrel{y}{\Rightarrow}$ Chapter 5.2.6.4.2.10 "Diagnosis" on page 1060 |
| Operation and error displays | 34 LEDs (totally) |
| Weight (without terminal unit) | Ca. 125 g |
| Mounting position | Horizontal <br> Or vertical with derating (output load reduced to $50 \%$ at $+40^{\circ} \mathrm{C}$ per group) |
| Cooling | The natural convection cooling must not be hindered by cable ducts or other parts in the control cabinet. |

## NOTICE!

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and temporary overvoltage up to 30 V DC.

## Multiple overloads

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an internal smart high-side switch.

## Technical data of the digital inputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 8 |
| Distribution of the channels into groups | 1 group of 8 channels |
| Terminals of the channels DI0 ... DI7 | Terminals $2.0 \ldots 2.7$ |
| Reference potential for all inputs | Terminals $1.9 \ldots 3.9$ (negative pole of the <br> supply voltage, signal name ZP) |
| Indication of the input signals | 1 yellow LED per channel, the LED is ON when <br> the input signal is high (signal 1 ) |
| Input type (according EN 61131-2) | Type 1 |
| Input delay (0->1 or 1->0) | Typ. 0.1 ms, configurable from $0.1 \mathrm{~ms} \ldots 32 \mathrm{~ms}$ |
| Input signal voltage | 24 V DC |
|  | $0-$ Signal |
|  | $-3 \mathrm{~V} \ldots+5 \mathrm{~V}$ |
|  | 1-Signal |
| Ripple with signal 0 | $>+5 \mathrm{~V} \ldots<+15 \mathrm{~V}$ |
| Ripple with signal 1 | $+15 \mathrm{~V} \ldots+30 \mathrm{~V}$ |
| Input current per channel | Within $-3 \mathrm{~V} \ldots+5 \mathrm{~V}$ |
|  | Input voltage +24 V |
| Input voltage +5 V | Within $+15 \mathrm{~V} \ldots+30 \mathrm{~V}$ |
|  | Input voltage +15 V |
| Input voltage +30 V | $>1 \mathrm{~mA}$ |
| Max. cable length | $>2 \mathrm{~mA}$ |
|  | Shielded |
| Unshielded | $<8 \mathrm{~mA}$ |
|  | 600 m |

## Technical data of the digital outputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 8 |
| Distribution of the channels into groups | 1 group of 8 channels |


| Parameter | Value |
| :---: | :---: |
| Terminals of the channels DO0 ... DO7 | Terminals 3.0 ... 3.7 |
| Reference potential for all outputs | Terminals 1.9 ... 3.9 (negative pole of the supply voltage, signal name ZP) |
| Common power supply voltage | For all outputs terminal 3.8 (positive pole of the supply voltage, signal name UP3) |
| Output voltage for signal 1 | UP3 (-0.8 V) |
| Output delay (0->1 or 1->0) | On request |
| Output current |  |
| Rated value per channel | 500 mA at UP3 $=24 \mathrm{~V}$ |
| Max. value (all channels together) | 4 A |
| Leakage current with signal 0 | $<0.5 \mathrm{~mA}$ |
| Fuse for UP3 | 10 A fast |
| Demagnetization with inductive DC load | Via internal varistors (see figure below this table) |
| Output switching frequency |  |
| With resistive load | On request |
| With inductive loads | Max. 0.5 Hz |
| With lamp loads | 11 Hz max. at 5 W max. |
| Short-circuit-proof / overload-proof | Yes |
| Overload message ( $1>0.7 \mathrm{~A}$ ) | Yes, after ca. 100 ms |
| Output current limitation | Yes, automatic reactivation after short circuit/overload |
| Resistance to feedback against 24 V signals | Yes (software-controlled supervision) |
| Max. cable length |  |
| Shielded | 1000 m |
| Unshielded | 600 m |

The following drawing shows the circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off.


Fig. 204: Digital input/output (circuit diagram)
1 Digital output
2 Varistors for demagnetization when inductive loads are turned off
Figure:

## Technical data of the configurable digital inputs/outputs

Each of the configurable I/O channels is defined as input or output by the user program. This is done by interrogating or allocating the corresponding channel.

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 8 inputs/outputs (with transistors) |
| Distribution of the channels into groups | 1 group for 8 channels |
| If the channels are used as inputs |  |
|  | Channels DC0 $\ldots$ DC07 | Terminals $1.0 \ldots 1.7$.

## Technical data of the digital inputs/outputs if used as inputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 8 |
| Distribution of the channels into groups | 1 group of 8 channels |
| Terminals of the channels DC0 ... DC7 | Terminals 1.0 ... 1.7 |
| Reference potential for all inputs | Terminals 1.9 ... 3.9 (negative pole of the supply voltage, signal name ZP) |
| Indication of the input signals | 1 yellow LED per channel, the LED is ON when the input signal is high (signal 1 ) |
| Input type (according EN 61131-2) | Type 1 |
| Input delay (0->1 or 1->0) | Typ. 0.1 ms , configurable from $0.1 \mathrm{~ms} \ldots 32 \mathrm{~ms}$ |
| Input signal voltage | 24 V DC |
| 0-Signal | -3 V ... +5 V *) |
| Undefined Signal | > +5V ... $<+15 \mathrm{~V}$ |
| 1-Signal | +15V ... +30 V |
| Ripple with signal 0 | Within -3 V ... +5V *) |
| Ripple with signal 1 | Within +15 V ... +30 V |
| Input current per channel |  |
| Input voltage +24 V | Typ. 5 mA |
| Input voltage +5 V | $>1 \mathrm{~mA}$ |
| Input voltage +15 V | $>2 \mathrm{~mA}$ |
| Input voltage +30 V | $<8 \mathrm{~mA}$ |
| Max. cable length |  |
| Shielded | 1000 m |
| Unshielded | 600 m |

*) Due to the direct connection to the output, the demagnetizing varistor is also effective at the input (see figure) above. This is why the difference between UPx and the input signal may not exceed the clamp voltage of the varistor. The varistor limits the voltage to approx. 36 V . Following this, the input voltage must range from $-12 \mathrm{~V} \ldots+30 \mathrm{~V}$ when UPx $=24 \mathrm{~V}$ and from $-6 \vee \ldots+30 \mathrm{~V}$ when $\mathrm{UPx}=30 \mathrm{~V}$.

## Technical data of the digital inputs/outputs if used as outputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 8 |
| Distribution of the channels into groups | 1 group of 8 channels |
| Terminals of the channels DC0 ... DC7 | Terminals 1.0 ... 1.7 |
| Reference potential for all outputs | Terminals $1.9 \ldots 3.9$ (negative pole of the supply voltage, signal name ZP) |
| Common power supply voltage | For all outputs terminal 3.8 (positive pole of the supply voltage, signal name UP3) |
| Output voltage for signal 1 | UP3 (-0.8 V) |
| Output delay (0->1 or 1->0) | On request |
| Output current |  |
| Rated value per channel | 500 mA at UP3 $=24 \mathrm{~V}$ |
| Max. value (all channels together) | 4 A |
| Leakage current with signal 0 | $<0.5 \mathrm{~mA}$ |
| Fuse for UP3 | 10 A fast |
| Demagnetization with inductive DC load | Via internal varistors (see figure below this table) |
| Output switching frequency |  |
| With resistive load | On request |
| With inductive loads | Max. 0.5 Hz |
| With lamp loads | 11 Hz max. at 5 W max. |
| Short-circuit-proof / overload-proof | Yes |
| Overload message ( $1>0.7$ A) | Yes, after ca. 100 ms |
| Output current limitation | Yes, automatic reactivation after short circuit/overload |
| Resistance to feedback against 24 V signals | Yes (software-controlled supervision) |
| Max. cable length |  |
| Shielded | 1000 m |
| Unshielded | 600 m |

The following drawing shows the circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off.


Fig. 205: Digital input/output (circuit diagram)
1 Digital input/output
2 For demagnetization when inductive loads are turned off

## Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm


The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 221 000 R0001 | CIL12-ETHCAT, EtherCAT communi- <br> cation interface module, 8 DI, 8 DO <br> and 8 DC | Active |

> *) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.6.5 Modbus

### 5.2.6.5.1 Cl521-MODTCP

## Features

- 4 analog inputs (resolution 12 bits including sign)
- 2 analog outputs (resolution 12 bits including sign)
- 8 digital inputs 24 V DC
- 8 digital outputs 24 V DC, 0.5 A max.
- Module-wise galvanically isolated
- Fast counter
- XC version for usage in extreme ambient conditions available


1 I/O bus
2 Allocation between terminal number and signal name
36 yellow LEDs to display the signal states of the analog inputs/outputs (AIO ... AI3, AOO ... AO1)
48 yellow LEDs to display the signal states of the digital inputs (DIO ... DI7)
58 yellow LEDs to display the signal states of the digital outputs (DO0 ... DO7)
62 green LEDs to display the process supply voltage UP and UP3
73 red LEDs to display errors (CH-ERR1, CH-ERR2, CH-ERR3)
85 system LEDs: PWR/RUN, STA1 ETH, STA2 ETH, S-ERR, I/O-Bus
9 Label
102 rotary switches for setting the IP address
11 Ethernet interfaces (ETH1, ETH2) on the terminal unit
12 Terminal unit
13 DIN rail


## Intended purpose

The Modbus TCP communication interface module CI521-MODTCP is used as decentralized I/O module in Modbus TCP networks. The network connection is performed via 2 RJ45 connectors which are integrated in the terminal unit. The communication interface module contains 22 I/O channels with the following properties:

- 4 analog inputs (1.0 ... 1.3)
- 2 analog outputs (1.5 ... 1.6)
- 8 digital inputs 24 V DC in 1 group (2.0 ... 2.7)
- 8 digital outputs 24 V DC in 1 group (3.0 ... 3.7)

The inputs/outputs are galvanically isolated from the Ethernet network. There is no potential separation between the channels. The configuration of the analog inputs/outputs is performed by software.

For usage in enhanced ambient conditions (e.g. wider temperature and humidity range), a special XC version of the device is available.

## Functionality

| Parameter | Value |
| :--- | :--- |
| Interface | Ethernet |
| Protocol | Modbus TCP |
| Power supply | From the process supply voltage UP |
| Supply of the electronic circuitry of the I/O <br> expansion modules attached | Through the I/O bus interface (I/O bus) |
| Rotary switches | for setting the last BYTE of the IP (00h to FFh) |
| Analog inputs | 4 (configurable via software) |
| Analog outputs | 2 (configurable via software) |
| Digital inputs | 8 (24 V DC; delay time configurable via soft- <br> ware) |
| Digital outputs | 8 (24 V DC, 0.5 A max.) |
| LED displays | For system displays, signal states, errors and <br> power supply |
| External supply voltage | Via terminals ZP, UP and UP3 (process supply <br> voltage 24 V DC) |
| Required terminal unit | TU507 or TU508 \& Chapter 5.2.7.2 "TU507- <br> ETH and TU508-ETH for Ethernet communica- <br> tion interface modules" on page 1270 |

## Connections

## General

The Ethernet communication interface module CI521-MODTCP is plugged on the I/O terminal unit TU507-ETH or TU508-ETH $\Leftrightarrow$ Chapter 5.2.7.2 "TU507-ETH and TU508-ETH for Ethernet communication interface modules" on page 1270. Properly seat the module and press until it locks in place. The terminal unit is mounted on a DIN rail or with 2 screws plus the additional accessory for wall mounting ${ }^{\circ}$ Chapter 5.2.8.2.5 "TA526-Wall mounting accessory" on page 1324.

The connection of the I/O channels is carried out using the 30 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

The terminals 1.8 and 2.8 as well as $1.9,2.9$ and 3.9 are electrically interconnected within the terminal unit and have always the same assignment, independent of the inserted module:

Terminals 1.8 and 2.8: Process supply voltage UP $=+24 \mathrm{~V}$ DC
Terminal 3.8: Process supply voltage UP3 $=+24 \mathrm{~V}$ DC
Terminals 1.9, 2.9 and 3.9: Process supply voltage $\mathrm{ZP}=0 \mathrm{~V}$

With a separate UP3 power supply, the digital outputs can be switched off externally. This way, an emergency-off functionality can be realized.

## Conditions for undisturbed operating with older I/O expansion modules

All I/O expansion modules that are attached to the CI52x-MODTCP must be powered up together with the CI52x-MODTCP if the firmware version of these I/O expansion modules is V1.9 or lower.

The firmware version is related to the index. The index is printed on the module type label on the right side.

Modules as of index listed in the following table can be powered up independently.

| S500 I/O module type | First index with firmware version above 1.9 |
| :--- | :--- |
| Al523 | D0 |
| AI523-XC | D0 |
| AI531 | A3 |
| A1531-XC | A0 |
| AO523 | D0 |
| AO523-XC | D0 |
| AX521 | D0 |
| AX521-XC | D0 |
| AX522 | D0 |
| AX522-XC | D0 |
| CD522 | A2 |
| CD522-XC | A0 |
| DA501 | A2 |
| DA501-XC | A0 |
| DA502 | A1 |
| DA502-XC | A1 |
| DC522 | D0 |
| DC522-XC | D0 |
| DC523 | D0 |


| S500 I/O module type | First index with firmware version above $\mathbf{1 . 9}$ |
| :--- | :--- |
| DC523-XC | D0 |
| DC532 | D0 |
| DC532-XC | D0 |
| DI524 | D0 |
| DI524-XC | D0 |
| DO524 | A2 |
| DO524-XC | A2 |
| DX522 | D0 |
| DX522-XC | D0 |
| DX531 | D0 |
| AC522 | D0 |
| PD501 | D0 |

Do not connect any voltages externally to digital outputs!
Reason: Externally voltages at an output or several outputs may cause that other outputs are supplied through that voltage instead of voltage UP3 (reverse voltage). This ist not intended usage.

## CAUTION!

Risk of malfunction by unintended usage!
If the function cut-off of the digital outputs is to be used by deactivation of the supply voltage UP3, be sure that no external voltage is conncted at the outputs DOO..DO7.

Table 238: Assignment of the other terminals

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 1.0 | AI0+ | Positive pole of analog input signal 0 |
| 1.1 | Al1+ | Positive pole of analog input signal 1 |
| 1.2 | Al2+ | Positive pole of analog input signal 2 |
| 1.3 | Al3+ | Positive pole of analog input signal 3 |
| 1.4 | AO0+ | Negative pole of analog input signals 0 to 3 |
| 1.5 | AO1+ | Positive pole of analog output signal 0 |
| 1.6 | AI- | Positive pole of analog output signal 1 |
| 1.7 | ZP | Negative pole of analog output signals 0 and 1 |
| 1.8 | DI0 | Process voltage UP (24 V DC) |
| 1.9 | DI1 | Process voltage ZP (0 V DC) |
| 2.0 | DI2 | Signal of the digital input DI0 |
| 2.1 | DI3 | Signal of the digital input DI1 |
| 2.2 | DI4 | Signal of the digital input DI2 |
| 2.3 |  | Signal of the digital input DI4 |
| 2.4 |  |  |


| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 2.5 | DI5 | Signal of the digital input DI5 |
| 2.6 | DI6 | Signal of the digital input DI6 |
| 2.7 | DI7 | Signal of the digital input DI7 |
| 2.8 | UP | Process voltage UP (24 V DC) |
| 2.9 | ZP | Process voltage ZP (0 V DC) |
| 3.0 | DO0 | Signal of the digital output DO0 |
| 3.1 | DO2 | Signal of the digital output DO1 |
| 3.2 | DO3 | Signal of the digital output DO2 |
| 3.3 | SO5 | Signal of the digital output DO4 |
| 3.4 | DO6 | Signal of the digital output DO5 |
| 3.5 | DO7 | Signal of the digital output DO6 |
| 3.6 | UP3 | Signal of the digital output DO7 |
| 3.7 | ZP | Process voltage UP3 (24 V DC) |
| 3.8 | Process voltage ZP (0 V DC) |  |
| 3.9 |  |  |

## WARNING!

## Removal/Insertion under power

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.
Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.

Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.
The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

## Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

For the open-circuit detection (wire break), each analog input channel is pulled up to "plus" by a high-resistance resistor. If nothing is connected, the maximum voltage will be read in then.

Generally, analog signals must be laid in shielded cables. The cable shields must be grounded at both sides of the cables. In order to avoid unacceptable potential differences between different parts of the installation, low resistance equipotential bonding conductors must be laid.


Fig. 206: Connection of the communication interface module CI521-MODTCP * Chapter 6.8.4.1 "Modbus communication interface module" on page 4568

## Connection of the digital inputs



Fig. 207: Connection of the digital inputs (DOO ... DO7) to the module CI521-MODTCP The meaning of the LEDs is described in Displays ${ }^{\circ}$ Chapter 5.2.6.5.1.10 "State LEDs" on page 1101.

## Connection of the digital outputs



Fig. 208: Connection of configurable digital inputs/outputs (DOO ... DO7) to the module CI521MODTCP

The meaning of the LEDs is described in Displays ${ }^{\circ}$ Chapter 5.2.6.5.1.10 "State LEDs" on page 1101.

## Connection of resistance thermometers in 2-wire configuration to the analog inputs

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the module CI521MODTCP provides a constant current source which is multiplexed over the max. 4 analog input channels.


Fig. 209: Connection of resistance thermometers in 2-wire configuration to the analog inputs (AIO ... Al3)
The following measuring ranges can be configured ${ }^{\wedge} \Rightarrow$ Chapter 5.2.6.5.1.8 "Parameterization" on page $1090 \Leftrightarrow$ Chapter 5.2.6.5.1.11 "Measuring ranges" on page 1102:

| Pt100 | $-50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ | 2-wire configuration, 1 <br> channel used |
| :--- | :--- | :--- |
| Pt100 | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 2-wire configuration, 1 <br> channel used |
| Pt1000 | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 2-wire configuration, 1 <br> channel used |
| Ni1000 | $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ | 2-wire configuration, 1 <br> channel used |

The function of the LEDs is described under Diagnosis and displays / Displays ${ }_{\wedge}{ }^{\mu}$ Chapter 5.2.6.5.1.9 "Diagnosis" on page 1096.

The module CI521-MODTCP performs a linearization of the resistance characteristic.
To avoid error messages from unused analog input channels, configure them as "unused".

## Connection of resistance thermometers in 3-wire configuration to the analog inputs

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the module $\mathrm{Cl} 521-$ MODTCP provides a constant current source which is multiplexed over the max. 4 analog input channels.


Fig. 210: Connection of resistance thermometers in 3-wire configuration to the analog inputs (AIO ... Al3)
With 3-wire configuration, 2 adjacent analog channels belong together (e. g. the channels 0 and 1). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0), the next higher address must be the odd address (channel 1).

The constant current of one channel flows through the resistance thermometer. The constant current of the other channel flows through one of the cores. The module calculates the measured value from the two voltage drops and stores it under the input with the higher channel number (e. g. I1).
In order to keep measuring errors as small as possible, it is necessary to have all the involved conductors in the same cable. All the conductors must have the same cross section.

The following measuring ranges can be configured ${ }^{\geqslant}$Chapter 5.2.6.5.1.8 "Parameterization" on page 1090 and $\xi^{\xi}$ Chapter 5.2.6.5.1.11 "Measuring ranges" on page 1102:

| Pt100 | $-50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ | 3-wire configuration, 2 chan- <br> nels used |
| :--- | :--- | :--- |
| Pt100 | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 3-wire configuration, 2 chan- <br> nels used |
| Pt1000 | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 3-wire configuration, 2 chan- <br> nels used |
| Ni1000 | $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ | 3-wire configuration, 2 chan- <br> nels used |

The function of the LEDs is described under Diagnosis and displays / Displays ${ }_{幺}{ }^{\circ}$ Chapter 5.2.6.5.1.9 "Diagnosis" on page 1096.

The module CI521-MODTCP performs a linearization of the resistance characteristic.
To avoid error messages from unused analog input channels, configure them as "unused".

## Connection of active-type analog sensors (voltage) with galvanically isolated power supply to the analog inputs



Fig. 211: Connection of active-type analog sensors (voltage) with galvanically isolated power supply to the analog inputs (AIO ... AI3)
The following measuring ranges can be configured ${ }^{\wedge} \Rightarrow$ Chapter 5.2.6.5.1.8 "Parameterization" on page $1090 \stackrel{ }{\mu} \stackrel{y}{c}$ Chapter 5.2.6.5.1.11 "Measuring ranges" on page 1102:

| Voltage | $0 \ldots 10 \mathrm{~V}$ | 1 channel used |
| :--- | :--- | :--- |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | 1 channel used |

The function of the LEDs is described under Diagnosis and displays / Displays ${ }_{\nu} \Rightarrow$ Chapter 5.2.6.5.1.9 "Diagnosis" on page 1096.

To avoid error messages from unused analog input channels, configure them as "unused".

Connection of active-type analog sensors (Current) with galvanically isolated power supply to the analog inputs


Fig. 212: Connection of active-type analog sensors (current) with galvanically isolated power supply to the analog inputs (AIO ... AI3)
The following measuring ranges can be configured $\left.{ }^{\psi}\right\rangle$ Chapter 5.2.6.5.1.8 "Parameterization" on page $1090 \Leftrightarrow$ Chapter 5.2.6.5.1.11 "Measuring ranges" on page 1102:

| Current | $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 1 channel used |
| :--- | :--- | :--- |
| Current | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 1 channel used |

The function of the LEDs is described under Diagnosis and displays / Displays $\Leftrightarrow$ Chapter 5.2.6.5.1.9 "Diagnosis" on page 1096.

Unused input channels can be left open-circuited, because they are of low resistance.
To avoid error messages through unused analog input channels in measuring range $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$, these channels should be configured as "Not used".

## Connection of active-type analog sensors (Voltage) with no galvanically isolated power supply to the analog inputs



Fig. 213: Connection of active-type sensors (voltage) with no galvanically isolated power supply to the analog inputs (AIO ... AI3)

## CAUTION!

## Risk of faulty measurements!

The negative pole at the sensors must not have too big a potential difference with respect to ZP (max. $\pm 1 \mathrm{~V}$ ).

Make sure that the potential difference never exceeds $\pm 1 \mathrm{~V}$ (also not with long cable lengths).

The following measuring ranges can be configured ${ }^{\mu} \Rightarrow$ Chapter 5.2.6.5.1.8 "Parameterization" on page $1090 \Leftrightarrow$ Chapter 5.2.6.5.1.11 "Measuring ranges" on page 1102.

| Voltage | $0 \ldots 10 \mathrm{~V}$ | 1 channel used |
| :--- | :--- | :--- |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | 1 channel used |

The function of the LEDs is described under Diagnosis and displays / Displays ${ }_{幺}$ Chapter 5.2.6.5.1.9 "Diagnosis" on page 1096.

To avoid error messages from unused analog input channels, configure them as "unused".

## Connection of passive-type analog sensors (Current) to the analog inputs

The following figure shows the connection of passive-type analog sensors (current) to the analog input AIO. Proceed with the analog inputs AI1 ... AI3 in the same way.


Fig. 214: Connection of passive-type analog sensors (current) to the analog inputs (AIO ... AI3)
The following measuring ranges can be configured ${ }^{\geqslant}$Chapter 5.2.6.5.1.8 "Parameterization" on page 1090 出 Chapter 5.2.6.5.1.11 "Measuring ranges" on page 1102:

| Current | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 1 channel used |
| :--- | :--- | :--- |

The function of the LEDs is described under Diagnosis and displays / Displays ${ }^{\mu} \boldsymbol{y}$ Chapter 5.2.6.5.1.9 "Diagnosis" on page 1096.

## CAUTION!

## Risk of overloading the analog input!

If an analog current sensor supplies more than 25 mA for more than 1 second during initialization, this input is switched off by the module (input protection).

Use only sensors with fast initialization or without current peaks higher than 25 mA . If not possible, connect a 10 -volt Zener diode in parallel to Alx+ and ZP.

Unused input channels can be left open-circuited, because they are of low resistance.
To avoid error messages through unused analog input channels in measuring range $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$, these channels should be configured as "Not used".

## Connection of active-type analog sensors (Voltage) to differential analog inputs

Differential inputs are very useful, if analog sensors are used which are remotely non-isolated (e.g. the minus terminal is remotely grounded).

The evaluation using differential inputs helps to considerably increase the measuring accuracy and to avoid ground loops.

With differential input configurations, two adjacent analog channels belong together (e.g. the channels 0 and 1). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0), the next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1).
The analog value is calculated by subtraction of the input value with the higher address from the input value of the lower address.

The converted analog value is available at the odd channel (higher address).

## CAUTION!

## Risk of faulty measurements!

The negative pole at the sensors must not have too big a potential difference with respect to ZP (max. $\pm 1 \mathrm{~V}$ ).

Make sure that the potential difference never exceeds $\pm 1 \mathrm{~V}$.

The following figure shows the connection of active-type analog sensors (voltage) to differential analog inputs AIO and AI1. Proceed with AI2 and AI3 in the same way.


Fig. 215: Connection of active-type analog sensors (voltage) to differential analog inputs (AIO ... Al3)

The following measuring ranges can be configured $\stackrel{y}{ }{ }^{\circ}$ Chapter 5.2.6.5.1.8 "Parameterization" on page $1090 \Leftrightarrow$ Chapter 5.2.6.5.1.11 "Measuring ranges" on page 1102:

| Voltage | $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ | With differential inputs, 2 <br> channels used |
| :--- | :--- | :--- |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | With differential inputs, 2 <br> channels used |

The function of the LEDs is described under Diagnosis and displays / Displays ${ }^{\mu} \Rightarrow$ Chapter 5.2.6.5.1.9 "Diagnosis" on page 1096.

To avoid error messages from unused analog input channels, configure them as "unused".

## Use of analog inputs as digital inputs

Several (or all) analog inputs can be configured as digital inputs ${ }_{y}{ }^{\circ}$ Chapter $5 \cdot 2 \cdot 6 \cdot 5 \cdot 1 \cdot 12.5$ "Technical data of the analog inputs if used as digital inputs" on page 1108. The inputs are not galvanically isolated against the other analog channels.


Fig. 216: Connection of digital sensors to the analog inputs (AIO ... AI3)
The following measuring ranges can be configured ${ }_{y}{ }^{4}$ Chapter 5.2.6.5.1.8 "Parameterization" on page 1090 and $\left.{ }^{\mu}\right\rangle$ Chapter 5.2.6.5.1.11 "Measuring ranges" on page 1102 :

| Digital input | 24 V | 1 channel used |
| :--- | :--- | :--- |

The function of the LEDs is described under Diagnosis and displays / Displays ${ }^{\mu}$ Chapter 5.2.6.5.1.9 "Diagnosis" on page 1096.

## Connection of analog output loads (Voltage)



Fig. 217: Connection of analog output loads (voltage) to the analog outputs (AO0 ... AO1)
The following measuring ranges can be configured ${ }^{\mu} \Rightarrow$ Chapter 5.2.6.5.1.8 "Parameterization" on page $1090 \Leftrightarrow$ Chapter 5.2.6.5.1.11 "Measuring ranges" on page 1102

| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | Load $\pm 10 \mathrm{~mA}$ max. | 1 channel used |
| :--- | :--- | :--- | :--- |

The function of the LEDs is described under Diagnosis and displays / Displays ${ }_{\circ} \Rightarrow$ Chapter 5.2.6.5.1.9 "Diagnosis" on page 1096.

Unused analog outputs can be left open-circuited.

## Connection of analog output loads (Current)



Fig. 218: Connection of analog output loads (current) to the analog outputs (AOO and AO1)
The following measuring ranges can be configured Chapter 5.2.6.5.1.8 "Parameterization" on page 1090 \& Chapter 5.2.6.5.1.11 "Measuring ranges" on page 1102:

| Current | $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | Load $0 \ldots 500 \Omega$ | 1 channel used |
| :--- | :--- | :--- | :--- |
| Current | $4 \ldots 20 \mathrm{~mA}$ | Load $0 \ldots 500 \Omega$ | 1 channel used |

The function of the LEDs is described under Diagnosis and displays / Displays ${ }_{\wedge} \Rightarrow$ Chapter 5.2.6.5.1.9 "Diagnosis" on page 1096.

Unused analog outputs can be left open-circuited.

## Assignment of the Ethernet ports

The terminal unit for the communication interface module provides two Ethernet interfaces with the following pin assignment:

Pin assignment

| Interface | Pin | Signal | Description |
| :---: | :---: | :---: | :---: |
|  | 1 | TxD+ | Transmit data + |
|  | 2 | TxD- | Transmit data - |
|  | 3 | RxD+ | Receive data + |
|  | 4 | NC | Not connected |
|  | 5 | NC | Not connected |
|  | 6 | RxD- | Receive data - |


| Interface | Pin | Signal | Description |
| :--- | :--- | :--- | :--- |
|  | 7 | NC | Not connected |
|  | 8 | NC | Not connected |
|  | Shield | Cable shield | Functional earth |

In corrosive environment, please protect unused connectors using the TA535 accessory.

Not supplied with this device.

③ Further information about wiring and cable types

## Internal data exchange

| Parameter | Value |
| :--- | :--- |
| Digital inputs (bytes) | 3 |
| Digital outputs (bytes) | 3 |
| Analog inputs (words) | 4 |
| Analog outputs (words) | 2 |
| Counter input data (words) | 4 |
| Counter output data (words) | 8 |

Replacement of a Modbus TCP communication interface module:
The module must be powered off before it is replaced. If the configuration data is stored in the module, then the configuration data must be downloaded into the new module, either by using Modbus communication or by using the Modbus configurator which is contained in the Automation Builder distribution.
ⓨ Chapter 6.3.2.12.2 "Unbundled CI52x-MODTCP configuration" on page 1565

## Addressing

The module reads the position of the rotary switches only during power-up, i.e. changes of the switch position during operation will have no effect until the next module initialization.

The IP address of the CI521-MODTCP Module can be set with the "ABB IP Configuration Tool" $\Leftrightarrow$ Chapter 6.3.2.2.4.3 "Configuration of the IP settings with the IP configuration tool" on page 1457.
If the last byte of the IP is set to 0 , the address switch will be used instead.
Address switch position 255 is mapped to fixed IP 192.168.0.254 independent of other stored settings. This is a backup so the module can always get a valid IP address and can be configured by the "ABB IP Configuration Tool".

Address switch position 0 is mapped to last byte equal 1 and DHCP enabled.

The factory setting for the IP is 192.168.0.x (last byte is address switch).

I/O configuration
The Cl521-MODTCP stores configuration parameters (IP address configuration, module parameters).
The analog/digital I/O channels are configured via software.
Details about configuration are described in Parameterization * Chapter 5.2.6.5.1.8 "Parameterization" on page 1090.

## Parameterization

## Parameters of the module

| Name | Value | Internal value | Internal value, type | Default |
| :---: | :---: | :---: | :---: | :---: |
| Module ID ${ }^{1}$ ) | Internal | 7400 | WORD | 7000 |
| Ignore Module | Internal | 0 | BYTE | 0 |
| Parameter length | Internal | 63 | BYTE | 63 |
| Error LED / Failsafe function see table Error LED / Failsafe function * Table 239 "Err or LED / Failsafe function" on page 1091 | On | 0 | BYTE | 0 |
|  | Off by E4 | 1 |  |  |
|  | Off by E3 | 3 |  |  |
|  | On + failsafe | 16 |  |  |
|  | $\begin{aligned} & \text { Off by E4 + fail- } \\ & \text { safe } \end{aligned}$ | 17 |  |  |
|  | Off by E3 + failsafe | 19 |  |  |
| Master IP for Write restriction ${ }^{4}$ ) | No master IP Master IP | 0,0,0,0 W, X,y,z | $\begin{aligned} & \text { ARRAY[0.3] OF } \\ & \text { BYTE } \end{aligned}$ | 0,0,0,0 |
| Master IP for Write restriction ${ }^{4}$ ) | No master IP Master IP | 0,0,0,0 W, X,y,z | $\begin{aligned} & \text { ARRAY[0..3] OF } \\ & \text { BYTE } \end{aligned}$ | 0,0,0,0 |
| Master IP for Write restriction ${ }^{4}$ ) | No master IP Master IP | 0,0,0,0 W, X,y,z | $\begin{aligned} & \text { ARRAY[0.3] OF } \\ & \text { BYTE } \end{aligned}$ | 0,0,0,0 |
| Master IP for Write restriction ${ }^{4}$ ) | No master IP Master IP | 0,0,0,0 W, X,y,z | $\begin{aligned} & \text { ARRAY[0..3] OF } \\ & \text { BYTE } \end{aligned}$ | 0,0,0,0 |
| Master IP for Write restriction ${ }^{4}$ ) | No master IP Master IP | 0,0,0,0 W, X,y,z | $\begin{aligned} & \text { ARRAY[0..3] OF } \\ & \text { BYTE } \end{aligned}$ | 0,0,0,0 |
| Master IP for Write restriction ${ }^{4}$ ) | No master IP Master IP | 0,0,0,0 W, X,y,z | $\begin{aligned} & \text { ARRAY[0.3] OF } \\ & \text { BYTE } \end{aligned}$ | 0,0,0,0 |
| Master IP for Write restriction ${ }^{4}$ ) | No master IP Master IP | 0,0,0,0 W, X,y,z | $\begin{aligned} & \text { ARRAY[0..3] OF } \\ & \text { BYTE } \end{aligned}$ | 0,0,0,0 |


| Name | Value | Internal value | Internal value, type | Default |
| :---: | :---: | :---: | :---: | :---: |
| Master IP for Write restriction ${ }^{4}$ ) | No master IP Master IP | 0,0,0,0 W, X, y,z | ARRAY[0..3] OF BYTE | 0,0,0,0 |
| Timeout for Bus supervision | No supervision 10 ms timeout 20 ms timeout | $\begin{aligned} & 0 \\ & 1 \\ & 2 \end{aligned}$ | BYTE | No supervision |
| IO Mapping Structure ${ }^{3}$ ) | Fixed Mapping Dynamic Mapping | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | BYTE | 0 |
| Reserved | Internal | 0 | ARRAY[0..2] OF BYTE | 0,0,0 |
| Check supply | off on | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | BYTE | 1 |
| Fast counter | $0$ $\left.10^{3}\right)$ | 0 $:$ 10 | BYTE | 0 |
| ${ }^{1}$ ) With a faulty ID, the Modules reports a "parameter error" and does not perform cyclic process data transmission. |  |  |  |  |
| ${ }^{2}$ ) Counter operating modes $\stackrel{\xi}{ }{ }^{\circ}$ Chapter 6.8.2.12 "Fast counters in AC500 devices" on page 4536. |  |  |  |  |
| ${ }^{3}$ ) Fixed Mapping means each module has its own Modbus registers for data transfer independent of the IO bus constellation $\stackrel{\text { H Chapter 6.8.4.1.2 "Modbus TCP registers" }}{ }$ on page 4570. |  |  |  |  |
| Dynamic mapping means the structure of the IO Date is dependent on the I/O bus constellation. Each I/O bus expansion module starts directly after the module before on the next Word address. |  |  |  |  |
| ${ }^{4}$ ) If none of the parameters is set all masters / clients in the network have read and write rights on the CI52x-MODTCP device and its connected expansion modules. |  |  |  |  |
| If at least one parameter is set only the configured masters / clients have write rights on the CI52x-MODTCP device, all other masters / clients still have read access to the CI52xMODTCP device. |  |  |  |  |

Table 239: Error LED / Failsafe function

| Setting | Description |
| :--- | :--- |
| On | Error LED (S-ERR) lights up at errors of all <br> error classes, Failsafe-mode off |
| Off by E4 | Error LED (S-ERR) lights up at errors of error <br> classes E1, E2 and E3, Failsafe-mode off |
| Off by E3 | Error LED (S-ERR) lights up at errors of error <br> classes E1 and E2, Failsafe-mode off |
| On +Failsafe | Error LED (S-ERR) lights up at errors of all <br> error classes, Failsafe-mode on *) |
| Off by E4 + Failsafe | Error LED (S-ERR) lights up at errors of error <br> classes E1, E2 and E3, Failsafe-mode on *) |


| Setting | Description |
| :--- | :--- |
| Off by E3 + Failsafe | Error LED (S-ERR) lights up at errors of error <br> classes E1 and E2, Failsafe-mode on *) |
| *) The parameters Behaviour AO at comm. error and Behaviour DO at comm. error are only <br> analyzed if the Failsafe-mode is ON. |  |

## Group parameters for the analog part

| Name | Value | Internal value | Internal value, type | Default |
| :---: | :---: | :---: | :---: | :---: |
| Analog data format | Standard Reserved | $\begin{aligned} & \hline 0 \\ & 255 \end{aligned}$ | BYTE | 0 |
| Behaviour AO at comm. error *) | Off <br> Last value <br> Last value 5 s <br> Last value 10 s <br> Substitute value <br> Substitute value <br> 5 s <br> Substitute value 10 s | $\begin{aligned} & \hline 0 \\ & 1 \\ & 6 \\ & 6 \\ & 11 \\ & 2 \\ & 7 \\ & 12 \end{aligned}$ | BYTE | 0 |

Channel parameters for the analog inputs (4x)

| Name | Value | Internal value | Internal value, type | Default |
| :---: | :---: | :---: | :---: | :---: |
| Input 0, Channel configuration | Table Operating modes of the analog inputs Table 240 "Ch annel configuration" <br> on page 1093 | Table Operating modes of the analog inputs Table 240 "Ch annel configuration" <br> on page 1093 | BYTE | 0 |
| Input 0, Check channel | Table Channel montoring ② Table 241 "Ch annel monitoring" on page 1093 | Table Channel montoring ② Table 241 "Ch annel monitoring" on page 1093 | BYTE | 0 |
| : | : | : | : | : |
| : | : | : | : | : |


| Name | Value | Internal value | Internal value, type | Default |
| :---: | :---: | :---: | :---: | :---: |
| Input 3, Channel configuration | Table Operating modes of the analog inputs <br> Table 240 "Ch annel configuration" <br> on page 1093 | Table Operating modes of the analog inputs Table 240 "Ch annel configuration" on page 1093 | BYTE | 0 |
| Input 3, Check channel | Table Channel montoring ② Table 241 "Ch annel monitoring" on page 1093 | Table Channel montoring (4) Table 241 "Ch annel monitoring" on page 1093 | BYTE | 0 |

Table 240: Channel configuration

| Internal value | Operating modes of the analog inputs, individually configurable |
| :---: | :---: |
| 0 (default) | Not used |
| 1 | 0 ... 10 V |
| 2 | Digital input |
| 3 | 0 ... 20 mA |
| 4 | $4 \ldots 20 \mathrm{~mA}$ |
| 5 | -10 V ... +10 V |
| 8 | 2-wire Pt100-50 ... $+400^{\circ} \mathrm{C}$ |
| 9 | 3-wire Pt100-50 ... $+400^{\circ} \mathrm{C}$ *) |
| 10 | 0 ... 10 V (voltage diff.) *) |
| 11 | -10 V ... +10 V (voltage diff.) *) |
| 14 | 2-wire Pt100-50 ... $+70^{\circ} \mathrm{C}$ |
| 15 | 3 -wire Pt100 -50 ... $+70^{\circ} \mathrm{C}$ *) |
| 16 | 2-wire Pt1000-50 ... $+400{ }^{\circ} \mathrm{C}$ |
| 17 | 3 -wire Pt1000-50 ... $+400{ }^{\circ} \mathrm{C}$ *) |
| 18 | 2-wire Ni1000-50 ... $+150^{\circ} \mathrm{C}$ |
| 19 | 3-wire Ni1000-50 ... $+150^{\circ} \mathrm{C}$ *) |
| *) In the operating modes with 3 -wire configuration or with differential inputs, two adjacent analog inputs belong together (e.g. the channels 0 and 1 ). In these cases, both channels are configured in the desired operating mode. The lower address must be the even address (channel 0 ). The next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1). |  |

Table 241: Channel monitoring

| Internal Value | Check Channel |
| :--- | :--- |
| 0 (default) | Plausibility, wire break, short circuit |
| 3 | Not used |

## Channel parameters for the analog outputs (2x)

| Name | Value | Internal value | Internal value, type | Default |
| :---: | :---: | :---: | :---: | :---: |
| Output 0, <br> Channel configuration | Table Operating modes of the analog outputs ② Table 242 "Ch annel configuration" on page 1094 | Table Operating modes of the analog outputs ③ Table 242 "Ch annel configuration" <br> on page 1094 | BYTE | 0 |
| Output 0, Check channel | Table Channel monitoring <br> « Table 243 "Ch annel monitoring" on page 1094 | Table Channel monitoring <br> « Table 243 "Ch annel monitoring" on page 1094 | BYTE | 0 |
| Output 0, Substitute value | Table Substitute value (3) Table 244 "Su bstitute value" on page 1095 | Table Substitute value « Table 244 "Su bstitute value" on page 1095 | WORD | 0 |
| Output 1, <br> Channel configuration | Table Operating modes of the analog outputs を Table 242 "Ch annel configuration" <br> on page 1094 | Table Operating modes of the analog outputs Table 242 "Ch annel configuration" <br> on page 1094 | BYTE | 0 |
| Output 1, Check channel | Table Channel monitoring <br> ヶ Table 243 "Ch annel monitoring" on page 1094 | Table Channel monitoring <br> « Table 243 "Ch annel monitoring" on page 1094 | BYTE | 0 |
| Output 1, Substitute value | Table Substitute value <br> ๕ Table 244 "Su bstitute value" on page 1095 | Table Substitute value <br> ๕ Table 244 "Su bstitute value" on page 1095 | WORD | 0 |

Table 242: Channel configuration

| Internal value | Operating modes of the analog outputs, individually configu- <br> rable |
| :--- | :--- |
| 0 (default) | Not used |
| 128 | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ |
| 129 | $0 \ldots 20 \mathrm{~mA}$ |
| 130 | $4 \ldots 20 \mathrm{~mA}$ |

Table 243: Channel monitoring

| Internal value | Check channel |
| :--- | :--- |
| 0 | Plausibility, wire break, short circuit |
| 3 | None |

Table 244: Substitute value

| Intended behavior of output <br> channel when the control <br> system stops | Required setting of the <br> module parameter "Behav- <br> iour of outputs in case of a <br> communication error" | Required setting of the <br> channel parameter "Substi- <br> tute value" |
| :--- | :--- | :--- |
| Output OFF | Off | 0 |
| Last value infinite | Last value | 0 |
| Last value for 5 s and then <br> turn off | Last value 5 sec | 0 |
| Last value for 10 s and then <br> turn off | Last value 10 sec | 0 |
| Substitute value infinite | Substitute value | Depending on configuration |
| Substitute value for 5 s and <br> then turn off | Substitute value 5 sec | Depending on configuration |
| Substitute value for 10 s and <br> then turn off | Substitute value 10 sec | Depending on configuration |

## Group parameters for the digital part

| Name | Value | Internal value | Internal value, type | Default |
| :---: | :---: | :---: | :---: | :---: |
| Input delay | $\begin{aligned} & \hline 0.1 \mathrm{~ms} \\ & 1 \mathrm{~ms} \\ & 8 \mathrm{~ms} \\ & 32 \mathrm{~ms} \end{aligned}$ | $\begin{array}{\|l\|} \hline 0 \\ 1 \\ 2 \\ 3 \end{array}$ | BYTE | $\begin{aligned} & 0.1 \mathrm{~ms} \\ & 0 \times 00 \end{aligned}$ |
| Detect short circuit at outputs | $\begin{aligned} & \text { Off } \\ & \text { On } \end{aligned}$ | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | BYTE | $\begin{aligned} & \text { On } \\ & 0 \times 01 \end{aligned}$ |
| Behaviour DO at comm. error ${ }^{1}$ ) | Off <br> Last value <br> Last value 5 sec <br> Last value 10 sec <br> Substitute value <br> Substitute value 5 sec <br> Substitute value 10 sec | $\begin{aligned} & \hline 0 \\ & 1 \\ & 6 \\ & 6 \\ & 11 \\ & 2 \\ & 7 \\ & 12 \end{aligned}$ | BYTE | $\begin{aligned} & \text { Off } \\ & 0 \times 00 \end{aligned}$ |
| Substitute value at output | 0 ... 255 | 00h ... FFh | BYTE | $\begin{aligned} & 0 \\ & 0 \times 0000 \end{aligned}$ |
| Detect voltage overflow at outputs ${ }^{2}$ ) | $\begin{aligned} & \text { Off } \\ & \text { On } \end{aligned}$ | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | BYTE | $\begin{aligned} & \mathrm{On} \\ & 0 \times 01 \end{aligned}$ |
| ${ }^{1}$ ) The parameters Behaviour DO at comm. error is only analyzed if the Failsafe-mode is ON. <br> ${ }^{2}$ ) The state "externally voltage detected" appears, if the output of a channel DC0 ... DC7 should be switched on while an externally voltage is connected ${ }^{\mu}$ Chapter 5.2.6.5.1.4 "Connections" on page 1072. In this case the start up is disabled, as long as the externally voltage is connected. The monitoring of this state and the resulting diagnosis message can be disabled by setting the parameters to "OFF". |  |  |  |  |

## Diagnosis

Table 245: Structure of the diagnosis block

| Byte Number | Description | Possible Values |
| :---: | :---: | :---: |
| 1 | Diagnosis Byte, slot number | $31=$ CI521-MODTCP (e. g. error at integrated 8 DI / 8 DO) <br> 1 = 1st connected S500 I/O Module <br> ... <br> $10=10$ th connected S500 I/O Module |
| 2 | Diagnosis Byte, module number | According to the I/O bus specification passed on by modules to the fieldbus master |
| 3 | Diagnosis Byte, channel | According to the I/O bus specification passed on by modules to the fieldbus master |
| 4 | Diagnosis Byte, error code | According to the I/O bus specification Bit 7 and bit 6, coded error class $\begin{aligned} & 0=\mathrm{E} 1 \\ & 1=\mathrm{E} 2 \\ & 2=\mathrm{E} 3 \\ & 3=\mathrm{E} 4 \end{aligned}$ <br> Bit 0 to bit 5, coded error description |
| 5 | Diagnosis Byte, flags | According to the I/O bus specification <br> Bit 7: 1 = coming error <br> Bit 6: 1 = leaving error |

In cases of short circuit or overload, the digital outputs are turned off. The modules performs reactivation automatically. Thus an acknowledgement of the errors is not necessary. The error message is stored via the LED.

For diagnosis firmware version $\geq 3.2 .6$ is required.

| E1 ... E4 | d1 | d2 | d3 | d4 | Identifier $000 \ldots 0$ | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err |  |  |
| $\begin{array}{\|l} \hline \text { Byte } 4 \\ \text { Bit } \\ 6 \text {... } 7 \end{array}$ | - | Byte 1 | Byte 2 | Byte 3 | Byte 4 Bit 0 ... 5 |  |  |
| Class | Interface | Device | Module | Channel | Error-Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) |  |  |  |  |
| Module errors |  |  |  |  |  |  |  |
| 3 | - | 31 | 31 | 31 | 19 | Checksum error in the I/O module | Replace I/O module |
| 3 | - | 31 | 31 | 31 | 3 | Timeout in the I/O module |  |
| 3 | - | 31 | 31 | 31 | 40 | Different hard-/firmware versions in the module |  |
| 3 | - | 31 | 31 | 31 | 43 | Internal error in the module |  |
| 3 | - | 31 | 31 | 31 | 36 | Internal data exchange failure |  |
| 3 | - | 31 | 31 | 31 | 9 | Overflow diagnosis buffer | Restart |
| 3 | - | 31 | 31 | 31 | 26 | Parameter error | Check Master |
| 3 | - | 31 | 31 | 31 | 11 | Process voltage UP too low | Check process supply voltage |
| 3 | - | 31 | 31 | 31 | 45 | No process voltage UP | Check process supply voltage |
| 3 | - | 31/1... 10 | 31 | 31 | 17 | No communication with I/O module | Replace I/O module |
| 3 | - | 1... 10 | 31 | 31 | 32 | Wrong I/O module type on socket | Replace I/O module / Check configuration |
| 4 | - | 1... 10 | 31 | 31 | 31 | At least one module does not support failsafe function | Check modules and parameterization |


| E1 ... E4 | d1 | d2 | d3 | d4 | Identifier $000 \ldots$ | AC500Display | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err | $\begin{aligned} & \text { PS501 } \\ & \text { PLC } \\ & \text { Browser } \end{aligned}$ |  |  |
| $\begin{array}{\|l\|} \hline \text { Byte } 4 \\ \text { Bit } \\ 6 \text {... } 7 \end{array}$ | - | Byte 1 | Byte 2 | Byte 3 | $\begin{aligned} & \text { Byte } 4 \\ & \text { Bit } \\ & 0 \text {... } 5 \end{aligned}$ | PNIO diagnosis block |  |  |
| Class | Interface | Device | Module | Channel | Error- <br> Identifier | Error message |  | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) |  |  |  |  |  |
| 4 | - | 1... 10 | 31 | 5 | 8 | I/O module removed from hot swap terminal unit or defective module on hot swap terminal unit ${ }^{9}$ ) |  | Plug I/O module, replace I/O module |
| 4 | - | 1... 10 | 31 | 5 | 28 | Wrong I/O module plugged on hot swap terminal unit ${ }^{9}$ ) |  | Remove wrong I/O module and plug projected I/O module |
| 4 | - | 1... 10 | 31 | 5 | 42 | No communication with I/O module on hot swap terminal unit ${ }^{9}$ ) |  | Replace I/O module |
| 4 | - | 1... 10 | 31 | 5 | 54 | I/O module does not support hot swap $\left.{ }^{8}\right)^{9}$ ) |  | Power off system and replace I/O module |
| 4 | - | 1... 10 | 31 | 6 | 8 | Hot swap terminal unit configured but not found |  | Replace terminal unit by hot swap terminal unit |
| 4 | - | 1... 10 | 31 | 6 | 42 | No communication with hot swap terminal unit ${ }^{9}$ ) |  | Restart, if error persists replace terminal unit |
| 4 | - | 31 | 31 | 31 | 46 | Voltage feedback on activated digital outputs DO0...DO7 on UP3 ${ }^{4}$ ) |  | Check terminals |


| E1 ... E4 | d1 | d2 | d3 | d4 | Identifier 000 ... | $\left\|\begin{array}{l\|l}\text { AC500- } \\ \text { Display }\end{array}\right\|<-$ Displa | <- Display in |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err | PS501 <br> PLC <br> Browser |  |
|  | - | Byte 1 | Byte 2 | Byte 3 | $\begin{aligned} & \hline \text { Byte } 4 \\ & \text { Bit } \\ & 0 \text {... } 5 \end{aligned}$ | PNIO diagnosis block |  |
| Class | $\begin{aligned} & \text { Inter- } \\ & \text { face } \end{aligned}$ | Device | Module | Channel | Error- <br> Identi- <br> fier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) |  |  |  |  |
| 4 | - | 31/1... 10 | 31 | 31 | 34 | No response during initialization of the I/O module | Replace I/O module |
| 4 | - | 31 | 31 | 31 | 11 | Process voltage UP3 too low | Check process supply voltage |
| 4 | - | 31 | 31 | 31 | 45 | No process voltage UP3 | Check process supply voltage |
| 4 | - | 31 | 31 | 31 | 10 | Voltage overflow on outputs (above UP3 level) ${ }^{5}$ ) | Check terminals/ check process supply voltage |
| Channel error digital |  |  |  |  |  |  |  |
| 4 | - | 31 | 2 | 0... 7 | 46 | Externally voltage detected at digital output DO0...DO7 ${ }^{6}$ ) | Check terminals |
| 4 | - | 31 | 2 | 0... 7 | 47 | Short circuit at digital output ${ }^{7}$ ) | Check terminals |
| Channel error analog |  |  |  |  |  |  |  |
| 4 | - | 31 | 1 | $0 . .3$ | 48 | Analog value overflow or broken wire at an analog input | Check value or check terminals |
| 4 | - | 31 | 1 | $0 . .3$ | 7 | Analog value underflow at an analog input | Check value |
| 4 | - | 31 | 1 | $0 . .3$ | 47 | Short circuit at an analog input | Check terminals |


| E1 ... E4 | d1 | d2 | d3 | d4 | Identifier $000 \ldots$ | AC500- <br> Display$\|<-$ Display | <- Display in |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err | $\begin{aligned} & \text { PS501 } \\ & \text { PLC } \\ & \text { Browser } \end{aligned}$ |  |
| $\begin{array}{\|l} \hline \text { Byte } 4 \\ \text { Bit } \\ 6 \text {... } 7 \end{array}$ | - | Byte 1 | Byte 2 | Byte 3 | $\begin{aligned} & \text { Byte } 4 \\ & \text { Bit } \\ & 0 \text {... } 5 \end{aligned}$ | PNIO diagnosis block |  |
| Class | Interface | Device | Module | Channel | Error- <br> Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) |  |  |  |  |
| 4 | - | 31 | 3 | $0 . .1$ | 4 | Analog value overflow at an analog output | Check output value |
| 4 | - | 31 | 3 | $0 . .1$ | 7 | Analog value underflow at an analog output | Check output value |

Remarks:

| ${ }^{1}$ ) | In AC500 the following interface identifier applies: <br> "-" = Diagnosis via bus-specific function blocks; 0 ... 4 or $10=$ Position of the Communication Module;14 = I/O bus; $31=$ Module itself <br> The identifier is not contained in the CI521-MODTCP diagnosis block. |
| :---: | :---: |
| ${ }^{2}$ ) | With "Device" the following allocation applies: $31=$ Module itself; $1 . .10=$ Expansion module |
| ${ }^{3}$ ) | With "Module" the following allocation applies: 31 = Module itself <br> Module type (1 = AI, 2 = DO, 3 = AO) |
| ${ }^{4}$ ) | This message appears, if externally voltages at one or more terminals DO0...DO7 cause that other digital outputs are supplied through that voltage © Chapter 5.2.6.5.1.4 "Connections" on page 1072. All outputs of the apply digital output groups will be turned off for 5 seconds. The diagnosis message appears for the whole output group. |
| ${ }^{5}$ ) | The voltage on digital outputs DO0...DO7 has overrun the process supply voltage UP3 « Chapter 5.2.6.5.1.4 "Connections" on page 1072. Diagnosis message appears for the whole module. |
| ${ }^{6}$ ) | This message appears, if the output of a channel DOO...DO7 should be switched on while an externally voltage is connected. In this case the start up is disabled, as long as the externally voltage is connected. Otherwise this could produce reverse voltage from this output to other digital outputs. This diagnosis message appears per channel. |
| ${ }^{7}$ ) | Short circuit: After a detected short circuit, the output is deactivated for 100 ms . Then a new start up will be executed. This diagnosis message appears per channel. |


| ${ }^{8}$ ) | In case of an I/O module doesn't support hot swapping, do not perform any <br> hot swap operations (also not on any other terminal units (slots)) as modules <br> may be damaged or I/O bus communication may be disturbed. |
| :--- | :--- |
| ${ }^{9}$ ) | Diagnosis for hot swap available as of version index FO. |

## State LEDs

The LEDs are located at the front of module. There are 2 different groups:

- The 5 system LEDs (PWR, STA1 ETH, STA2 ETH, S-ERR and I/O-Bus) show the operation state of the module and display possible errors.
- The 27 process LEDs (UP, UP3, inputs, outputs, CH-ERR1 to CH-ERR3) show the process supply voltage and the states of the inputs and outputs and display possible errors.

Table 246: States of the 5 system LEDs

| LED | Color | OFF | ON | Flashing |
| :---: | :---: | :---: | :---: | :---: |
| PWR/RUN | Green | Process supply voltage missing | Internal supply voltage OK, module ready for communication with IO Controller | Start-up / preparing communication |
|  | Yellow | --- | --- | --- |
| STA1 ETH <br> (System LED "BF") | Green | --- | Device configured, cyclic data exchange running | Device configured, acyclic data exchange running |
|  | Red | -- | Communication error (timeout) appeared | IP address error |
| STA2 ETH(System LED"SF") | Green | Device has valid parameters | Device is running parameterization sequenze | Device has no parameters |
|  | Red | --- | --- | Device has invalid parameters |
| S-ERR | Red | No error | Internal error | -- |
| I/O-Bus | Green | No expansion modules connected or communication error | Expansion modules connected and operational | --- |
| ETH1 | Green | No connection at Ethernet interface | Connected to Ethernet interface | --- |
|  | Yellow | --- | Device is transmitting telegrams | Device is transmitting telegrams |
| ETH2 | Green | No connection at Ethernet interface | Connected to Ethernet interface | --- |
|  | Yellow | --- | Device is transmitting telegrams | Device is transmitting telegrams |

Table 247: States of the 27 process LEDs

| LED | Color | OFF | ON | Flashing |
| :--- | :--- | :--- | :--- | :--- |
| AIO ... AI3 | Yellow | Input is OFF | Input is ON <br> (brightness <br> depends on the <br> value of the <br> analog signal) | -- |
| AO0 ... AO1 | Yellow | Output is OFF | Output is ON <br> (brightness <br> depends on the <br> value of the <br> analog signal) | -- |
| DI0 ... DI7 | Yellow | Input is OFF | Input is ON (the <br> input voltage is <br> even displayed if <br> the supply <br> voltage is OFF) | -- |
| DO0 ... DO7 | Yellow | Green | Output is OFF <br> Process supply <br> voltage missing | Output is ON <br> Poltage OK and <br> initialization fin- <br> ished |
| UP | Green | Process supply <br> voltage missing | Process supply <br> voltage OK | --- |
| UP3 | No error or <br> process supply <br> voltage missing | Internal error <br> ERRR3 | Error on one <br> channel of the <br> corresponding <br> group |  |

## Measuring ranges

Input ranges voltage, current and digital input

| Range | $\begin{aligned} & \mathrm{O} \text { V } \ldots+10 \\ & \mathrm{~V} \end{aligned}$ | $\begin{aligned} & -10 \mathrm{~V} \ldots \\ & +10 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 0 \mathrm{~mA} . . \\ & 20 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 4 \mathrm{~mA} \ldots 20 \\ & \mathrm{~mA} \end{aligned}$ | Digital input | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  | Decimal | Hex. |
| Overflow | > 11.7589 | > 11.7589 | > 23.5178 | > 22.8142 |  | 32767 | 7FFF |
| Measured value too high |  |  | 23.5178 $:$ 20.0007 |  |  | $\begin{aligned} & 32511 \\ & \vdots \\ & 27649 \end{aligned}$ | $\begin{aligned} & \text { 7EFF } \\ & : \\ & 6 \mathrm{C} 01 \end{aligned}$ |
| Normal range <br> Normal range or measured value too low | 10.0000 $:$ 0.0004 | $\begin{aligned} & \hline 10.0000 \\ & : \\ & 0.0004 \\ & \hline \end{aligned}$ | 20.0000 $:$ 0.0007 | 20.0000 $:$ 4.0006 | On | 27648 | $\begin{array}{\|l} \hline 6 \mathrm{C} 00 \\ : \\ 0001 \\ \hline \end{array}$ |
|  | 0.0000 | 0.0000 | 0 | 4 | Off | 0 | 0000 |
|  | $\begin{array}{\|l\|} \hline-0.0004 \\ -1.7593 \end{array}$ | $\begin{aligned} & -0.0004 \\ & : \\ & : \\ & -10,0000 \end{aligned}$ |  | $\begin{aligned} & 3.9994 \\ & 1.1858 \end{aligned}$ |  | -1 <br> -4864 -27648 | $\begin{aligned} & \text { FFFF } \\ & \text { ED00 } \\ & : \\ & 9400 \end{aligned}$ |


| Range | 0 V ... +10 | -10 V ... |  | $4 \mathrm{~mA} . .220$ | Digital | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  | Decimal | Hex. |
| Measured value too low |  | $\begin{aligned} & -10.0004 \\ & : \\ & -11.7589 \end{aligned}$ |  |  |  | $\begin{aligned} & -27649 \\ & : \\ & -32512 \end{aligned}$ | $\begin{aligned} & \text { 93FF } \\ & : \\ & 8100 \end{aligned}$ |
| Underflow | < 1.7593 | <-11.7589 | $<0.0000$ | < 1.1858 |  | -32768 | 8000 |

The represented resolution corresponds to 16 bits.

Input ranges resistance temperature detector

| Range | Pt100/ <br> Pt1000 $-50^{\circ} \mathrm{C} \ldots+$ | $\begin{aligned} & \text { Pt100 / Pt1000 } \\ & -50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \mathrm{Ni} 1000 \\ & -50^{\circ} \mathrm{C} \ldots+150 \end{aligned}$ | Digital value ${ }^{\circ} \mathrm{C}$ |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  | Decimal | Hex. |
| Overflow | > $+80.0{ }^{\circ} \mathrm{C}$ | $>+450.0^{\circ} \mathrm{C}$ | $>+160.0{ }^{\circ} \mathrm{C}$ | 32767 | 7FFF |
| Measured value too high |  | $\begin{aligned} & +450.0^{\circ} \mathrm{C} \\ & \text { } \\ & +400.1^{\circ} \mathrm{C} \end{aligned}$ |  | $\begin{aligned} & 4500 \\ & : \\ & 4001 \end{aligned}$ | $1194$ <br> 0FA1 |
|  |  |  | $\begin{aligned} & +160.0^{\circ} \mathrm{C} \\ & + \\ & +150.1^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & 1600 \\ & : \\ & 1501 \end{aligned}$ | $\begin{aligned} & 0640 \\ & : \\ & \text { 05DD } \end{aligned}$ |
|  | $+80.0^{\circ} \mathrm{C}$ $+70.1^{\circ} \mathrm{C}$ |  |  | $\begin{aligned} & 800 \\ & : \\ & 701 \end{aligned}$ | $\begin{aligned} & 0320 \\ & : \\ & \text { 02BD } \end{aligned}$ |
| Normal range | $\begin{aligned} & +70.0^{\circ} \mathrm{C} \\ & = \\ & +0.1^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & +400.0^{\circ} \mathrm{C} \\ & : \\ & : \\ & : \\ & +0.1^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & +150.0^{\circ} \mathrm{C} \\ & : \\ & : \\ & +0.1^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & 4000 \\ & 1500 \\ & 700 \\ & : \\ & 1 \end{aligned}$ | $\begin{aligned} & \text { 0FAO } \\ & \text { 05DC } \\ & \text { 02BC } \\ & : \\ & 0001 \end{aligned}$ |
|  | $0.0{ }^{\circ} \mathrm{C}$ | $0.0{ }^{\circ} \mathrm{C}$ | $0.0{ }^{\circ} \mathrm{C}$ | 0 | 0000 |
| Normal range | $\begin{aligned} & -0.1^{\circ} \mathrm{C} \\ & : \\ & -50.0^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & -0.1^{\circ} \mathrm{C} \\ & : \\ & -50.0^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & -0.1^{\circ} \mathrm{C} \\ & : \\ & -50,0^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & -1 \\ & : \\ & -500 \end{aligned}$ | $\begin{aligned} & \text { FFFF } \\ & : \\ & \text { FE0C } \end{aligned}$ |
| Measured value too low | $\begin{aligned} & -50.1^{\circ} \mathrm{C} \\ & : \\ & -60.0^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & -50.1^{\circ} \mathrm{C} \\ & : \\ & -60.0^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & -50.1^{\circ} \mathrm{C} \\ & : \\ & -60.0^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & -501 \\ & : \\ & -600 \end{aligned}$ | $\begin{aligned} & \text { FE0B } \\ & : \\ & \text { FDA8 } \end{aligned}$ |
| Underflow | $<-60.0^{\circ} \mathrm{C}$ | $<-60.0{ }^{\circ} \mathrm{C}$ | $<-60.0^{\circ} \mathrm{C}$ | -32768 | 8000 |

## Output ranges voltage and current

| Range | -10 V ... +10 V | $0 \mathrm{~mA} . . .20 \mathrm{~mA}$ | $4 \mathrm{~mA} . . .20 \mathrm{~mA}$ | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  | Decimal | Hex. |
| Overflow | 0 V | 0 mA | 0 mA | > 32511 | > 7EFF |
| Measured value too high | $\begin{aligned} & 11.7589 \mathrm{~V} \\ & : \\ & 10.0004 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 23.5178 \mathrm{~mA} \\ & : \\ & 20.0007 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 22.8142 \mathrm{~mA} \\ & : \\ & 20.0006 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 32511 \\ & : \\ & 27649 \end{aligned}$ | $\begin{aligned} & \text { 7EFF } \\ & : \\ & 6 \mathrm{C} 01 \end{aligned}$ |
| Normal range | $\begin{aligned} & 10.0000 \mathrm{~V} \\ & : \\ & 0.0004 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 20.0000 \mathrm{~mA} \\ & : \\ & 0,0007 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 20.0000 \mathrm{~mA} \\ & : \\ & 4.0006 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 27648 \\ & : \\ & 1 \end{aligned}$ | $\begin{aligned} & 6 \mathrm{C} 00 \\ & : \\ & 0001 \end{aligned}$ |
|  | 0.0000 V | 0.0000 mA | 4.0000 mA | 0 | 0000 |
|  | $\begin{aligned} & -0.0004 \mathrm{~V} \\ & : \\ & -10.0000 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 0 \mathrm{~mA} \\ & : \\ & 0 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 3.9994 \mathrm{~mA} \\ & 0 \mathrm{~mA} \\ & 0 \mathrm{~mA} \end{aligned}$ | $\begin{array}{\|l} \hline-1 \\ -6912 \\ -27648 \end{array}$ | $\begin{aligned} & \text { FFFF } \\ & \text { E500 } \\ & 9400 \end{aligned}$ |
| Measured value too low | $-10.0004 \mathrm{~V}$ -11.7589 V | $\begin{aligned} & 0 \mathrm{~mA} \\ & : \\ & 0 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 0 \mathrm{~mA} \\ & : \\ & 0 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & -27649 \\ & : \\ & -32512 \end{aligned}$ | $\begin{aligned} & 93 F F \\ & : \\ & 8100 \end{aligned}$ |
| Underflow | 0 V | 0 mA | 0 mA | <-32512 | < 8100 |

The represented resolution corresponds to 16 bits.

## Technical data

## Technical data of the module

The system data of AC500 and S500 are applicable to the standard version Chapter 5.1.2 "System data AC500" on page 166.
The system data of AC500-XC are applicable to the XC version $\&$ Chapter 5.1.3 "System data AC500-XC" on page 169.
Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter |  | Value |
| :--- | :--- | :--- |
| Process supply voltages UP/UP3 |  |  |
|  | Rated value | $24 \mathrm{~V} \mathrm{DC} \mathrm{(for} \mathrm{inputs} \mathrm{and} \mathrm{outputs)}$ |
|  | Max. load for the terminals | 10 A |
|  | Protection against reversed voltage | Yes |
|  | Rated protection fuse on UP/UP3 | 10 A fast |
|  | Galvanic isolation | Ethernet interface against the rest of the <br> module |
|  | Inrush current from UP (at power up) | On request |
|  | Current consumption via UP (normal <br> operation) | 0.2 A |
|  | Current consumption via UP3 | $0.06 \mathrm{~A}+0.5$ A max. per output |


| Parameter | Value |
| :---: | :---: |
| Connections | Terminals 1.8 and 2.8 for +24 V (UP) <br> Terminal 3.8 for +24 V (UP3) <br> Terminals 1.9, 2.9 and 3.9 for 0 V (ZP) |
| Max. power dissipation within the module | 6 W |
| Number of digital inputs | 8 |
| Number of digital outputs | 8 |
| Number of analog inputs | 4 |
| Number of analog outputs | 2 |
| Reference potential for all digital inputs and outputs | Negative pole of the supply voltage, signal name ZP |
| Ethernet | 10/100 base-TX, internal switch, $2 \times$ RJ45 socket |
| Setting of the IP address | With ABB IP config tool and 2 rotary switches at the front side of the module |
| Diagnose | See Diagnosis and Displays $\stackrel{\mu}{ }{ }^{\mu}$ Chapter 5.2.6.5.1.9 "Diagnosis" on page 1096 |
| Operation and error displays | 32 LEDs (totally) |
| Weight (without terminal unit) | Ca. 125 g |
| Mounting position | Horizontal or vertical with derating (output load reduced to $50 \%$ at $+40^{\circ} \mathrm{C}$ per group) |
| Extended ambient temperature (XC version) | > +60 ${ }^{\circ} \mathrm{C}$ on request |
| Cooling | The natural convection cooling must not be hindered by cable ducts or other parts in the control cabinet. |

0

## NOTICE!

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and temporary overvoltage up to 30 V DC.

## Technical data of the digital inputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 8 |
| Distribution of the channels into groups | 1 group of 8 channels |
| Terminals of the channels DI0 ... DI7 | Terminals $2.0 \ldots 2.7$ |
| Reference potential for all inputs | Terminals $1.9 \ldots 3.9$ (negative pole of the <br> supply voltage, signal name ZP) |
| Indication of the input signals | 1 yellow LED per channel, the LED is ON when <br> the input signal is high (signal 1$)$ |
| Input type (according EN 61131-2) | Type 1 |
| Input delay (0->1 or 1->0) | Typ. 0.1 ms, configurable from $0.1 \ldots 32 \mathrm{~ms}$ |
| Input signal voltage | 24 V DC |
|  | $0-$ Signal |


| Parameter |  | Value |
| :--- | :--- | :--- |
|  | Undefined Signal | $>+5 \mathrm{~V} \ldots<+15 \mathrm{~V}$ |
|  | 1-Signal | $+15 \mathrm{~V} \ldots+30 \mathrm{~V}$ |
| Ripple with signal 0 | Within $-3 \mathrm{~V} \ldots+5 \mathrm{~V}$ |  |
| Ripple with signal 1 | Within $+15 \mathrm{~V} \ldots+30 \mathrm{~V}$ |  |
| Input current per channel |  |  |
|  | Input voltage +24 V | Typ. 5 mA |
|  | Input voltage +5 V | $>1 \mathrm{~mA}$ |
|  | Input voltage +15 V | $>2 \mathrm{~mA}$ |
|  | Input voltage +30 V | $<8 \mathrm{~mA}$ |
| Max. cable length |  |  |
|  | Shielded | 1000 m |
|  | Unshielded | 600 m |

## Technical data of the digital outputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 8 |
| Distribution of the channels into groups | 1 group of 8 channels |
| Terminals of the channels DO0 ... DO7 | Terminals 3.0 ... 3.7 |
| Reference potential for all outputs | Terminals 1.9 ... 3.9 (negative pole of the supply voltage, signal name ZP) |
| Common power supply voltage | For all outputs terminal 3.8 (positive pole of the supply voltage, signal name UP3) |
| Output voltage for signal 1 | UP3 (-0.8 V) |
| Output delay (0->1 or 1->0) | On request |
| Output current |  |
| Rated value per channel | 500 mA at UP3 $=24 \mathrm{~V}$ |
| Max. value (all channels together) | 4 A |
| Leakage current with signal 0 | $<0.5 \mathrm{~mA}$ |
| Fuse for UP3 | 10 A fast |
| Demagnetization with inductive DC load | Via internal varistors (see figure below this table) |
| Output switching frequency |  |
| With resistive load | On request |
| With inductive loads | Max. 0.5 Hz |
| With lamp loads | 11 Hz max. at 5 W max. |
| Short-circuit-proof / overload-proof | Yes |
| Overload message ( $1>0.7$ A) | Yes, after ca. 100 ms |
| Output current limitation | Yes, automatic reactivation after short circuit/overload |
| Resistance to feedback against 24 V signals | Yes (software-controlled supervision) |


| Parameter | Value |
| :--- | :--- |
| Max. cable length |  |
|  | Shielded |
|  | Unshielded |

The following drawing shows the circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off.


Fig. 219: Digital input/output (circuit diagram)
1 Digital Output
2 Varistors for demagnetization when inductive loads are turned off

## Technical data of the analog inputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 4 |
| Distribution of channels into groups | 1 group with 4 channels |
| Connection if channels $\mathrm{AlO}+\ldots \mathrm{Al3+}$ | Terminals 1.0 ... 1.3 |
| Reference potential for $\mathrm{AlO}+\ldots \mathrm{Al3+}$ | Terminal 1.4 (AI-) for voltage and RTD measurement <br> Terminal 1.9, 2.9 and 3.9 for current measurement |
| Input type |  |
| Unipolar | Voltage 0 V ... 10 V , current or Pt100/Pt1000/ Ni1000 |
| Bipolar | Voltage -10 V ... +10 V |
| Galvanic isolation | Against Ethernet network |
| Configurability | $0 \vee \ldots 10 \mathrm{~V},-10 \mathrm{~V} \ldots+10 \mathrm{~V}, 0 / 4 \mathrm{~mA} . . .20 \mathrm{~mA}$, Pt100/1000, Ni1000 (each input can be configured individually) |
| Channel input resistance | Voltage: > $100 \mathrm{k} \Omega$ <br> Current: ca. $330 \Omega$ |
| Time constant of the input filter | Voltage: $100 \mu \mathrm{~s}$ Current: $100 \mu \mathrm{~s}$ |
| Indication of the input signals | 1 LED per channel (brightness depends on the value of the analog signal) |
| Conversion cycle | 1 ms (for 4 inputs +2 outputs); with RTDs Pt/Ni ... 1 s |


| Parameter | Value |
| :--- | :--- |
| Resolution | Range $0 \mathrm{~V} \ldots .10 \mathrm{~V}: 12$ bits <br> Range $-10 \mathrm{~V} \ldots+10 \mathrm{~V}: 12$ bits including sign <br> Range $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}: 12$ bits <br> Range $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}: 12$ bits <br> Range RTD (Pt100, PT1000, Ni1000): +0.1 ${ }^{\circ} \mathrm{C}$ |
| Conversion error of the analog values <br> caused by non-linearity, adjustment error <br> at factory and resolution within the normal <br> range | Typ. $0.5 \%$, max. $1 \%$ |
| Relationship between input signal and hex <br> code | «̌ Chapter 5.2.6.5.1.11.2 "Input ranges resist- <br> ance temperature detector" on page 1103 |
| Unused inputs | Are configured as "unused" (default value) |
| Overvoltage protection | Yes |

## Technical data of the analog inputs if used as digital inputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | Max. 4 |
| Distribution of channels into groups | 1 group of 4 channels |
| Connections of the channels $\mathrm{AlO+} . . \mathrm{Al3}+$ | Terminals $1.0 \ldots 1.3$ |
| Reference potential for the inputs | Terminals $1.9,2.9$ and 3.9 (ZP) |
| Indication of the input signals | 1 LED per channel |
| Input signal voltage | 24 V DC |
|  | Signal 0 |
|  | Undefined signal |
|  | Signal 1 |
| Input current per channel | $+5 \mathrm{~V} \ldots+13 \mathrm{~V}$ |
|  | Input voltage +24 V |
|  | Input voltage +5 V |
|  | Input voltage +15 V |
|  | Input voltage +30 V |
| Input resistance | Typ. 7 mA |

## Technical data of the analog outputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 2 |
| Distribution of channels into groups | 1 group for 2 channels |
| Connection of the channels $\mathrm{AO}+\ldots \mathrm{AO} 1+$ | Terminals $1.5 \ldots 1.6$ |
| Reference potential for $\mathrm{AO}+\ldots \mathrm{AO} 1+$ | Terminal 1.7 (AO-) for voltage outputTerminal |
|  | $1.9,2.9$ and 3.9 for current output |


| Parameter | Value |
| :---: | :---: |
| Output type |  |
| Unipolar | Current |
| Bipolar | Voltage |
| Galvanic isolation | Against internal supply and other modules |
| Configurability | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}, 0 \mathrm{~mA} . .220 \mathrm{~mA}, 4 \mathrm{~mA} . .20 \mathrm{~mA}$ (each output can be configured individually) |
| Output resistance (load), as current output | $0 \Omega \ldots 500 \Omega$ |
| Output loadability, as voltage output | $\pm 10 \mathrm{~mA}$ max. |
| Indication of the output signals | 1 LED per channel (brightness depends on the value of the analog signal) |
| Resolution | 12 bits including sign |
| Settling time for full range change (resistive load, output signal within specified tolerance) | Typ. 5 ms |
| Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range | Typ. 0.5 \%, max. 1 \% |
| Relationship between input signal and hex code | Table Output ranges voltage and current Chapter 5.2.6.5.1.11.3 "Output ranges voltage and current" on page 1104 |
| Unused outputs | Are configured as "unused" (default value) and can be left open-circuited |

Technical data of the fast counter

| Parameter | Value |
| :--- | :--- |
| Used inputs | Terminal 2.0 (DI0), 2.1 (DI1) |
| Used outputs | Terminal 3.0 (DO0) |
| Counting frequency | Depending on operation mode: |
|  | Mode 1 - 6: max. 200 kHz |
|  | Mode 7: max. 50 kHz |
|  | Mode 9: max. 35 kHz |
|  | Mode 10: max. 20 kHz |

(2) Chapter 6.8.2.12 "Fast counters in AC500 devices" on page 4536

## Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm

The dimensions are in mm and in brackets in inch.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 222 100 R0001 | Cl521-MODTCP, Modbus TCP com- <br> munication interface module, 4 AI, <br> 2 AO, 8 DI and 8 DO | Active |
| 1SAP 422 100 R0001 | Cl521-MODTCP-XC, Modbus TCP <br> communication interface module, 4 AI, <br> 2 AO, 8 DI and 8 DO, XC version | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.6.5.2 CI522-MODTCP

Features

- 8 digital inputs 24 V DC
- 8 digital outputs $24 \mathrm{~V} \mathrm{DC}, 0.5 \mathrm{~A}$ max.
- 8 configurable digital inputs/outputs $24 \mathrm{~V} D C, 0.5 \mathrm{~A}$ max.
- Module-wise galvanically isolated
- Fast counter
- XC version for usage in extreme ambient conditions available


1 I/O bus
2 Allocation between terminal number and signal name
38 yellow LEDs to display the signal states of the digital configurable inputs/outputs (DC0 ... DC7)
8 yellow LEDs to display the signal states of the digital inputs (DI8 ... DI15)
8 yellow LEDs to display the signal states of the digital outputs (DO8 ... DO15)
2 green LEDs to display the process supply voltage UP and UP3
3 red LEDs to display errors (CH-ERR1, CH-ERR2, CH-ERR3)
5 system LEDs: PWR/RUN, STA1 ETH, STA2 ETH, S-ERR, I/O-Bus
Label

102 rotary switches for setting the IP address
11 Ethernet interfaces (ETH1, ETH2) on the terminal unit
12 Terminal unit
13 DIN rail
${ }_{*}^{*}+{ }_{*}^{*}$ Sign for XC version

## Intended purpose

Modbus TCP communication interface module CI522-MODTCP is used as decentralized I/O module in Modbus TCP networks. The network connection is performed via 2 RJ45 connectors which are integrated in the terminal unit. The communication interface module contains 24 I/O channels with the following properties:

- 8 digital configurable inputs/outputs in 1 group (1.0 ... 1.7)
- 8 digital inputs 24 V DC in 1 group (2.0 ... 2.7)
- 8 digital outputs 24 V DC in 1 group (3.0 ... 3.7)

The inputs/outputs are galvanically isolated from the Ethernet network. There is no potential separation between the channels. The configuration of the configurable digital inputs/outputs is performed by software.

For usage in extreme ambient conditions (e.g. wider temperature and humidity range), a special $X C$ version of the device is available.

## Functionality

| Interface | Ethernet |
| :--- | :--- |
| Protocol | Modbus TCP |
| Power supply | From the process supply voltage UP |
| Supply of the electronic circuitry of the I/O <br> expansion modules attached | Through the I/O bus interface (I/O bus) |
| Rotary switches | for setting the last BYTE of the IP ADDRESS <br> $(00 h ~ . . . ~ F F h) ~$ |
| Configurable digital inputs/outputs | 8 (configurable via software) |
| Digital inputs | $8(24$ V DC; delay time configurable via soft- <br> ware) |
| Digital outputs | 8 (24 V DC, 0.5 A max.) |
| LED displays | For system displays, signal states, errors and <br> power supply |
| External supply voltage | Via terminals ZP, UP and UP3 (process supply <br> voltage 24 V DC) |
| Required terminal unit | TU507 or TU508 « Chapter 5.2.7.2 "TU507- <br> ETH and TU508-ETH for Ethernet communi- <br> cation interface modules" on page 1270 |

## Connections

## General

The Ethernet communication interface module CI522-MODTCP is plugged on the I/O terminal unit TU507-ETH $\Rightarrow$ Chapter 5.2.7.2 "TU507-ETH and TU508-ETH for Ethernet communication interface modules" on page 1270 or TU508-ETH $\Leftrightarrow$ Chapter 5.2.7.2 "TU507-ETH and TU508ETH for Ethernet communication interface modules" on page 1270. Properly seat the module and press until it locks in place. The terminal unit is mounted on a DIN rail or with 2 screws plus the additional accessory for wall mounting $\Leftrightarrow$ Chapter 5.2.8.2.5 "TA526-Wall mounting accessory" on page 1324.

The connection of the I/O channels is carried out using the 30 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

The terminals 1.8 and 2.8 as well as $1.9,2.9$ and 3.9 are electrically interconnected within the terminal unit and have always the same assignment, independent of the inserted module:

Terminals 1.8 and 2.8: Process supply voltage UP $=+24 \mathrm{~V}$ DC
Terminal 3.8: Process supply voltage UP3 $=+24 \mathrm{~V}$ DC
Terminals 1.9, 2.9 and 3.9: Process supply voltage $\mathrm{ZP}=0 \mathrm{~V}$

With a separate UP3 power supply, the digital outputs can be switched off externally. This way, an emergency-off functionality can be realized.

## Conditions for undisturbed operating with older I/O expansion modules

All I/O expansion modules that are attached to the CI52x-MODTCP must be powered up together with the CI52x-MODTCP if the firmware version of these I/O expansion modules is V1.9 or lower.

The firmware version is related to the index. The index is printed on the module type label on the right side.

Modules as of index listed in the following table can be powered up independently.

| S500 I/O module type | First index with firmware version above 1.9 |
| :--- | :--- |
| Al523 | D0 |
| Al523-XC | D0 |
| AI531 | A3 |
| AI531-XC | A0 |
| AO523 | D0 |
| AO523-XC | D0 |
| AX521 | D0 |
| AX521-XC | D0 |
| AX522 | D0 |
| AX522-XC | D0 |
| CD522 | A2 |


| S500 I/O module type | First index with firmware version above 1.9 |
| :--- | :--- |
| CD522-XC | A0 |
| DA501 | A2 |
| DA501-XC | A0 |
| DA502 | A1 |
| DA502-XC | A1 |
| DC522 | D0 |
| DC522-XC | D0 |
| DC523 | D0 |
| DC523-XC | D0 |
| DC532 | D0 |
| DC532-XC | D0 |
| D5524 | D0 |
| DI524-XC | D0 |
| DO524 | A2 |
| DO524-XC | A2 |
| DX522 | D0 |
| DX522-XC | D0 |
| DX531 | D0 |
| AC522 | D0 |
| PD501 | D0 |

Do not connect any voltages externally to digital outputs!
This ist not intended usage.
Reason: Externally voltages at one or more terminals DC0 ... DC7 or DO8 ... DO15 may cause that other digital outputs are supplied through that voltage instead of voltage UP3 (reverse voltage).
This is also possible, if DC channels are used as inputs. For this, the source for the input signals should be the impressed UP3 of the device.
This limitation does not apply for the input channels DIO ... DI7.

## CAUTION!

## Risk of malfunction by unintended usage!

If the function cut-off of the digital outputs is to be used by deactivation of the supply voltage UP3, be sure that no external voltage is conncted at the outputs DO8 ... DO15 and DC0 ... DC7.

The assignment of the other terminals:

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 1.0 | DC0 | Signal of the configurable digital input/output <br> DC0 |
| 1.1 | DC1 | Signal of the configurable digital input/output <br> DC1 |


| Terminal | Signal | Description |
| :---: | :---: | :---: |
| 1.2 | DC2 | Signal of the configurable digital input/output DC2 |
| 1.3 | DC3 | Signal of the configurable digital input/output DC3 |
| 1.4 | DC4 | Signal of the configurable digital input/output DC4 |
| 1.5 | DC5 | Signal of the configurable digital input/output DC5 |
| 1.6 | DC6 | Signal of the configurable digital input/output DC6 |
| 1.7 | DC7 | Signal of the configurable digital input/output DC7 |
| 1.8 | UP | Process voltage UP (24 V DC) |
| 1.9 | ZP | Process voltage ZP (0 V DC) |
| 2.0 | DI8 | Signal of the digital input DI8 |
| 2.1 | DI9 | Signal of the digital input DI9 |
| 2.2 | DI10 | Signal of the digital input DI10 |
| 2.3 | DI11 | Signal of the digital input DI11 |
| 2.4 | DI12 | Signal of the digital input DI12 |
| 2.5 | DI13 | Signal of the digital input DI13 |
| 2.6 | DI14 | Signal of the digital input DI14 |
| 2.7 | DI15 | Signal of the digital input DI15 |
| 2.8 | UP | Process voltage UP (24 V DC) |
| 2.9 | ZP | Process voltage ZP (0 V DC) |
| 3.0 | DO8 | Signal of the digital output DO8 |
| 3.1 | DO9 | Signal of the digital output DO9 |
| 3.2 | DO10 | Signal of the digital output DO10 |
| 3.3 | DO11 | Signal of the digital output DO11 |
| 3.4 | DO12 | Signal of the digital output DO12 |
| 3.5 | DO13 | Signal of the digital output DO13 |
| 3.6 | DO14 | Signal of the digital output DO14 |
| 3.7 | DO15 | Signal of the digital output DO15 |
| 3.8 | UP3 | Process voltage UP3 (24 V DC) |
| 3.9 | ZP | Process voltage ZP (0 V DC) |

WARNING!

## Removal/Insertion under power

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.
Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.
The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

Risk of damaging the PLC modules!
Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.


Fig. 220: Connection of the communication interface module CI522-MODTCP
\& Chapter 6.8.4.1 "Modbus communication interface module" on page 4568

## Connection of the digital inputs



Fig. 221: Connection of the digital inputs (DI8 ... DI15) to the module CI522-MODTCP The meaning of the LEDs is described in Displays $\stackrel{\xi}{ }{ }^{\circ}$ Chapter 5.2.6.5.2.10 "State LEDs" on page 1128.

## Connection of the digital outputs



Fig. 222: Connection of the digital output DO8. Proceed with the digital outputs DO9 ... DO15 in the same way
The meaning of the LEDs is described in Displays ${ }^{\mu}$ Chapter 5.2.6.5.2.10 "State LEDs" on page 1128.

## Connection of the configurable digital inputs/outputs

The following figure shows the connection of the configurable digital input/output DC0 and DC1. DC0 is connected as an input and DC1 is connected as an output. Proceed with the configurable digital inputs/outputs DC2 ... DC7 in the same way.

## CAUTION!

If a DC channel is used as input, the source for the input signals should be the impressed UP3 of the device $\left.{ }^{\star}\right\rangle$ Chapter 5.2.6.5.2.4 "Connections" on page 1113.


The meaning of the LEDs is described in Displays ${ }^{\leftrightarrows}$ Chapter 5.2.6.5.2.10 "State LEDs" on page 1128.

## Assignment of the Ethernet ports

The terminal unit for the communication interface module provides two Ethernet interfaces with the following pin assignment:

| Pin assignment | Interface | Pin | Signal | Description |
| :---: | :---: | :---: | :---: | :---: |
|  |  | 1 | TxD+ | Transmit data + |
|  |  | 2 | TxD- | Transmit data - |
|  |  | 3 | RxD+ | Receive data + |
|  | RJ45 | 4 | NC | Not connected |
|  | $\square 8$ | 5 | NC | Not connected |
|  |  | 6 | RxD- | Receive data - |
|  |  | 7 | NC | Not connected |
|  |  | 8 | NC | Not connected |
|  |  | Shield | Cable shield | Functional earth |

In corrosive environment, please protect unused connectors using the TA535 accessory.
Not supplied with this device.

②) Further information about wiring and cable types

## Internal data exchange

| Digital inputs (bytes) | 5 |
| :--- | :--- |
| Digital outputs (bytes) | 5 |
| Counter input data (words) | 4 |
| Counter output data (words) | 8 |

## Replacement of a Modbus TCP communication interface module:

The module must be powered off before it is replaced. If the configuration data is stored in the module, then the configuration data must be downloaded into the new module, either by using Modbus communication or by using the Modbus configurator which is contained in the Automation Builder distribution.
ⓨ Chapter 6.3.2.12.2 "Unbundled CI52x-MODTCP configuration" on page 1565

## Addressing

The IP address of the CI5221-MODTCP Module can be set with the "ABB IP Configuration Tool" $\Leftrightarrow$ Chapter 6.3.2.2.4.3 "Configuration of the IP settings with the IP configuration tool" on page 1457.
If the last byte of the IP is set to 0 , the address switch will be used instead.
Address switch position 255 is mapped to fixed IP 192.168.0.254 independent of other stored settings. This is a backup so the module can always get a valid IP address and can be configured by the "ABB IP Configuration Tool".
Address switch position 0 is mapped to last byte equal 1 and DHCP enabled.
The factory setting for the IP is 192.168.0.x (last byte is address switch).

The module reads the position of the rotary switches only during power-up, i.e. changes of the switch position during operation will have no effect until the next module initialization.

## I/O configuration

The CI522-MODTCP stores configuration parameters (IP address configuration, module parameters).
The digital I/O channels are configured via software.
Details about configuration are described in Parameterization $\Leftrightarrow$ Chapter 5.2.6.5.2.8 "Parameterization" on page 1120.

## Parameterization

## Parameters of the module

| Name | Value | Internal value | Internal value, <br> type | Default |
| :--- | :--- | :--- | :--- | :--- |
| Module ID ${ }^{1}$ ) | Internal | 7405 | WORD | 7405 |
| Ignore Module | Internal | 0 | BYTE | 0 |
| Parameter length | Internal | 47 | BYTE | 47 |


| Name | Value | Internal value | Internal value, type | Default |
| :---: | :---: | :---: | :---: | :---: |
| Error LED / Failsafe function (Table Error LED / Failsafe function Table 248 " Table Error LED / Failsafe function" on page 1122) | On | 0 | BYTE | 0 |
|  | Off by E4 | 1 |  |  |
|  | Off by E3 | 3 |  |  |
|  | On + failsafe | 16 |  |  |
|  | Off by E4 + failsafe | 17 |  |  |
|  | Off by E3 + failsafe | 19 |  |  |
| Master IP for Write restriction ${ }^{4}$ ) | No master IP Master IP | 0,0,0,0 W, X, y,z | ARRAY[0..3] OF BYTE | 0,0,0,0 |
| Master IP for Write restriction ${ }^{4}$ ) | No master IP Master IP | 0,0,0,0 W, X, y,z | ARRAY[0..3] OF BYTE | 0,0,0,0 |
| Master IP for Write restriction ${ }^{4}$ ) | No master IP Master IP | 0,0,0,0 W, $\mathrm{X}, \mathrm{y}, \mathrm{z}$ | ARRAY[0..3] OF BYTE | 0,0,0,0 |
| Master IP for Write restriction ${ }^{4}$ ) | No master IP Master IP | 0,0,0,0 W,X,y,z | ARRAY[0..3] OF BYTE | 0,0,0,0 |
| Master IP for Write restriction ${ }^{4}$ ) | No master IP Master IP | 0,0,0,0 W, X, y,z | ARRAY[0..3] OF BYTE | 0,0,0,0 |
| Master IP for Write restriction ${ }^{4}$ ) | No master IP Master IP | 0,0,0,0 W, $\mathrm{X}, \mathrm{y}, \mathrm{z}$ | ARRAY[0..3] OF BYTE | 0,0,0,0 |
| Master IP for Write restriction ${ }^{4}$ ) | No master IP Master IP | 0,0,0,0 W, X,y,z | ARRAY[0..3] OF BYTE | 0,0,0,0 |
| Master IP for Write restriction ${ }^{4}$ ) | No master IP Master IP | 0,0,0,0 W, X, y,z | ARRAY[0..3] OF BYTE | 0,0,0,0 |
| Timeout for Bus supervision | No supervision 10 ms timeout 20 ms timeout | $\begin{aligned} & 0 \\ & 1 \\ & 2 \end{aligned}$ | BYTE | No supervision |
| IO Mapping Structure ${ }^{3}$ ) | Fixed Mapping Dynamic Mapping | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | BYTE | 0 |
| Reserved | Internal | 0 | ARRAY[0..2] OF BYTE | 0,0,0 |
| Check supply | off on | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | BYTE | 1 |
| Fast counter | 0 $10^{2} \text { ) }$ | $\begin{aligned} & 0 \\ & : \\ & 10 \end{aligned}$ | BYTE | 0 |

Remarks:

| $\left.{ }^{1}\right)$ | With a faulty ID, the module reports a "parameter error" and does not <br> perform cyclic process data transmission. |
| :--- | :--- |
| $\left.{ }^{2}\right)$ | Counter operating modes $\leadsto$ Chapter 5.2.4.3.2.9 "Fast counter" <br> on page 606 |
| $\left.{ }^{3}\right)$ | Fixed Mapping means each module has its own Modbus registers <br> for data transfer independent of the I/O bus constellation description <br> G Chapter 6.8.4.1.2 "Modbus TCP registers" on page 4570. <br> Dynamic mapping means the structure of the IO Date is dependent on <br> the I/O bus constellation. Each I/O bus expansion module starts directly <br> after the module before on the next Word address. |
| $\left.{ }^{4}\right)$ | If none of the parameters is set all masters / clients in the network have <br> read and write rights on the CI52x-MODTCP device and its connected <br> expansion modules. <br> If at least one parameter is set only the configured masters / clients have <br> write rights on the CI52x-MODTCP device, all other masters / clients still <br> have read access to the CI52x-MODTCP device. |

Table 248: Table Error LED / Failsafe function

| Setting | Description |
| :--- | :--- |
| On | Error LED (S-ERR) lights up at errors of all <br> error classes, Failsafe-mode off |
| Off by E4 | Error LED (S-ERR) lights up at errors of error <br> classes E1, E2 and E3, Failsafe-mode off |
| Off by E3 | Error LED (S-ERR) lights up at errors of error <br> classes E1 and E2, Failsafe-mode off |
| On + Failsafe | Error LED (S-ERR) lights up at errors of all <br> error classes, Failsafe-mode on *) |
| Off by E4 + Failsafe | Error LED (S-ERR) lights up at errors of error <br> classes E1, E2 and E3, Failsafe-mode on *) |
| Off by E3 + Failsafe | Error LED (S-ERR) lights up at errors of error <br> classes E1 and E2, Failsafe-mode on *) |
| *) The parameter Behaviour DO at comm. error is only analyzed if the Failsafe-mode is ON. |  |

## Group parameters for the digital part

| Name | Value | Internal <br> value | Internal value, <br> type | Default |
| :--- | :--- | :--- | :--- | :--- |
| Input delay | 0.1 ms | 0 | BYTE | 0.1 ms |
|  | 1 ms | 2 | $0 \times 00$ |  |
|  | 8 ms |  |  |  |
| 32 ms |  |  |  |  |$\quad 3$|  |  |  |
| :--- | :--- | :--- |
| Detect short cir- <br> cuit at outputs | Off <br> On | 0 |


| Name | Value | Internal <br> value | Internal value, <br> type | Default |
| :--- | :--- | :--- | :--- | :--- |
| Behaviour DO at <br> comm. error ${ }^{1}$ ) | Off <br> Last value <br> Last value 5 sec <br> Last value 10 sec <br> Substitute value <br> Substitute value 5 <br> sec <br> Substitute value 10 <br> sec | 11 <br> 6 | BYTE <br> 12 | Off <br> $0 \times 00$ |
| Substitute value <br> at output | $0 \ldots 65535$ | $0000 \mathrm{~h} \ldots$ FFFFWVORD | 0 |  |
| Preventive <br> voltage feedback <br> monitoring for <br> DC0 ... DC7 ${ }^{2}$ ) | Off <br> On | 0 | 1 | 0x0000 |
| Detect voltage <br> overflow at out- <br> puts ${ }^{3}$ ) | Off <br> On | 0 | Off |  |

Remarks:

| ${ }^{1}$ ) | The parameter Behaviour DO at comm. error is apply to DC and DO <br> channels and only analyzed if the Failsafe-mode is ON. |
| :--- | :--- |
| ${ }^{2}$ ) | The state "externally voltage detected" appears, if the output of a <br> channel DC0 ... DC7 should be switched on while an externally voltage <br> is connected. In this case the start up is disabled, as long as the exter- <br> nally voltage is connected. The monitoring of this state and the resulting <br> diagnosis message can be disabled by setting the parameters to "OFF". |
| $\left.{ }^{3}\right)^{2}$ | The error state "voltage overflow at outputs" appears, if externally <br> voltage at digital outputs DC0 ... DC7 and accordingly DO8 ... DO15 <br> has exceeded the process supply voltage UP3 』 Chapter 5.2.6.5.2.4 <br> "Connections" on page 1113. The according diagnosis message "Voltage <br> overflow on outputs " can be disabled by setting the parameters on <br> "OFF". This parameter should only be disabled in exceptional cases for <br> voltage overflow may produce reverse voltage. |

## Diagnosis

Table 249: Structure of the Diagnosis Block

| Byte Number | Description | Possible Values |
| :--- | :--- | :--- |
| 1 | Diagnosis Byte, slot <br> number | $31=\mathrm{CI} 502-\mathrm{PNIO}$ (e. g. error at integrated <br> $8 \mathrm{DI} / 8 \mathrm{DO})$ <br> $1=1$ st connected S500 I/O Module <br> $\ldots$ |
| 2 | Diagnosis Byte, module <br> number | According to the I/O bus specification <br> passed on by modules to the fieldbus <br> master |


| Byte Number | Description | Possible Values |
| :--- | :--- | :--- |
| 3 | Diagnosis Byte, channel | According to the I/O bus specification <br> passed on by modules to the fieldbus <br> master |
| 4 | Diagnosis Byte, error code | According to the I/O bus specification <br> Bit 7 and bit 6, coded error class <br> $0=\mathrm{E} 1$ <br> $1=\mathrm{E} 2$ <br> $2=\mathrm{E} 3$ <br> $3=\mathrm{E} 4$ |
| 5 |  | Biagnosis Byte, flags 0 to bit 5, coded error description |
|  |  | According to the I/O bus specification <br> Bit $7: 1=$ coming error |
|  |  | Bit 6: $1=$ leaving error |

In cases of short circuit or overload, the digital outputs are turned off. The modules performs reactivation automatically. Thus an acknowledgement of the errors is not necessary. The error message is stored via the LED.

For diagnosis firmware version $\geq 3.2 .6$ is required.

| E1 ... E4 | d1 | d2 | d3 | d4 | Identifier 000 ... | AC500Display | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err | $\begin{array}{\|l\|} \hline \text { PS501 } \\ \text { PLC } \\ \text { Browser } \end{array}$ |  |  |
| $\begin{array}{\|l} \hline \text { Byte } 4 \\ \text { Bit } \\ 6 \ldots 7 \end{array}$ | - | Byte 1 | Byte 2 | Byte 3 | Byte 4 <br> Bit <br> 0 ... 5 | PNIO diagnosis block |  |  |
| Class | Interface | Device | Module | Channel | Error- <br> Identifier | Error message |  | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | $\left.{ }^{3}\right)$ |  |  |  |  |  |
| Module errors |  |  |  |  |  |  |  |  |
| 3 | - | 31 | 31 | 31 | 19 | Checksum error in the I/O module |  | Replace I/O module |
| 3 | - | 31 | 31 | 31 | 3 | Timeout in the I/O module |  |  |
| 3 | - | 31 | 31 | 31 | 40 | Different hard-/firmware versions in the module |  |  |
| 3 | - | 31 | 31 | 31 | 43 | Internal error in the module |  |  |


| E1 ... E4 | d1 | d2 | d3 | d4 | Identifier $000 \ldots 0$ | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err |  |  |
| Byte 4 <br> Bit <br> 6 ... 7 | - | Byte 1 | Byte 2 | Byte 3 | Byte 4 <br> Bit <br> 0 ... 5 |  |  |
| Class | $\begin{aligned} & \text { Inter- } \\ & \text { face } \end{aligned}$ | Device | Module | Channel | Error- <br> Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) |  |  |  |  |
| 3 | - | 31 | 31 | 31 | 36 | Internal data exchange failure |  |
| 3 | - | 31 | 31 | 31 | 9 | Overflow diagnosis buffer | Restart |
| 3 | - | 31 | 31 | 31 | 26 | Parameter error | Check Master |
| 3 | - | 31 | 31 | 31 | 11 | Process voltage UP too low | Check process supply voltage |
| 3 | - | 31 | 31 | 31 | 45 | Process voltage UP gone | Check process supply voltage |
| 3 | - | 31/1... 10 | 31 | 31 | 17 | No communication with I/O module | Replace I/O module |
| 3 | - | 1... 10 | 31 | 31 | 32 | Wrong I/O module type on socket | Replace I/O module / Check configuration |
| 4 | - | 1... 10 | 31 | 31 | 31 | At least one module does not support failsafe function | Check modules and parameterization |
| 4 | - | 1... 10 | 31 | 5 | 8 | I/O module removed from hot swap terminal unit or defective module on hot swap terminal unit ${ }^{9}$ ) | Plug I/O module, replace I/O module |


| E1 ... E4 | d1 | d2 | d3 | d4 | Identifier 000 ... | AC500- <br> Display <- Display | <- Display in |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err | PS501 <br> PLC <br> Browser |  |
|  | - | Byte 1 | Byte 2 | Byte 3 | $\begin{array}{\|l} \hline \text { Byte } 4 \\ \text { Bit } \\ 0 \text {... } 5 \end{array}$ | PNIO diagnosis block |  |
| Class | Inter- <br> face | Device | Module | Channel | Error-Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) |  |  |  |  |
| 4 | - | 1... 10 | 31 | 5 | 28 | Wrong I/O module plugged on hot swap terminal unit ${ }^{9}$ ) | Remove wrong I/O module and plug projected I/O module |
| 4 | - | 1... 10 | 31 | 5 | 42 | No communication with I/O module on hot swap terminal unit ${ }^{9}$ ) | Replace I/O module |
| 4 | - | 1... 10 | 31 | 5 | 54 | I/O module does not support hot swap $\left.{ }^{8}\right)^{9}$ ) | Power off system and replace I/O module |
| 4 | - | 1... 10 | 31 | 6 | 8 | Hot swap terminal unit configured but not found | Replace terminal unit by hot swap terminal unit |
| 4 | - | 1... 10 | 31 | 6 | 42 | No communication with hot swap terminal unit ${ }^{9}$ ) | Restart, if error persists replace terminal unit |
| 4 | 1... 6 | 255 | 2 | 0 | 45 | The connected Communication Module has no connection to the network | Check cabeling |
| 4 | - | 31 | 31 | 31 | 45 | Process voltage UP3 too low | Check process voltage |


| E1 ... E4 | d1 | d2 | d3 | d4 | Identifier $000 \ldots$ | AC500Display | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err | $\begin{aligned} & \hline \text { PS501 } \\ & \text { PLC } \\ & \text { Browser } \end{aligned}$ |  |  |
| $\begin{array}{\|l} \hline \text { Byte } 4 \\ \text { Bit } \\ 6 \ldots 7 \end{array}$ | - | Byte 1 | Byte 2 | Byte 3 | $\begin{array}{\|l} \hline \text { Byte } 4 \\ \text { Bit } \\ 0 \text {... } 5 \end{array}$ | PNIO diagnosis block |  |  |
| Class | Interface | Device | Module | Channel | Error- <br> Identifier | Error message |  | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | $\left.{ }^{3}\right)$ |  |  |  |  |  |
| 4 | - | 31 | 31 | 31 | 46 | Reverse voltage from digital outputs DO8...DO15 to UP3 ${ }^{4}$ ) |  | Check terminals |
| 4 | - | 31/1... 10 | 31 | 31 | 34 | No response during initialization of the I/O module |  | Replace I/O module |
| 4 | - | 31 | 31 | 31 | 11 | Process voltage UP3 too low |  | Check process supply voltage |
| 4 | - | 31 | 31 | 31 | 45 | Process voltage UP3 gone |  | Check process supply voltage |
| 4 | - | 31 | 31 | 31 | 10 | Voltage overflow at outputs (above UP3 level) ${ }^{5}$ ) |  | Check terminals/ check process supply voltage |
| Channel error digital |  |  |  |  |  |  |  |  |
| 4 | - | 31 | 2 | $8 . .15$ | 46 | Externally voltage detected at digital output DO8...DO15 ${ }^{6}$ ) |  | Check terminals |
| 4 | - | 31 | 4 | 0...7 | 46 | Externally voltage detected at digital output DC0...DC7 ${ }^{6}$ ) |  | Check terminals |
| 4 | - | 31 | 4 | 0... 7 | 47 | Short circuit at digital output DC0...DC7 ${ }^{7}$ ) |  | Check terminals |
| 4 | - | 31 | 2 | 8... 15 | 47 | Short circuit at digital output DO8...DO157) |  | Check terminals |

Remarks:

| ${ }^{1}$ ) | In AC500 the following interface identifier applies: <br> "-" = Diagnosis via bus-specific function blocks; 0 ... 4 or $10=$ Position of the Communication Module; $14=1 / \mathrm{O}$ bus; $31=$ Module itself <br> The identifier is not contained in the CI502-PNIO diagnosis block. |
| :---: | :---: |
| ${ }^{2}$ ) | With "Device" the following allocation applies: 31 = Module itself, $1 . .10=$ Expansion module |
| ${ }^{3}$ ) | With "Module" the following allocation applies dependent of the master: <br> Module error: 31 = Module itself <br> Channel error: Module type ( 1 = AI, 2 = DO, 3 = AO) |
| ${ }^{4}$ ) | This message appears, if externally voltages at one or more terminals DC0 ... DC7 oder DO8 ... DO15 cause that other digital outputs are supplied through that voltage (voltage feedback, see description in 'Connections' * Chapter 5.2.6.5.2.4 "Connections" on page 1113. All outputs of the apply digital output groups will be turned off for 5 seconds. The diagnosis message appears for the whole output group. |
| ${ }^{5}$ ) | The voltage at digital outputs DC0 ... DC7 and accordingly DO8 ... DO15 has exceeded the process supply voltage UP3 $\%$ Chapter 5.2.6.5.2.4 "Connections" on page 1113. Diagnosis message appears for the whole module. |
| ${ }^{6}$ ) | This message appears, if the output of a channel DC0 ... DC7 or DO8 ... DO15 should be switched on while an externally voltage is connected. In this case the start up is disabled, as long as the externally voltage is connected. Otherwise this could produce reverse voltage from this output to other digital outputs. This diagnosis message appears per channel. |
| ${ }^{7}$ ) | Short circuit: After a detected short circuit, the output is deactivated for 2000 ms . Then a new start up will be executed. This diagnosis message appears per channel. |
| $\left.{ }^{8}\right)$ | In case of an I/O module doesn't support hot swapping, do not perform any hot swap operations (also not on any other terminal units (slots)) as modules may be damaged or I/O bus communication may be disturbed. |
| ${ }^{9}$ ) | Diagnosis for hot swap available as of version index F0. |

## State LEDs

The LEDs are located at the front of module. There are 2 different groups:

- The 5 system LEDs (PWR, STA1 ETH, STA2 ETH, S-ERR and I/O-Bus) show the operation state of the module and display possible errors.
- The 29 process LEDs (UP, UP3, inputs, outputs, CH-ERR1 to CH-ERR3) show the process supply voltage and the states of the inputs and outputs and display possible errors.

Table 250: States of the 5 system LEDs

| LED | Color | OFF | ON | Flashing |
| :--- | :--- | :--- | :--- | :--- |
| PWR/RUN | Green | Process supply <br> voltage missing | Internal supply <br> voltage OK, <br> module ready for <br> communication <br> with I/O Con- <br> troller | Start-up / pre- <br> paring communi- <br> cation |
|  | Yellow | --- | --- |  |


| LED | Color | OFF | ON | Flashing |
| :--- | :--- | :--- | :--- | :--- |
| STA1 ETH <br> (System LED <br> "BF") | Green | --- | Device config- <br> ured, cyclic data <br> exchange run- <br> ning | Device config- <br> ured, acyclic data <br> exchange run- <br> ning |
|  | Red | --- | Communication <br> error (timeout) <br> appeared | IP address error |
| STA2 ETH <br> (System LED <br> "SF") | Green | Device has valid <br> parameters | Device is running <br> parameterization <br> sequenze | Device has no <br> parameters |
|  | Red | Red | --- | Device has <br> invalid parame- <br> ters |
|  | No error | No expansion <br> modules con- <br> nected or com- <br> munication error | Internal error <br> Expansion <br> modules con- <br> nected and <br> operational | --- |
| I/O-Bus | Green | No connection at <br> Ethernet inter- <br> face | Connected to <br> Ethernet inter- <br> face | ---- |
| ETH1 | Ye-- | Device is trans- <br> mitting telegrams | Device is trans- <br> mitting telegrams |  |
|  | Yellow | No connection at <br> Ethernet inter- <br> face | Connected to <br> Ethernet inter- <br> face | ---- |
| ETH2 | Yellow | Device is trans- <br> mitting telegrams | Device is trans- <br> mitting telegrams |  |

Table 251: States of the 29 process LEDs

| LED | Color | OFF | ON | Flashing |
| :--- | :--- | :--- | :--- | :--- |
| DC0 ... DC7 | Yellow | Input/Output is <br> OFF | Input/Output is <br> ON | -- |
| DI8 ... DI15 | Yellow | Input is OFF | Input is ON (the <br> input voltage is <br> even displayed if <br> the supply <br> voltage is OFF) | -- <br> DO8 ... DO15 <br> UP Yellow |
| Green | Output is OFF | Output is ON | -- |  |
| UP3 | Process supply <br> voltage missing | Process supply <br> voltage OK and <br> initialization fin- <br> ished | --- |  |
| CH-ERR1 to CH- <br> ERR3 | Red | Process supply <br> voltage missing | Process supply <br> voltage OK | --- |

## Technical data

## Technical data of the module

The system data of AC500 and S500 are applicable to the standard version Chapter 5.1.2 "System data AC500" on page 166.

The system data of AC500-XC are applicable to the XC version ${ }^{\circledR}$ Chapter 5.1.3 "System data AC500-XC" on page 169.

Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter | Value |
| :---: | :---: |
| Process supply voltages UP/UP3 |  |
| Rated value | 24 V DC (for inputs and outputs) |
| Max. load for the terminals | 10 A |
| Protection against reversed voltage | Yes |
| Rated protection fuse on UP/UP3 | 10 A fast |
| Galvanic isolation | Ethernet interface against the rest of the module |
| Inrush current from UP (at power up) | On request |
| Current consumption via UP (normal operation) | 0.15 A |
| Current consumption via UP3 | 0.06 A + 0.5 A max. per output |
| Connections | Terminals 1.8 and 2.8 for +24 V (UP) <br> Terminal 3.8 for +24 V (UP3) <br> Terminals 1.9, 2.9 and 3.9 for 0 V (ZP) |
| Max. power dissipation within the module | 6 W |
| Number of digital inputs | 8 |
| Number of digital outputs | 8 |
| Number of configurable digital inputs/outputs | 8 |
| Reference potential for all digital inputs and outputs | Negative pole of the supply voltage, signal name ZP |
| Ethernet | 10/100 base-TX, internal switch, $2 \times$ RJ45 socket |
| Setting of the I/O device identifier | With 2 rotary switches at the front side of the module |
| Diagnosis | See Diagnosis and Displays $\stackrel{H}{ }$ Chapter 5.2.6.5.2.9 "Diagnosis" on page 1123 |
| Operation and error displays | 34 LEDs (totally) |
| Weight (without terminal unit) | Ca. 125 g |
| Mounting position | Horizontal or vertical with derating (output load reduced to $50 \%$ at $+40^{\circ} \mathrm{C}$ per group) |
| Extended ambient temperature (XC version) | > +60 ${ }^{\circ} \mathrm{C}$ on request |
| Cooling | The natural convection cooling must not be hindered by cable ducts or other parts in the control cabinet. |

## NOTICE!

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and temporary overvoltage up to 30 V DC.

## Multiple overloads

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an internal smart high-side switch.

## Technical data of the digital inputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 8 |
| Distribution of the channels into groups | 1 group of 8 channels |
| Terminals of the channels DI8 ... DI15 | Terminals $2.0 \ldots 2.7$ |
| Reference potential for all inputs | Terminals $1.9 \ldots 3.9$ (negative pole of the supply <br> voltage, signal name ZP) |
| Indication of the input signals | 1 yellow LED per channel, the LED is ON when <br> the input signal is high (signal 1) |
| Input type (according EN 61131-2) | Type 1 |
| Input delay (0->1 or 1->0) | Typ. 0.1 ms, configurable from $0.1 \mathrm{~ms} \ldots 32 \mathrm{~ms}$ |
| Input signal voltage | 24 V DC |
|  | Signal 0 |
| Undefined Signal | $-3 \mathrm{~V} \ldots+5 \mathrm{~V}$ |
|  | $>+5 \mathrm{~V} \ldots<+15 \mathrm{~V}$ |
| Ripnal 1 | $+15 \mathrm{~V} \ldots+30 \mathrm{~V}$ |
| Ripple with signal 1 | Within $-3 \mathrm{~V} \ldots+5 \mathrm{~V}$ |
| Input current per channel | Within $+15 \mathrm{~V} \ldots+30 \mathrm{~V}$ |
|  | Input voltage +24 V |
| Input voltage +5 V | Typ. 5 mA |
|  | Input voltage +15 V |
| Input voltage +30 V | $>1 \mathrm{~mA}$ |
| Max. cable length | $>2 \mathrm{~mA}$ |
|  | Shielded |
| Unshielded | $<8 \mathrm{~mA}$ |
|  | 600 m |

## Technical data of the digital outputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 8 |
| Distribution of the channels into groups | 1 group of 8 channels |


| Parameter | Value |
| :---: | :---: |
| Terminals of the channels DO8 ... DO15 | Terminals 3.0 ... 3.7 |
| Reference potential for all outputs | Terminals 1.9 ... 3.9 (negative pole of the supply voltage, signal name ZP) |
| Common power supply voltage | For all outputs terminal 3.8 (positive pole of the supply voltage, signal name UP3) |
| Output voltage for signal 1 | UP3 (-0.8 V) |
| Output delay (0->1 or 1->0) | On request |
| Output current |  |
| Rated value per channel | 500 mA at UP3 $=24 \mathrm{~V}$ |
| Max. value (all channels together) | 4 A |
| Leakage current with signal 0 | $<0.5 \mathrm{~mA}$ |
| Fuse for UP3 | 10 A fast |
| Demagnetization with inductive DC load | Via internal varistors (see figure below this table) |
| Output switching frequency |  |
| With resistive load | On request |
| With inductive loads | Max. 0.5 Hz |
| With lamp loads | 11 Hz max. at 5 W max. |
| Short-circuit-proof / overload-proof | Yes |
| Overload message ( $1>0.7$ A) | Yes, after ca. 100 ms |
| Output current limitation | Yes, automatic reactivation after short circuit/overload |
| Resistance to feedback against 24 V signals | Yes (software-controlled supervision) |
| Max. cable length |  |
| Shielded | 1000 m |
| Unshielded | 600 m |

The following drawing shows the circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off.


Fig. 223: Digital input/output (circuit diagram)
1 Digital output
2 Varistors for demagnetization when inductive loads are turned off

Technical data of the configurable digital inputs/outputs
Each of the configurable I/O channels is defined as input or output by the user program. This is done by interrogating or allocating the corresponding channel.

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 8 inputs/outputs (with transistors) |
| Distribution of the channels into groups | 1 group for 8 channels |
| If the channels are used as inputs |  |
|  | Channels DC0 ... DC7 |
| If the channels are used as outputs | Terminals $1.0 \ldots 1.7$ |
| Channels DC0 ... DC7 |  |
| Indication of the input/output signals | 1 yellow LED per channel, the LED is ON when <br> the input/output signal is high (signal 1) |
| Galvanic isolation | From the Ethernet network |

## Technical data of the digital inputs/outputs if used as inputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 8 |
| Distribution of the channels into groups | 1 group of 8 channels |
| Terminals of the channels DC0 ... DC7 | Terminals 1.0 ... 1.7 |
| Reference potential for all inputs | Terminals 1.9 ... 3.9 (negative pole of the supply voltage, signal name ZP) |
| Indication of the input signals | 1 yellow LED per channel, the LED is ON when the input signal is high (signal 1) |
| Input type (according EN 61131-2) | Type 1 |
| Input delay (0->1 or 1->0) | Typ. 0.1 ms , configurable from 0.1 ms .. 32 ms |
| Input signal voltage | 24 V DC |
| Signal 0 | $-3 \mathrm{~V} \ldots+5 \mathrm{~V}$ ) |
| Undefined Signal | > +5 V ... < +15 V |
| Signal 1 | +15 V ... +30 V |
| Ripple with signal 0 | Within $-3 \mathrm{~V} \ldots+5 \mathrm{~V}$ *) |
| Ripple with signal 1 | Within +15 V ... +30 V |
| Input current per channel |  |
| Input voltage +24 V | Typ. 5 mA |
| Input voltage +5 V | $>1 \mathrm{~mA}$ |
| Input voltage +15 V | $>2 \mathrm{~mA}$ |
| Input voltage +30 V | $<8 \mathrm{~mA}$ |
| Max. cable length |  |
| Shielded | 1000 m |
| Unshielded | 600 m |

[^20]
## Technical data of the digital inputs/outputs if used as outputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 8 |
| Distribution of the channels into groups | 1 group of 8 channels |
| Terminals of the channels DC0 ... DC7 | Terminals 1.0 ... 1.7 |
| Reference potential for all outputs | Terminals 1.9 ... 3.9 (negative pole of the supply voltage, signal name ZP) |
| Common power supply voltage | For all outputs terminal 3.8 (positive pole of the supply voltage, signal name UP3) |
| Output voltage for signal 1 | UP3 (-0,8 V) |
| Output delay (0->1 or 1->0) | On request |
| Output current |  |
| Rated value per channel | 500 mA at UP3 $=24 \mathrm{~V}$ |
| Max. value (all channels together) | 4 A |
| Leakage current with signal 0 | $<0.5 \mathrm{~mA}$ |
| Fuse for UP3 | 10 A fast |
| Demagnetization with inductive DC load | Via internal varistors (see figure below this table) |
| Output switching frequency |  |
| With resistive load | On request |
| With inductive loads | Max. 0.5 Hz |
| With lamp loads | 11 Hz max. at 5 W max. |
| Short-circuit-proof / overload-proof | Yes |
| Overload message ( $1>0.7$ A) | Yes, after ca. 100 ms |
| Output current limitation | Yes, automatic reactivation after short circuit/overload |
| Resistance to feedback against 24V signals | Yes (software-controlled supervision) |
| Max. cable length |  |
| Shielded | 1000 m |
| Unshielded | 600 m |

The following drawing shows the circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off.


Fig. 224: Digital input/output (circuit diagram)
1 Digital input/output
2 For demagnetization when inductive loads are turned off

Technical data of the fast counter

| Parameter | Value |
| :--- | :--- |
| Used inputs | Terminal 2.0 (DI8),Terminal 2.1 (DI9) |
| Used outputs | Terminal 3.0 (DO8) |
| Counting frequency | Depending on operation mode: |
|  | Mode 1- 6: max. 200 kHz |
|  | Mode 7: max. 50 kHz |
|  | Mode 9: max. 35 kHz |
|  | Mode 10: max. 20 kHz |

## \# Chapter 6.8.2.12 "Fast counters in AC500 devices" on page 4536

How to prepare a device as fast counter and how to connect it to the PLC is described in an application example.

## Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm

The dimensions are in mm and in brackets in inch.

## Ordering data

| Ordering No. | Scope of delivery | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 222 200 <br> R0001 | Cl522-MODTCP, Modbus TCP com- <br> munication interface module, 8 DC, <br> 8 DI and 8 DO | Active |
| 1SAP 422 200 <br> R0001 | Cl522-MODTCP-XC, Modbus TCP <br> communication interface module, <br> 8 DC, 8 DI and 8 DO, XC version | Active |

> *) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.6.6 PROFIBUS

### 5.2.6.6.1 CI541-DP

## Features

- 4 configurable analog inputs (2-wire/single-ended) or 2 configurable analog inputs (3-wire/ differential)
Resolution 12 bits including sign
- 2 analog outputs

Resolution 12 bits including sign

- 8 digital inputs 24 V DC in 1 group
- 8 digital outputs 24 V DC in 1 group, 0.5 A max.
- Fast counter
- Module-wise galvanically isolated
- XC version for usage in extreme ambient conditions available


1 I/O bus
2 Allocation between terminal number and signal name
36 yellow LEDs to display the signal states of the analog inputs/outputs (AIO ... AI3, AO0 ... AO1)
48 yellow LEDs to display the signal states of the digital inputs (DIO ... DI7)
58 yellow LEDs to display the signal states of the digital outputs (DO0 ... DO7)
62 green LEDs to display the process supply voltage UP and UP3
3 red LEDs to display errors (CH-ERR1, CH-ERR2, CH-ERR3)
5 system LEDs: PWR/RUN, STA1 DP, STA2 DP, S-ERR, I/O-Bus
Label
2 rotary switches for setting the PROFIBUS ID
9-pin D-SUB connector to connect the PROFIBUS DP signals
Terminal unit
DIN rail
Sign for XC version

## Intended purpose

The PROFIBUS DP communication interface module is used as decentralized I/O module in PROFIBUS DP networks. Depending on the used terminal unit the network connection is performed either via 9-pole female D-sub or via 10 terminals (screw-type or spring terminals) which are integrated in the terminal unit. The communication interface module contains 22 I/O channels.
The inputs/outputs are galvanically isolated from the PROFIBUS DP network. There is no potential separation between the channels. The configuration of the analog inputs/outputs is performed by software.
For usage in extreme ambient conditions (e.g. wider temperature and humidity range), a special XC version of the device is available.

- 4 configurable analog inputs (2-wire/single-ended) or 2 configurable analog inputs (3-wire/ differential)
Resolution 12 bits including sign
- 2 analog outputs

Resolution 12 bits including sign

- 8 digital inputs 24 V DC in 1 group
- 8 digital outputs 24 V DC in 1 group, 0.5 A max.
- Fast counter
- Module-wise galvanically isolated
- XC version for usage in extreme ambient conditions available


## Diagnosis settings

The current CI54x does not run in combination with a V3 PLC if in the "General" tab the parameter "Diagnosis behavior" is set to "AC500 V3 compatible". How to change the setting in your $A B$ project is described below.


1. Double click in the "Device" tree on "CI541_IO".
$\Rightarrow$ The tab for the various settings opens.
2. Double click on the "General" tab.
3. Double click on the "Value" of the parameter "Diagnosis behavior".
4. Click on the small arrow.
$\Rightarrow$ A submenu with two values opens.
5. Click on "AC500 V2 compatible" as setting.
6. Close the tab.

After changing the parameter to "AC500 V2 compatible" the CI54x get in "RUN".
If the CI54x indicates a S500 diagnosis message, following AC500 diagnosis entry (" 655374 CI54x communication interface module is sending not supported diagnosis format - Check configuration and FW revision of communication interface module") is shown in the diagnosis editor and history. This diagnosis message does not have impact to cyclic data exchange between the master and the CI 54 x .

In case of a parameter change from V 2 to V 3 the parameter at the Cl 54 x of V 3 has the same value than at the CI54x below V2 (that means AC500 V2 compatible).

## Functionality

| Parameter | Value |
| :--- | :--- |
| Interface | PROFIBUS |
| Protocol | PROFIBUS DP (DP-V0 and DP-V1) |


| Parameter | Value |
| :---: | :---: |
| Power supply | From the process supply voltage UP |
| Supply of the electronic circuitry of the I/O expansion modules attached | Through the I/O bus interface (I/O bus) |
| Rotary switches | For setting the PROFIBUS ID for configuration purposes (00h to FFh) |
| Expandability | Max. 10 S500 I/O modules |
| Fast counter | Integrated, configurable operating modes |
| LED displays | For system displays, signal states, errors and power supply |
| External supply voltage | Via terminals ZP, UP and UP3 (process supply voltage 24 V DC) |
| Effect of incorrect input terminal connection | Wrong or no signal detected, no damage up to 35 V |
| Required terminal unit | TU509, TU510, TU517 or TU518 $\stackrel{y}{ }{ }^{\wedge}$ Chapter 5.2.7.3 "TU509 and TU510 for communication interface modules" on page $1274 \Leftrightarrow$ Chapter 5.2.7.4 "TU517 and TU518 for communication interface modules" on page 1278 |

## Connections

## General

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

The PROFIBUS DP communication interface module CI541-DP is plugged on the I/O terminal units TU509 ${ }^{\mu}$ chapter 5.2.7.3 "TU509 and TU510 for communication interface modules" on page 1274 or TU510 $\&$ Chapter 5.2.7.3 "TU509 and TU510 for communication interface modules" on page 1274 and accordingly TU517 $\Leftrightarrow$ Chapter 5.2.7.4 "TU517 and TU518 for communication interface modules" on page 1278 or TU518 $\Leftrightarrow$ Chapter 5.2.7.4 "TU517 and TU518 for communication interface modules" on page 1278. Properly seat the module and press until it locks in place. The terminal unit is mounted on a DIN rail or with 2 screws plus the additional accessory for wall mounting $\stackrel{y}{ }{ }^{\circ}$ Chapter 5.2.8.2.5 "TA526-Wall mounting accessory" on page 1324.
The connection of the I/O channels is carried out using the 30 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

The terminals 2.8 and 3.8 as well as 2.9, 3.9 and 4.9 are interconnected within the terminal unit and have always the same assignment, independent of the inserted module:

Terminals 2.8 and 3.8: Process supply voltage UP $=+24 \mathrm{~V}$ DC
Terminal 4.8: Process supply voltage UP3 $=+24 \mathrm{~V}$ DC
Terminals 2.9, 3.9 and 4.9: Process supply voltage $\mathrm{ZP}=0 \mathrm{~V}$


With a separate UP3 power supply, the digital outputs can be switched off externally. This way, an emergency-off functionality can be realized.

## Do not connect any voltages externally to digital outputs!

Reason: Externally voltages at an output or several outputs may cause that other outputs are supplied through that voltage instead of voltage UP3 (reverse voltage). This ist not intended usage.

## CAUTION!

## Risk of malfunction by unintended usage!

If the function cut-off of the digital outputs is to be used by deactivation of the supply voltage UP3, be sure that no external voltage is conncted at the outputs DO0 ... DO7.

## Possibilities of connection

## Connection on terminal units TU509 or TU510

The assignment of the 9-pole female D-sub for the PROFIBUS signals:

|  | 1 | --- | Reserved |
| :---: | :---: | :---: | :---: |
|  | 2 | --- | Reserved |
|  | 3 | B | Data line B (receive and send line, positive) |
|  | 4 | --- | Reserved |
|  | 5 | DGND | Reference potential for data transmissions and +5 V |
|  | 6 | VP (5 V) | +5 V (Power supply voltage for terminating resistors) |
|  | 7 | --- | Reserved |
|  | 8 | A | Data line A (receive and send line, negative) |
|  | 9 | --- | Reserved |
|  | Shield | Shield | Shield, functional earth |

## Bus termination

The line ends of the bus segment must be equipped with bus terminating resistors. Normally, these resistors are integrated in the interface connectors.


The grounding of the shield should take place at the control cabinet $\&$ Chapter 5.1.2 "System data AC500" on page 166.

## Mounting on terminal units TU517 or TU518

The assignment of the terminals 1.0 ... 1.9:

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 1.0 | B | Data line B (receive and send line, positive) |
| 1.1 | B | Data line B (receive and send line, positive) |
| 1.2 | A | Data line A (receive and send line, negative) |
| 1.3 | A | Data line A (receive and send line, negative) |
| 1.4 | TermB | Bus termination data line B |
| 1.5 | TermB | Bus termination data line B |
| 1.6 | TermA | Bus termination data line A |
| 1.7 | TermA | Bus termination data line A |
| 1.8 | DGND | Reference potential for data transmission |
| 1.9 | DGND | Reference potential for data transmission |

At the line ends of a bus segment, terminating resistors must be connected. If using TU517/ TU518, the bus terminating resistors can be enabled by connecting the terminals TermA and TermB to the data lines $A$ and $B$ (no external terminating resistors are required, see figure below).



If using TU517/TU518, note that the terminating resistors are not located inside the TU, but inside the communication interface module CI541-DP. I. e. when removing the device from the TU, the bus terminating resistors are not connected to the bus any more. The bus itself will not be disconnected if a device is removed.

If using TU517/TU518 the max. permitted transmission rate is limited to 1.5 MBaud.

The grounding of the shield should take place at the control cabinet ${ }^{\wedge} \Rightarrow$ Chapter 5.1.2 "System data AC500" on page 166.

## Technical data bus cable

| Parameter | Value |
| :--- | :--- |
| Type | Twisted pair (shielded) |
| Characteristic impedance | $135 \ldots . .165 \Omega$ |
| Cable capacitance | $<30 \mathrm{pF} / \mathrm{m}$ |
| Conductor diameter of the cores | $\geq 0.64 \mathrm{~mm}$ |
| Conductor cross section of the cores | $\geq 0.34 \mathrm{~mm}^{2}$ |
| Cable resistance per core | $\leq 55 \Omega / \mathrm{km}$ |
| Loop resistance (resistance of two cores) | $\leq 110 \Omega / \mathrm{km}$ |

## Cable length

The maximum possible cable length of a PROFIBUS subnet within a segment depends on the transmission rate (baud rate).

| Transmission rate | Maximum cable length |
| :--- | :--- |
| 9.6 kBaud to 93.75 kBaud | 1200 m |
| 187.5 kBaud | 1000 m |
| 500 kBaud | 400 m |
| 1.5 MBaud | 200 m |
| 3 MBaud to 12 MBaud | 100 m |

The assignment of the other terminals:

| Terminal | Signal | Description |
| :---: | :---: | :---: |
| 2.0 | Al0+ | Positive pole of analog input signal 0 |
| 2.1 | Al1+ | Positive pole of analog input signal 1 |
| 2.2 | Al2+ | Positive pole of analog input signal 2 |
| 2.3 | Al3+ | Positive pole of analog input signal 3 |
| 2.4 | AI- | Negative pole of analog input signals 0 to 3 |
| 2.5 | AO0+ | Positive pole of analog output signal 0 |
| 2.6 | AO1+ | Positive pole of analog output signal 1 |
| 2.7 | AI- | Negative pole of analog output signals 0 and 1 |
| 2.8 | UP | Process voltage UP (24 V DC) |
| 2.9 | ZP | Process voltage ZP (0 V DC) |
| 3.0 | DIO | Signal of the digital input DIO |
| 3.1 | DI1 | Signal of the digital input DI1 |
| 3.2 | DI2 | Signal of the digital input DI2 |
| 3.3 | DI3 | Signal of the digital input DI3 |
| 3.4 | DI4 | Signal of the digital input DI4 |
| 3.5 | DI5 | Signal of the digital input DI5 |
| 3.6 | DI6 | Signal of the digital input DI6 |
| 3.7 | DI7 | Signal of the digital input DI7 |
| 3.8 | UP | Process voltage UP (24 V DC) |
| 3.9 | ZP | Process voltage ZP (0 V DC) |
| 4.0 | DO0 | Signal of the digital output DOO |
| 4.1 | DO1 | Signal of the digital output DO1 |
| 4.2 | DO2 | Signal of the digital output DO2 |
| 4.3 | DO3 | Signal of the digital output DO3 |
| 4.4 | DO4 | Signal of the digital output DO4 |
| 4.5 | DO5 | Signal of the digital output DO5 |
| 4.6 | DO6 | Signal of the digital output DO6 |
| 4.7 | DO7 | Signal of the digital output DO7 |
| 4.8 | UP3 | Process voltage UP3 (24 V DC) |
| 4.9 | ZP | Process voltage ZP (0 V DC) |

## WARNING!

## Removal/Insertion under power

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.
Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.
Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.
The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

Risk of damaging the PLC modules!
Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

For the open-circuit detection (cut wire), each channel is pulled up to "plus" by a high-resistance resistor. If nothing is connected, the maximum voltage will be read in then.

Analog signals are always laid in shielded cables. The cable shields are grounded at both ends of the cables. In order to avoid unacceptable potential differences between different parts of the installation, low resistance equipotential bonding conductors must be laid.
For simple applications (low electromagnetic disturbances, no high requirement on precision), the shielding can also be omitted.


Fig. 225: Connection of the PROFIBUS DP communication interface module CI541-DP

## Connection of the digital inputs



Fig. 226: Connection of the digital input DIO (Proceed with the digital inputs DI1 ... DI7 in the same way)
The meaning of the LEDs is described in Displays ${ }^{\mu}$ Chapter 5.2.6.6.1.11 "State LEDs" on page 1165.

## Connection of the digital outputs



Fig. 227: Connection of the digital output DO0 (Proceed with the digital outputs DO1 ... DO7 in the same way)

The meaning of the LEDs is described in Displays $\Leftrightarrow$ Chapter 5.2.6.6.1.11 "State LEDs" on page 1165.

## Connection of resistance thermometers in 2-wire configuration to the analog inputs

When resistance thermometers ( $\mathrm{Pt} 100, \mathrm{Pt} 1000, \mathrm{Ni} 1000$ ) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the module CI541-DP provides a constant current source which is multiplexed over the max. 4 analog input channels.


Fig. 228: Connection of resistance thermometers in 2-wire configuration to the analog input AIO (Proceed with the analog inputs Al1 ... Al3 in the same way)

The following measuring ranges can be configured ${ }^{4}$ " Chapter 5.2.6.6.1.9 "Parameterization" on page $1155{ }^{\wedge}>$ Chapter 5.2.6.6.1.12 "Measuring ranges" on page 1166:

| Pt100 | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 2-wire configuration, 1 <br> channel used |
| :--- | :--- | :--- |
| Pt1000 | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 2-wire configuration, 1 <br> channel used |
| Ni1000 | $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ | 2-wire configuration, 1 <br> channel used |

The function of the LEDs is described under Diagnosis and displays / Displays $\Leftrightarrow$ Chapter 5.2.6.6.1.11 "State LEDs" on page 1165.

The module CI541-DP performs a linearization of the resistance characteristic.
To avoid error messages from unused analog input channels, configure them as "unused".

## Connection of resistance thermometers in 3-wire configuration to the analog inputs

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the module CI541-DP provides a constant current source which is multiplexed over the max. 4 analog input channels.


Fig. 229: Connection of resistance thermometers in 3-wire configuration to the analog inputs AIO and AI1 (Proceed with the analog inputs AI2 and AI3 in the same way)

With 3-wire configuration, 2 adjacent analog channels belong together (e. g. the channels 0 and 1). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0 ), the next higher address must be the odd address (channel 1).
The constant current of one channel flows through the resistance thermometer. The constant current of the other channel flows through one of the cores. The module calculates the measured value from the two voltage drops and stores it under the input with the higher channel number (e. g. I1).
In order to keep measuring errors as small as possible, it is necessary to have all the involved conductors in the same cable. All the conductors must have the same cross section.

The following measuring ranges can be configured ${ }^{\mu}$ Chapter 5.2.6.6.1.9 "Parameterization" on page $1155{ }^{\star} \Rightarrow$ Chapter 5.2.6.6.1.12 "Measuring ranges" on page 1166:

| Pt100 | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 3-wire configuration, 2 chan- <br> nels used |
| :--- | :--- | :--- |
| Pt1000 | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 3-wire configuration, 2 chan- <br> nels used |
| Ni1000 | $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ | 3-wire configuration, 2 chan- <br> nels used |

The function of the LEDs is described under Diagnosis and displays / Displays $\Leftrightarrow$ Chapter 5.2.6.6.1.11 "State LEDs" on page 1165.

The module CI541-DP performs a linearization of the resistance characteristic.
To avoid error messages from unused analog input channels, configure them as "unused".

## Connection of active-type analog sensors (Voltage) with galvanically isolated power supply to the analog inputs



Fig. 230: Connection of active-type analog sensors (voltage) with galvanically isolated power supply to the analog input AIO (Proceed with the analog inputs AI1 ... AI3 in the same way)

The following measuring ranges can be configured $\stackrel{\star}{ }{ }^{\circ}$ Chapter 5.2.6.6.1.9 "Parameterization" on page $\left.1155{ }^{*}\right\rangle$ Chapter 5.2.6.6.1.12 "Measuring ranges" on page 1166:

| Voltage | $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ | 1 channel used |
| :--- | :--- | :--- |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | 1 channel used |

The function of the LEDs is described under Diagnosis and displays / Displays ${ }_{幺}{ }^{\circ}$ Chapter 5.2.6.6.1.11 "State LEDs" on page 1165.

To avoid error messages from unused analog input channels, configure them as "unused".

Connection of active-type analog sensors (Current) with galvanically isolated power supply to the analog inputs


Fig. 231: Connection of active-type analog sensors (current) with galvanically isolated power supply to the analog input AIO (Proceed with the analog inputs Al1 ... AI3 in the same way)
The following measuring ranges can be configured ${ }^{\mu} \Rightarrow$ Chapter 5.2.6.6.1.9 "Parameterization" on page 1155 * Chapter 5.2.6.6.1.12 "Measuring ranges" on page 1166:

| Current | $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 1 channel used |
| :--- | :--- | :--- |
| Current | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 1 channel used |

The function of the LEDs is described under 'State LEDs' $\Longleftrightarrow$ Chapter 5.2.6.6.1.11 "State LEDs" on page 1165.

Unused input channels can be left open-circuited, because they are of low resistance.
To avoid error messages through unused analog input channels in measuring range $4 \ldots 20 \mathrm{~mA}$, these channels should be configured as "Not used".

## Connection of active-type analog sensors (Voltage) with no galvanically isolated power supply to the analog inputs

The following figure shows the connection of active-type analog sensors (voltage) with no galvanically isolated power supply to the analog input AIO. Proceed with the analog inputs $\mathrm{Al} 1 . . \mathrm{Al} 3$ in the same way.


Fig. 232: Connection of active-type analog sensors (voltage) with no galvanically isolated power supply to the analog input AIO (Proceed with the analog inputs Al1 ... AI3 in the same way)

## CAUTION!

## Risk of faulty measurements!

The negative pole at the sensors must not have too big a potential difference with respect to ZP (max. $\pm 1 \mathrm{~V}$ ).
Make sure that the potential difference never exceeds $\pm 1 \mathrm{~V}$ (also not with long cable lengths).

The following measuring ranges can be configured $\left.{ }^{\$}\right\rangle$ Chapter 5.2.6.6.1.9 "Parameterization" on page 1155 * Chapter 5.2.6.6.1.12 "Measuring ranges" on page 1166:

| Voltage | $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ | 1 channel used |
| :--- | :--- | :--- |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | 1 channel used |

The function of the LEDs is described under Diagnosis and displays / Displays ${ }^{\mu}$ Chapter 5.2.6.6.1.11 "State LEDs" on page 1165.

To avoid error messages from unused analog input channels, configure them as "unused".

## Connection of passive-type analog sensors (Current) to the analog inputs



Fig. 233: Connection of passive-type analog sensors (current) to the analog input AIO (Proceed with the analog inputs Al1 ... Al3 in the same way)
The following measuring ranges can be configured ${ }_{y}{ }^{\mu}$ Chapter 5.2.6.6.1.9 "Parameterization" on page $1155 \stackrel{\mu}{\mu}$ Chapter 5.2.6.6.1.9 "Parameterization" on page 1155 :

| Current | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 1 channel used |
| :--- | :--- | :--- |

The function of the LEDs is described under Diagnosis and displays / Displays ${ }^{\mu} \boldsymbol{y}$ Chapter 5.2.6.6.1.11 "State LEDs" on page 1165.

## CAUTION!

## Risk of overloading the analog input!

If an analog current sensor supplies more than 25 mA for more than 1 second during initialization, this input is switched off by the module (input protection).

Use only sensors with fast initialization or without current peaks higher than 25 mA . If not possible, connect a 10 -volt Zener diode in parallel to Alx+ and ZP.

Unused input channels can be left open-circuited, because they are of low resistance.
To avoid error messages through unused analog input channels in measuring range $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$, these channels should be configured as "Not used".

## Connection of active-type analog sensors (Voltage) to differential analog inputs

Differential inputs are very useful, if analog sensors are used which are remotely non-isolated (e.g. the minus terminal is remotely grounded).

The evaluation using differential inputs helps to considerably increase the measuring accuracy and to avoid ground loops.
With differential input configurations, two adjacent analog channels belong together (e.g. the channels 0 and 1 ). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0 ), the next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1).

The analog value is calculated by subtraction of the input value with the higher address from the input value of the lower address.
The converted analog value is available at the odd channel (higher address).

## CAUTION!

## Risk of faulty measurements!

The negative pole at the sensors must not have too big a potential difference with respect to ZP (max. $\pm 1 \mathrm{~V}$ ).
Make sure that the potential difference never exceeds $\pm 1 \mathrm{~V}$.


Fig. 234: Connection of active-type analog sensors (voltage) to differential analog inputs AIO and Al1 (Proceed with AI2 and AI3 in the same way)
The following measuring ranges can be configured ${ }^{\sharp}$ Chapter 5.2.6.6.1.9 "Parameterization" on page $1155 \Leftrightarrow$ Chapter 5.2.6.6.1.12 "Measuring ranges" on page 1166:

| Voltage | $0 \vee \ldots 10 \mathrm{~V}$ | with differential inputs, 2 chan- <br> nels used |
| :--- | :--- | :--- |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | with differential inputs, 2 chan- <br> nels used |

The function of the LEDs is described under Diagnosis and displays / Displays $\Leftrightarrow$ Chapter 5.2.6.6.1.11 "State LEDs" on page 1165.

To avoid error messages from unused analog input channels, configure them as "unused".

## Use of analog inputs as digital inputs

Several (or all) analog inputs can be configured as digital inputs. The inputs are not galvanically isolated against the other analog channels.


Fig. 235: Connection of digital sensors to the analog input AIO (Proceed with the analog inputs Al1 ... Al3 in the same way)
The following measuring ranges can be configured ${ }^{\leftrightarrows}$ Chapter 5.2.6.6.1.9 "Parameterization" on page 1155 出 Chapter 5.2.6.6.1.12 "Measuring ranges" on page 1166:

| Digital input | 24 V | 1 channel used |
| :--- | :--- | :--- |

The function of the LEDs is described under Diagnosis and displays / Displays ${ }^{\mu}$ Chapter 5.2.6.6.1.11 "State LEDs" on page 1165.

## Connection of analog output loads (Voltage)



Fig. 236: Connection of analog output loads (voltage) to the analog output AOO (Proceed with the analog output AO1 in the same way)

The following measuring ranges can be configured ${ }^{\mu}$ Chapter 5.2.6.6.1.9 "Parameterization" on page $1155{ }^{\wedge}>$ Chapter 5.2.6.6.1.12 "Measuring ranges" on page 1166:

| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | Load $\pm 10 \mathrm{~mA}$ max. | 1 channel used |
| :--- | :--- | :--- | :--- |

The function of the LEDs is described under Diagnosis and displays / Displays ${ }^{\mu}$ Chapter 5.2.6.6.1.11 "State LEDs" on page 1165.

Unused analog outputs can be left open-circuited.

## Connection of analog output loads (Current)



Fig. 237: Connection of analog output loads (current) to the analog output AOO (Proceed with the analog output AO1 in the same way)
The following measuring ranges can be configured ${ }^{\wedge} \Rightarrow$ Chapter 5.2.6.6.1.9 "Parameterization" on page 1155 \& Chapter 5.2.6.6.1.12 "Measuring ranges" on page 1166:

| Current | $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | Load $0 \Omega \ldots 500 \Omega$ | 1 channel used |
| :--- | :--- | :--- | :--- |
| Current | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | Load $0 \Omega \ldots 500 \Omega$ | 1 channel used |

The function of the LEDs is described under Diagnosis and displays / Displays ${ }^{\mu}$ Chapter 5.2.6.6.1.10 "Diagnosis" on page 1160.

Unused analog outputs can be left open-circuited.

## Internal data exchange

| Parameter | Value |
| :--- | :--- |
| Digital inputs (bytes) | 3 |
| Digital outputs (bytes) | 3 |
| Analog inputs (words) | 4 |
| Analog outputs (words) | 2 |


| Parameter | Value |
| :--- | :--- |
| Counter input data (words) | 4 |
| Counter output data (words) | 8 |

## Addressing

The module reads the position of the rotary switches only during power-up, i.e. changes of the switch position during operation will have no effect until the next module initialization.

## I/O configuration

The CI541-DP PROFIBUS DP bus configuration is handled by PROFIBUS DP master with the exception of the slave bus ID (via rotary switches) and the transmission rate (automatic detection).

The analog/digital I/O channels and the fast counter are configured via software.
Details about configuration are described in Parameterization ${ }^{\mu} y$ Chapter 5.2.6.6.1.9 "Parameterization" on page 1155.

## Parameterization

## Parameters of the module

Table 252: Parameters of the module:

| Name | Value | Internal value | Internal value, type | Default |
| :---: | :---: | :---: | :---: | :---: |
| Module ID ${ }^{1}$ ) | Internal | 0x1C20 | WORD | 0x1C20 |
| Parameter length | Internal | 47 | BYTE | 47 |
| Reserved (1 byte) | 0 | 0 | BYTE | 0 |
| Error LED / Failsafe function (see ( ${ }^{4}$ Table 253 "Set tings "Error LED Failsafe function"" on page 1156) | On | 0 | BYTE | 0 |
|  | Off by E4 | 1 |  |  |
|  | Off by E3 | 2 |  |  |
|  | On + failsafe | 16 |  |  |
|  | Off by E4 + failsafe | 17 |  |  |
|  | Off by E3 + failsafe | 18 |  |  |
| Reserved (20 bytes) | 0 | 0 | BYTE | 0 |
| Check supply (UP and UP3) | On | 0 | BYTE |  |
|  | Off | 1 |  | 1 |
| Fast counter | 0 | 0 | BYTE | 0 |
|  | : | : |  |  |


| Name | Value | Internal value | Internal value, <br> type | Default |
| :--- | :--- | :--- | :--- | :--- |
|  | $10^{2}$ ) | 10 |  |  |
| 1) With a faulty ID, the Modules reports a "parameter error" and does not perform cyclic <br> process data transmission <br> ${ }^{2}$ ) Counter operating modes, see description of the fast counter $\Leftrightarrow$ Chapter 5.2.4.3.2.9 "Fast <br> counter" on page 606. |  |  |  |  |

Table 253: Settings "Error LED / Failsafe function"

| Setting | Description |
| :--- | :--- |
| On | Error LED (S-ERR) lights up at errors of all error classes, <br> Failsafe mode off |
| Off by E4 | Error LED (S-ERR) lights up at errors of error classes E1, E2 <br> and E3, Failsafe mode off |
| Off by E3 | Error LED (S-ERR) lights up at errors of error classes E1 and <br> E2, Failsafe mode off |
| On +Failsafe | Error LED (S-ERR) lights up at errors of all error classes, <br> Failsafe mode on *) |
| Off by E4 + Failsafe | Error LED (S-ERR) lights up at errors of error classes E1, E2 <br> and E3, Failsafe mode on *) |
| Off by E3 + Failsafe | Error LED (S-ERR) lights up at errors of error classes E1 and <br> E2, Failsafe mode on *) |
| ) The parameters Behaviour analog outputs at communication error and Behaviour digital |  |
|  |  |

## Group parameters for the analog part

| Name | Value | Internal value | Internal value, <br> type | Default |
| :--- | :--- | :--- | :--- | :--- |
| Analog data <br> format | Standard <br> Reserved | R <br> Behaviour analog <br> outputs at com- <br> munication error <br> $*$ | Off | Last value |
| Last value 5 s |  |  |  |  |
| Last value 10 s | 11 | 6 | 0 | BYTE |
| Substitute value | 2 | 0 |  |  |
| Substitute value | 7 |  |  |  |

## Channel parameters for the analog inputs (4x)

General

| Name | Value | Internal value | Internal value, type | Default |
| :---: | :---: | :---: | :---: | :---: |
| Input 0, Channel configuration | Operation modes of analog inputs \& Table 254 "Op eration modes of analog inputs:" on page 1157 | Operation modes of analog inputs « Table 254 "Op eration modes of analog inputs:" on page 1157 | BYTE | 0 |
| Input 0, Check channel | Settings channel monitoring *) Further information on page 1158 | Settings channel monitoring ๕ Further information on page 1158 | BYTE | 0 |
| : | : | : | : |  |
| : | : | : | : | . |
| Input 3, Channel configuration | Operation modes of analog inputs 4 Table 254 "Op eration modes of analog inputs:" on page 1157 | Operation modes of analog inputs 4 Table 254 "Op eration modes of analog inputs:" on page 1157 | BYTE | 0 |
| Input 3, Check channel | Settings channel monitoring ヶ ${ }^{4}$ Further information on page 1158 | Settings channel monitoring « Further information on page 1158 | BYTE | 0 |

## Channel configuration

Table 254: Operation modes of analog inputs:

| Internal value | Operating modes of the analog inputs, individually configurable |
| :---: | :---: |
| 0 (default) | Not used |
| 1 | 0 ... 10 V |
| 2 | Digital input |
| 3 | $0 \mathrm{~mA} . . .20 \mathrm{~mA}$ |
| 4 | $4 \mathrm{~mA} . . .20 \mathrm{~mA}$ |
| 5 | -10 V ... +10 V |
| 8 | 2-wire Pt100-50 ${ }^{\circ} \mathrm{C} . . .+400^{\circ} \mathrm{C}$ |
| 9 | 3-wire Pt100-50 ${ }^{\circ} \mathrm{C} . . .+400^{\circ} \mathrm{C}$ *) |
| 10 | $0 \mathrm{~V} . . .10 \mathrm{~V}$ (voltage diff.) *) |
| 11 | $-10 \mathrm{~V} . . .+10 \mathrm{~V}$ (voltage diff.) *) |
| 14 | 2-wire Pt100-50 ${ }^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ |
| 15 | 3 -wire Pt100-50 ${ }^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ *) |
| 16 | 2-wire Pt1000-50 ${ }^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ |
| 17 | 3-wire Pt1000-50 ${ }^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ *) |
| 18 | 2-wire Ni1000-50 ${ }^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ |


| 19 | 3 -wire Ni1000 $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ *) |
| :--- | :--- |
| ${ }^{*}$ ) In the operating modes with 3-wire configuration or with differential inputs, two adjacent |  |
| analog inputs belong together (e.g. the channels 0 and 1 ). In these cases, both channels |  |
| are configured in the desired operating mode. The lower address must be the even address |  |
| (channel 0 ). The next higher address must be the odd address (channel 1). The converted |  |
| analog value is available at the higher address (channel 1). |  |

## Channel monitoring

Table 255: Table settings channel monitoring:

| Internal Value | Check Channel |
| :--- | :--- |
| 0 (default) | Plausib(ility), cut wire, short circuit |
| 3 | Not used |

Channel parameters for the analog outputs (2x)

| Name | Value | Internal value | Internal value, type | Default |
| :---: | :---: | :---: | :---: | :---: |
| Output 0, <br> Channel configu- <br> ration | Operation modes of analog outputs « Table 256 "Tab le operation modes of analog outputs:" on page 1159 | Operation modes of analog outputs <br> « Table 256 "Tab <br> le operation <br> modes of analog outputs:" <br> on page 1159 | BYTE | 0 |
| Output 0, Check channel | Channel monitoring ③ Table 257" Table channel monitoring:" on page 1159 | Channel monitoring <br> Table 257 " Table channel monitoring:" on page 1159 | BYTE | 0 |
| Output 0, Substitute value | Substitute value « Table 258 " Table substitute value:" on page 1159 | Substitute value « Table 258 " Table substitute value:" on page 1159 | WORD | 0 |
| Output 1, <br> Channel configu- <br> ration | Operation modes of analog outputs <br> * Table 256 "Tab le operation modes of analog outputs:" on page 1159 | Operation modes of analog outputs <br> * Table 256 "Tab le operation modes of analog outputs:" on page 1159 | BYTE | 0 |
| Output 1, Check channel | Channel monitoring <br> (4) Table 257" Table channel monitoring:" on page 1159 | Channel monitoring <br> Table 257 " Table channel monitoring:" on page 1159 | BYTE | 0 |
| Output 1, Substitute value | Substitute value Table 258 " Table substitute value:" on page 1159 | Substitute value <br> ③ Table 258 " <br> Table substitute value:" <br> on page 1159 | WORD | 0 |

Channel config- Table 256: Table operation modes of analog outputs: uration

| Internal value | Operating modes of the analog outputs, <br> individually configurable |
| :--- | :--- |
| 0 (default) | Not used |
| 128 | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ |
| 129 | $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ |
| 130 | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ |

Channel moni- Table 257: Table channel monitoring:
toring

| Internal value | Check channel |
| :--- | :--- |
| 0 | Plausib(ility), cut wire, short circuit |
| 3 | None |

Substitute value Table 258: Table substitute value:

| Intended behavior of output <br> channel when the control <br> system stops | Required setting of the <br> module parameter "Behav- <br> iour of outputs in case of a <br> communication error" | Required setting of the <br> channel parameter "Substi- <br> tute value" |
| :--- | :--- | :--- |
| Output OFF | Off | 0 |
| Last value infinite | Last value | 0 |
| Last value for 5 s and then <br> turn off | Last value 5 sec | 0 |
| Last value for 10 s and then <br> turn off | Last value 10 sec | 0 |
| Substitute value infinite | Substitute value | depending on configuration |
| Substitute value for 5 s and <br> then turn off | Substitute value 5 sec | depending on configuration |
| Substitute value for 10 s and <br> then turn off | Substitute value 10 sec | depending on configuration |

## Group parameters for the digital part

| Name | Value | Internal value | Internal value, <br> type | Default |
| :--- | :--- | :--- | :--- | :--- |
| Input delay | 0.1 ms | 0 | BYTE | 0.1 ms |
|  | 1 ms | 1 |  | $0 \times 00$ |
|  | 8 ms | 2 | 3 |  |
| Detect short cir- <br> cuit at outputs | 32 ms | Off | On | 1 |


| Name | Value | Internal value | Internal value, type | Default |
| :---: | :---: | :---: | :---: | :---: |
| Behaviour digital outputs at communcation error ${ }^{1}$ ) | Off <br> Last value <br> Last value 5 sec <br> Last value 10 sec <br> Substitute value <br> Substitute value <br> 5 sec <br> Substitute value 10 sec | $\begin{aligned} & \hline 0 \\ & 1 \\ & 6 \\ & 11 \\ & 2 \\ & 7 \\ & 12 \end{aligned}$ | BYTE | $\begin{array}{l\|l\|} \hline \text { Off } \\ 0 \times 00 \end{array}$ |
| Substitute value at output | $0 \ldots 255$ | 00h ... FFh | BYTE | $\begin{aligned} & \hline 0 \\ & 0 \times 00 \end{aligned}$ |
| Detect voltage overflow at outputs ${ }^{2}$ ) | $\begin{aligned} & \text { Off } \\ & \text { On } \end{aligned}$ | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | BYTE | $\begin{array}{\|l\|} \hline \text { Off } \\ 0 \times 00 \end{array}$ |

${ }^{1}$ ) The parameters Behaviour digital outputs at communcation error is only analyzed if the Failsafe-mode is ON.
${ }^{2}$ ) The state "externally voltage detected" appears, if the output of a channel DC0 ... DC7 should be switched on while an externally voltage is connected ${ }^{\xi}$ Chapter 5.2.6.6.1.5 "Connections" on page 1139. In this case the start up is disabled, as long as the externally voltage is connected. The monitoring of this state and the resulting diagnosis message can be disabled by setting the parameters to "OFF".

## Diagnosis

Structure of the diagnosis block via DPM_SLV_DIAG function block.

| Byte Number | Description | Possible Values |
| :--- | :--- | :--- |
| 1 | Data length (header included) | 7 |
| 2 | PROFIBUS DP V1 coding: <br> Vendor specific | 129 |
| 3 | Diagnosis Byte, slot number | $31=\mathrm{CI} 541-\mathrm{DP}$ (e. g. error at integrated 8 <br> $\mathrm{DI} / 8 \mathrm{DO})$ <br> $1=1$ st connected S500 I/O Module <br> $\ldots$ |
| 4 | Diagnosis Byte, module <br> number | According to the I/O bus specification <br> passed on by modules to the fieldbus <br> master |
| 5 | Diagnosis Byte, channel | According to the I/O bus specification <br> passed on by modules to the fieldbus <br> master |


| Byte Number | Description | Possible Values |
| :--- | :--- | :--- |
| 6 | Diagnosis Byte, error code | According to the I/O bus specification |
|  |  | Bit 7 and bit 6, coded error class |
|  |  | $0=\mathrm{E} 1$ |
|  |  | $1=\mathrm{E} 2$ |
|  |  | $2=\mathrm{E} 3$ |
|  | $3=\mathrm{E} 4$ |  |
|  |  | Bit 0 to bit 5, coded error description |
| 7 | Diagnosis Byte, flags | According to the I/O bus specification |
|  |  | Bit 7: 1 = coming error |
|  | Bit 6: $1=$ leaving error |  |

In cases of short circuit or overload, the digital outputs are turned off. The modules performs reactivation automatically. Thus an acknowledgement of the errors is not necessary. The error message is stored via the LED.

| E1 ... E4 | d1 | d2 | d3 | d4 | Identifier 000 ... | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err |  |  |
| $\begin{array}{\|l} \hline \text { Byte } 6 \\ \text { Bit } \\ 6 \text {... } 7 \end{array}$ | - | Byte 3 | Byte 4 | Byte 5 | Byte 6 <br> Bit <br> 0 ... 5 |  |  |
| Class | Interface | Device | Module | Channel | Error- <br> Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) |  |  |  |  |
| Module errors |  |  |  |  |  |  |  |
| 3 | - | 31 | 31 | 31 | 19 | Checksum error in the I/O module | Replace I/O module |
| 3 | - | 31 | 31 | 31 | 3 | Timeout in the I/O module |  |
| 3 | - | 31 | 31 | 31 | 40 | Different hard-/firmware versions in the module |  |
| 3 | - | 31 | 31 | 31 | 43 | Internal error in the module |  |
| 3 | - | 31 | 31 | 31 | 36 | Internal data exchange failure |  |
| 3 | - | 31 | 31 | 31 | 9 | Overflow diagnosis buffer | Restart |
| 3 | - | 31 | 31 | 31 | 26 | Parameter error | Check master |


| E1 ... E4 | d1 | d2 | d3 | d4 | Identifier 000 ... 0 | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err |  |  |
| $\begin{array}{\|l} \hline \text { Byte } 6 \\ \text { Bit } \\ 6 \text {... } 7 \end{array}$ | - | Byte 3 | Byte 4 | Byte 5 | Byte 6 <br> Bit <br> 0 ... 5 |  |  |
| Class | $\begin{aligned} & \text { Inter- } \\ & \text { face } \end{aligned}$ | Device | Module | Channel | Error- <br> Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) |  |  |  |  |
| 3 | - | 31 | 31 | 31 | 11 | Process voltage UP too low | Check process supply voltage |
| 3 | - | 31 | 31 | 31 | 45 | Process voltage UP gone | Check process supply voltage |
| 3 | - | 31/1... 10 | 31 | 31 | 17 | No communication with I/O device | Replace I/O module |
| 3 | - | 1... 10 | 31 | 31 | 32 | Wrong I/O module type on socket | Replace I/O module / Check configuration |
| 4 | - | 1... 10 | 31 | 31 | 31 | At least one module does not support failsafe function | Check modules and parameterization |
| 4 | - | 1... 10 | 31 | 5 | 8 | I/O module removed from hot swap terminal unit or defective module on hot swap terminal unit ${ }^{9}$ ) | Plug I/O module, replace I/O module |
| 4 | - | 1... 10 | 31 | 5 | 28 | Wrong I/O module plugged on hot swap terminal unit ${ }^{9}$ ) | Remove wrong I/O module and plug projected I/O module |
| 4 | - | 1... 10 | 31 | 5 | 42 | No communication with I/O module on hot swap terminal unit ${ }^{9}$ ) | Replace I/O module |


| E1 ... E4 | d1 | d2 | d3 | d4 | Identi fier 000 ... | AC500Display | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err | $\begin{aligned} & \text { PS501 } \\ & \text { PLC } \\ & \text { Browser } \end{aligned}$ |  |  |
| $\begin{aligned} & \hline \text { Byte } 6 \\ & \text { Bit } \\ & 6 \ldots 7 \end{aligned}$ | - | Byte 3 | Byte 4 | Byte 5 | $\begin{array}{\|l} \hline \text { Byte } 6 \\ \text { Bit } \\ 0 \text {... } 5 \end{array}$ | PROFIB US DP diagnosis block |  |  |
| Class | $\begin{aligned} & \text { Inter- } \\ & \text { face- } \end{aligned}$ | Device | Module | Channel | Error- <br> Identifier | Error message |  | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) |  |  |  |  |  |
| 4 | - | 1... 10 | 31 | 5 | 54 | I/O module does not support hot swap $\left.{ }^{8}\right)^{9}$ ) |  | Power <br> off <br> system <br> and <br> replace <br> $1 /$ <br> module |
| 4 | - | 1... 10 | 31 | 6 | 8 | Hot swap terminal unit configured but not found |  | Replace terminal unit by hot swap terminal unit |
| 4 | - | 1... 10 | 31 | 6 | 42 | No communication with hot swap terminal unit ${ }^{9}$ ) |  | Restart, if error persists replace terminal unit |
| 4 | - | 31 | 31 | 31 | 46 | Reverse voltage from digital outputs DO0...DO7 to UP3 ${ }^{4}$ ) |  | Check connection |
| 4 | - | 31/1... 10 | 31 | 31 | 34 | No response during initialization of the I/O module |  | Replace I/O module |
| 4 | - | 31 | 31 | 31 | 11 | Process voltage UP3 too low |  | Check process supply voltage |
| 4 | - | 31 | 31 | 31 | 45 | Process voltage UP3 gone |  | Check process supply voltage |
| 4 | - | 31 | 31 | 31 | 10 | Voltage overflow on outputs (above UP3 level) ${ }^{5}$ ) |  | Check terminals/ check process supply voltage |


| E1 ... E4 | d1 | d2 | d3 | d4 | Identifier $000 \ldots$ | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err |  |  |
| $\begin{array}{\|l} \hline \text { Byte } 6 \\ \text { Bit } \\ 6 \ldots 7 \end{array}$ | - | Byte 3 | Byte 4 | Byte 5 | $\begin{array}{\|l\|} \hline \text { Byte } 6 \\ \text { Bit } \\ 0 \text {... } 5 \end{array}$ |  |  |
| Class | Interface | Device | Module | Channel | Error- <br> Identi- <br> fier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | $\left.{ }^{3}\right)$ |  |  |  |  |
| Channel error digital |  |  |  |  |  |  |  |
| 4 | - | 31 | 2 | 0...7 | 46 | Externally voltage detected on digital output DO0...DO7 ${ }^{6}$ ) | Check terminals |
| 4 | - | 31 | 2 | 0... 7 | 47 | Short circuit at digital output ${ }^{7}$ ) | Check terminals |
| Channel error analog |  |  |  |  |  |  |  |
| 4 | - | 31 | 1 | 0...3 | 48 | Analog value overflow or broken wire at an analog input | Check value or check terminals |
| 4 | - | 31 | 1 | 0... 3 | 7 | Analog value underflow at an analog input | Check value |
| 4 | - | 31 | 1 | 0... 3 | 47 | Short-circuit at an analog input | Check terminals |
| 4 | - | 31 | 3 | 0... 1 | 4 | Analog value overflow at an analog output | Check output value |
| 4 | - | 31 | 3 | 0... 1 | 7 | Analog value underflow at an analog output | Check output value |

Remarks:

| $\left.{ }^{1}\right)$ | In AC500 the following interface identifier applies: <br> "-" = Diagnosis via bus-specific function blocks; $0 \ldots 4$ or $10=$ Position of <br> the Communication Module; 14 = I/O bus; 31 = Module itself <br> The identifier is not contained in the CI541-DP diagnosis block. |
| :--- | :--- |
| $\left.{ }^{2}\right)$ | With "Device" the following allocation applies: $31=$ Module itself; $1 \ldots . .10=$ <br> Expansion module |
| $\left.{ }^{3}\right)$ | With "Module" the following allocation applies: <br> $31=$ Module itself <br> Channel error: Module type ( $1=$ AI, $2=$ DO, $3=$ AO $)$ |


| ${ }^{4}$ ) | This message appears, if externally voltages at one or more terminals DO0...DO7 cause that other digital outputs are supplied through that voltage (voltage feedback, see description in section 'Connection' \# Chapter 5.2.6.6.1.5 "Connections" on page 1139). All outputs of the apply digital output groups will be turned off for 5 seconds. The diagnosis message appears for the whole output group. |
| :---: | :---: |
| ${ }^{5}$ ) | The voltage on digital outputs DOO ... DO7 has overrun the process supply voltage UP3 (see description in section 'Connection' ${ }^{*}$ Chapter 5.2.6.6.1.5 "Connections" on page 1139). Diagnosis message appears for the whole module. |
| ${ }^{6}$ ) | This message appears, if the output of a channel DO0 ... DO7 should be switched on while an externally voltage is connected. In this case the start up is disabled, as long as the externally voltage is connected. Otherwise this could produce reverse voltage from this output to other digital outputs. This diagnosis message appears per channel. |
| ${ }^{7}$ ) | Short circuit: After a detected short circuit, the output is deactivated for 100 ms . Then a new start up will be executed. This diagnosis message appears per channel. |
| ${ }^{8}$ ) | In case of an I/O module doesn't support hot swapping, do not perform any hot swap operations (also not on any other terminal units (slots)) as modules may be damaged or I/O bus communication may be disturbed. |
| ${ }^{9}$ ) | Diagnosis for hot swap available as of version index F0. |

## State LEDs

The LEDs are located at the front of module. There are 2 different groups:

- The 5 system LEDs (PWR, STA1-DP, STA2-DP, S-ERR and I/O-Bus) show the operation state of the module and display possible errors.
- The 27 process LEDs (UP, UP3, inputs, outputs, CH-ERR1 to CH-ERR3) show the process supply voltage and the states of the inputs and outputs and display possible errors.

Table 259: States of the 5 system LEDs:

| LED | Color | OFF | ON | Flashing |
| :--- | :--- | :--- | :--- | :--- |
| PWR/RUN | Green | Process supply <br> voltage missing | Internal supply <br> voltage OK, <br> module ready for <br> communication <br> with I/O Con- <br> troller | Start-up / pre- <br> paring communi- <br> cation |
|  |  | Yellow | --- | --- |
| STA1-DP | Green | --- | PROFIBUS run- <br> ning | Invalid device <br> parameters |
| STA2-DP | Red | No error | Bus timeout | No communica- <br> tion to master |
| S-ERR | Red | No error | Internal error | -- |
| I/O-Bus | Green | No communica- <br> tion interface <br> modules con- <br> nected or com- <br> munication error | Communication <br> interface <br> modules con- <br> nected and <br> operational | --- |

Table 260: States of the 27 process LEDs:

| LED | Color | OFF | ON | Flashing |
| :--- | :--- | :--- | :--- | :--- |
| AIO to AI3 | Yellow | Input is OFF | Input is ON <br> (brightness <br> depends on the <br> value of the <br> analog signal) | -- |
| AO0 to AO1 | Yellow | Output is OFF | Output is ON <br> (brightness <br> depends on the <br> value of the <br> analog signal) | -- |
| DI0 to DI7 | Yellow | Input is OFF | Input is ON (the <br> input voltage is <br> even displayed if <br> the supply <br> voltage is OFF) | -- |
| DO0 toDO7 | Yellow | Green | Output is OFF <br> Process supply <br> voltage missing | Output is ON <br> Poltage OK and <br> initialization fin- <br> ished |
| UP | Green | Process supply <br> voltage missing | Process supply <br> voltage OK | --- |
| UP3 | No error or <br> process supply <br> voltage missing | Internal error <br> CH-ERR1 to CH- <br> ERR3 | Red | Error on one <br> channel of the <br> corresponding <br> group |

## Measuring ranges

Input ranges voltage, current and digital input

| Range | $\begin{aligned} & \mathrm{O} \text { V } \ldots+10 \\ & \mathrm{~V} \end{aligned}$ | $\begin{aligned} & -10 \mathrm{~V} \ldots \\ & +10 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 0 \mathrm{~mA} . . \\ & 20 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 4 \mathrm{~mA} \ldots 20 \\ & \mathrm{~mA} \end{aligned}$ | Digital input | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  | Decimal | Hex. |
| Overflow | > 11.7589 | > 11.7589 | > 23.5178 | > 22.8142 |  | 32767 | 7FFF |
| Measured value too high |  |  | 23.5178 $:$ 20.0007 |  |  | $\begin{aligned} & 32511 \\ & \vdots \\ & 27649 \end{aligned}$ | $\begin{aligned} & \text { 7EFF } \\ & : \\ & 6 \mathrm{C} 01 \end{aligned}$ |
| Normal range <br> Normal range or measured value too low | 10.0000 $:$ 0.0004 | $\begin{aligned} & \hline 10.0000 \\ & : \\ & 0.0004 \\ & \hline \end{aligned}$ | 20.0000 $:$ 0.0007 | 20.0000 $:$ 4.0006 | On | 27648 | $\begin{array}{\|l} \hline 6 \mathrm{C} 00 \\ : \\ 0001 \\ \hline \end{array}$ |
|  | 0.0000 | 0.0000 | 0 | 4 | Off | 0 | 0000 |
|  | $\begin{array}{\|l\|} \hline-0.0004 \\ -1.7593 \end{array}$ | $\begin{aligned} & -0.0004 \\ & : \\ & : \\ & -10,0000 \end{aligned}$ |  | $\begin{aligned} & 3.9994 \\ & 1.1858 \end{aligned}$ |  | -1 <br> -4864 -27648 | $\begin{aligned} & \text { FFFF } \\ & \text { ED00 } \\ & : \\ & 9400 \end{aligned}$ |


| Range | $0 \text { V ... +10 }$ | -10 V ... | $0 \mathrm{~mA} \ldots$ | $4 \mathrm{~mA} \ldots 20$ | Digital | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  | Decimal | Hex. |
| Measured value too low |  | $\begin{array}{\|l} \hline-10.0004 \\ : \\ -11.7589 \end{array}$ |  |  |  | $\begin{aligned} & -27649 \\ & : \\ & -32512 \end{aligned}$ | $\begin{aligned} & \text { 93FF } \\ & \text { : } \\ & 8100 \end{aligned}$ |
| Underflow | < 1.7593 | <-11.7589 | $<0.0000$ | < 1.1858 |  | -32768 | 8000 |

The represented resolution corresponds to 16 bits.

Input ranges resistance temperature detector

| Range | $\begin{aligned} & \text { Pt100 / Pt1000 } \\ & -50 \ldots+70^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \text { Pt100 / Pt1000 } \\ & -50 \ldots+400^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \mathrm{Ni} 1000 \\ & -50 \ldots+150^{\circ} \mathrm{C} \end{aligned}$ |
| :---: | :---: | :---: | :---: |
| Overflow | $>+80.0^{\circ} \mathrm{C}$ | $>+450.0{ }^{\circ} \mathrm{C}$ | $>+160.0^{\circ} \mathrm{C}$ |
| Measured value too high |  | $\begin{aligned} & +450.0^{\circ} \mathrm{C} \\ & + \\ & +400.1^{\circ} \mathrm{C} \end{aligned}$ |  |
|  |  |  | $\begin{aligned} & +160.0^{\circ} \mathrm{C} \\ & : \\ & +150.1^{\circ} \mathrm{C} \end{aligned}$ |
| Normal range |  | $\begin{aligned} & +400.0^{\circ} \mathrm{C} \\ & : \\ & : \\ & : \\ & +0.1^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & +150.0^{\circ} \mathrm{C} \\ & : \\ & \vdots \\ & +0.1^{\circ} \mathrm{C} \end{aligned}$ |
|  |  | $0.0{ }^{\circ} \mathrm{C}$ | $0.0{ }^{\circ} \mathrm{C}$ |
|  |  | $\begin{aligned} & -0.1^{\circ} \mathrm{C} \\ & : \\ & -50.0^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & -0.1^{\circ} \mathrm{C} \\ & : \\ & -50.0^{\circ} \mathrm{C} \end{aligned}$ |
| Measured value too low |  | $\begin{aligned} & -50.1^{\circ} \mathrm{C} \\ & : \\ & -60.0^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & -50.1^{\circ} \mathrm{C} \\ & : \\ & -60.0^{\circ} \mathrm{C} \end{aligned}$ |
| Underflow | $<-60.0{ }^{\circ} \mathrm{C}$ | $<-60.0{ }^{\circ} \mathrm{C}$ | $<-60.0^{\circ} \mathrm{C}$ |


| Range | Digital value | Hex. |
| :--- | :--- | :--- |
|  | Decimal | 7FFF |
| Overflow | 32767 | 1194 |
| Measured value too high | 4500 | $:$ |
|  | $:$ | 0FA1 |


| Range | Digital value |  |
| :---: | :---: | :---: |
|  | Decimal | Hex. |
|  | 1600 | 0640 |
|  |  |  |
|  | 1501 | 05DD |
|  | 800 | 0320 |
|  |  |  |
|  | 701 | O2BD |
| Normal range | 4000 | OFAO |
|  | 1500 | 05DC |
|  | 700 | 02BC |
|  | : |  |
|  | 1 | 0001 |
|  | 0 | 0000 |
|  | -1 | FFFF |
|  | : |  |
|  | -500 | FEOC |
| Measured value too low | -501 | FE0B |
|  | : |  |
|  | -600 | FDA8 |
| Underflow | -32768 | 8000 |

## Output ranges voltage and current

| Range | -10...+10 V | 0... 20 mA | $4 . .20 \mathrm{~mA}$ |
| :---: | :---: | :---: | :---: |
| Overflow | >11.7589 V | >23.5178 mA | >22.8142 mA |
| Measured value too high | $\begin{aligned} & 11.7589 \mathrm{~V} \\ & : \\ & 10.0004 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 23.5178 \mathrm{~mA} \\ & : \\ & 20.0007 \mathrm{~mA} \\ & \hline \end{aligned}$ | $\begin{aligned} & 22.8142 \mathrm{~mA} \\ & : \\ & 20.0006 \mathrm{~mA} \end{aligned}$ |
| Normal range | $\begin{aligned} & 10.0000 \mathrm{~V} \\ & : \\ & 0.0004 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 20.0000 \mathrm{~mA} \\ & : \\ & 0.0007 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 20.0000 \mathrm{~mA} \\ & : \\ & 4.0006 \mathrm{~mA} \end{aligned}$ |
|  | 0.0000 V | 0.0000 mA | 4.0000 mA |
|  | $\begin{aligned} & -0.0004 \mathrm{~V} \\ & : \\ & -10.0000 \mathrm{~V} \end{aligned}$ |  | $\begin{aligned} & 3.9994 \mathrm{~mA} \\ & 0 \mathrm{~mA} \\ & 0 \mathrm{~mA} \end{aligned}$ |
| Measured value too low | $\begin{aligned} & \hline-10.0004 \mathrm{~V} \\ & : \\ & -11.7589 \mathrm{~V} \\ & \hline \end{aligned}$ | $\begin{aligned} & \hline 0 \mathrm{~mA} \\ & : \\ & 0 \mathrm{~mA} \\ & \hline \end{aligned}$ |  |
| Underflow | 0 V | 0 mA | 0 mA |


| Range | Digital value |  |
| :--- | :--- | :--- |
|  | Decimal | Hex. |
| Overflow | $>32511$ | $>7$ EFF |
| Measured value too high | 32511 | 7EFF |
|  | $:$ | 67649 |
| Normal range | 27648 | 6 C01 |
|  | 1 | 6 C00 |
|  | 0 | $:$ |
|  | -1 | 0001 |
|  | -6912 | 0000 |
|  | -27648 | FFFF |
|  | -27649 | 9400 |
| Underflow | $:$ | $93 F F$ |

The represented resolution corresponds to 16 bits.

## Technical data

## Technical data of the module

The system data of AC500 and S500 are applicable to the standard version ${ }^{\wedge} \Rightarrow$ Chapter 5.1.2 "System data AC500" on page 166.
The system data of AC500-XC are applicable to the XC version ${ }^{\leftrightarrows}$ Chapter 5.1.3 "System data AC500-XC" on page 169.
Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter |  | Value |
| :--- | :--- | :--- |
| Process supply voltages UP/UP3 |  |  |
|  | Rated value | 24 V DC (for inputs and outputs) |
|  | Max. load for the terminals | 10 A |
|  | Protection against reversed voltage | Yes |
|  | Rated protection fuse on UP/UP3 | 10 A fast |
|  | Galvanic isolation | PROFIBUS interface against the rest of the <br> module |
|  | Inrush current from UP (at power up) | On request |
|  | Current consumption via UP (normal <br> operation) | 0.2 A |
|  | Current consumption via UP3 | $0.06 \mathrm{~A}+0.5 \mathrm{~A}$ max. per output |


| Parameter | Value |
| :---: | :---: |
| Connections | Terminals 2.8 and 3.8 for +24 V (UP) <br> Terminal 4.8 for +24 V (UP3) <br> Terminals 2.9, 3.9 and 4.9 for $0 \mathrm{~V}(\mathrm{ZP})$ |
| Max. power dissipation within the module | 6 W |
| Configurable digital inputs/outputs | 8 |
| Number of digital inputs | 8 |
| Number of digital outputs | 8 |
| Reference potential for all digital inputs and outputs | Negative pole of the supply voltage, signal name ZP |
| Setting of the PROFIBUS DP identifier | With 2 rotary switches at the front side of the module |
| Diagnose | See Diagnosis ${ }^{〔}$ Chapter 5.2.6.6.1.10 "Diagnosis" on page 1160 |
| Operation and error displays | 32 LEDs (totally) |
| Weight (without terminal unit) | Ca. 125 g |
| Mounting position | Horizontal <br> Or vertical with derating (output load reduced to $50 \%$ at $+40^{\circ} \mathrm{C}$ per group) |
| Cooling | The natural convection cooling must not be hindered by cable ducts or other parts in the control cabinet. |

## NOTICE!

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and temporary overvoltage up to 30 V DC.

## Multiple overloads

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an internal smart high-side switch.

## Technical data of the digital inputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 8 |
| Distribution of the channels into groups | 1 group of 8 channels |
| Terminals of the channels DIO ... DI7 | Terminals $3.0 \ldots 3.7$ |
| Reference potential for all inputs | Terminals $2.9 \ldots 4.9$ (negative pole of the <br> supply voltage, signal name ZP) |
| Indication of the input signals | 1 yellow LED per channel, the LED is ON when <br> the input signal is high (signal 1$)$ |
| Input type (according EN 61131-2) | Type 1 |
| Input delay (0->1 or 1->0) | Typ. 0.1 ms, configurable from $0.1 \mathrm{~ms} \ldots 32 \mathrm{~ms}$ |


| Parameter |  | Value |
| :--- | :--- | :--- |
| Input signal voltage | 24 V DC |  |
|  | 0 -Signal | $-3 \mathrm{~V} \ldots+5 \mathrm{~V}$ |
|  | Undefined Signal | $>+5 \mathrm{~V} \ldots<+15 \mathrm{~V}$ |
|  | 1-Signal | $+15 \mathrm{~V} \ldots+30 \mathrm{~V}$ |
| Ripple with signal 0 | Within $-3 \mathrm{~V} \ldots+5 \mathrm{~V}$ |  |
| Ripple with signal 1 | Within $+15 \mathrm{~V} \ldots+30 \mathrm{~V}$ |  |
| Input current per channel |  |  |
|  | Input voltage +24 V | Typ. 5 mA |
|  | Input voltage +5 V | $>1 \mathrm{~mA}$ |
|  | Input voltage +15 V | $>2 \mathrm{~mA}$ |
|  | Input voltage +30 V | $<8 \mathrm{~mA}$ |
| Max. cable length |  |  |
|  | Shielded | 1000 m |
|  | Unshielded | 600 m |

## Technical data of the digital outputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 8 |
| Distribution of the channels into groups | 1 group of 8 channels |
| Terminals of the channels DO0 ... DO7 | Terminals 4.0 ... 4.7 |
| Reference potential for all outputs | Terminals $2.9 \ldots 4.9$ (negative pole of the <br> supply voltage, signal name ZP) |
| Common power supply voltage | For all outputs terminal 4.8 (positive pole of the <br> supply voltage, signal name UP3) |
| Output voltage for signal 1 | UP3 (-0.8 V) |
| Output delay (0->1 or 1->0) | On request |
| Output current | 500 mA at UP3 = 24 V |
|  | Rated value per channel |
| Max. value (all channels together) | 4 A |
| Leakage current with signal 0 | $<0.5$ mA |
|  | Fuse for UP3 |
| Demagnetization with inductive DC load | Via internal varistors (see figure below this <br> table) |
| Output switching frequency |  |
|  | With resistive load |
| With inductive loads | On request |
|  | With lamp loads |
| Short-circuit-proof / overload-proof | Max. 0.5 Hz |
| Overload message (I > 0.7 A) | 11 Hz max. at 5 W max. |


| Parameter | Value |
| :--- | :--- |
| Output current limitation | Yes, automatic reactivation after short cir- <br> cuit/overload |
| Resistance to feedback against 24 V signals | Yes (software-controlled supervision) |
| Max. cable length |  |
|  | Shielded |
|  | Unshielded |

The module provides several diagnosis functions ${ }^{\circledR} \Rightarrow$ Chapter 5.2.6.6.1.10 "Diagnosis" on page 1160.

The following drawing shows the circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off.


1 Digital output
2 Varistors for demagnetization when inductive loads are turned off

## Technical data of the analog inputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 4 |
| Distribution of channels into groups | 1 group with 4 channels |
| Connection if channels Al0+ ... Al3+ | Terminals $2.0 \ldots 2.3$ |
| Reference potential for Al0+ ... Al3+ | Terminal 2.4 (AI-) for voltage and RTD meas- <br> urement <br> Terminal $2.9,3.9$ and 4.9 for current measure- <br> ment |
| Input type | Unipolar <br> Bipolar <br> Valtage $0 \mathrm{~V} \ldots 10 \mathrm{~V}$, current or Pt100/Pt1000/ <br> Ni1000 |
| Configurability | Voltage $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ |
| Channel input resistance | Against PROFIBUS <br> Pt100/1000, Ni1000 (each input can be config- <br> ured individually) |
| Time constant of the input filter | Voltage: $>100 \mathrm{k} \Omega$ <br> Current: ca. $330 \Omega$ |
| Indication of the input signals | Voltage: $100 \mu \mathrm{~s}$ <br> Current: $100 ~ \mu \mathrm{~s}$ |
|  | 1 LED per channel (brightness depends on the <br> value of the analog signal) |


| Parameter | Value |
| :--- | :--- |
| Conversion cycle | 1 ms (for 4 inputs + 2 outputs); with RTDs Pt/ <br> $\mathrm{Ni} . . .1 \mathrm{~s}$ |
| Resolution | Range $0 \mathrm{~V} \ldots 10 \mathrm{~V}: 12$ bits <br> Range $-10 \mathrm{~V} \ldots+10 \mathrm{~V}: 12$ bits including sign <br> Range $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}: 12$ bits <br> Range $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}: 12$ bits <br> Range RTD (Pt100, PT1000, Ni1000): +0.1 ${ }^{\circ} \mathrm{C}$ |
| Conversion error of the analog values <br> caused by non-linearity, adjustment error <br> at factory and resolution within the normal <br> range | Typ. 0.5 \%, max. $1 \%$ |
| Relationship between input signal and hex <br> code | Tables Input Ranges Voltage, Current and Dig- <br> ital Input and Input range resistance temper- <br> ature detector $\_$Chapter 5.2.6.6.1.12 "Meas- <br> uring ranges" on page 1166 |
| Unused inputs | Are configured as "unused" (default value) |
| Overvoltage protection | Yes |

Technical data of the analog inputs if used as digital inputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | Max. 4 |
| Distribution of channels into groups | 1 group of 4 channels |
| Connections of the channels $\mathrm{AlO}+\ldots \mathrm{Al3+}$ | Terminals 2.0 ... 2.3 |
| Reference potential for the inputs | Terminals 2.9, 3.9 and 4.9 (ZP) |
| Indication of the input signals | 1 LED per channel |
| Input signal voltage | 24 V DC |
| Signal 0 | -30 V ... +5 V |
| Undefined signal | +5 V ... +15 V |
| Signal 1 | +15 V ... +30 V |
| Input current per channel |  |
| Input voltage +24 V | Typ. 7 mA |
| Input voltage +5 V | Typ. 1.4 mA |
| Input voltage +15 V | Typ. 3.7 mA |
| Input voltage +30 V | < 9 mA |
| Input resistance | Ca. $3.5 \mathrm{k} \Omega$ |

## Technical data of the analog outputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 2 |
| Distribution of channels into groups | 1 group for 2 channels |


| Parameter | Value |
| :---: | :---: |
| Connection of the channels $\mathrm{AO}+{ }^{\text {+ }}$. $\mathrm{AO} 1+$ | Terminals 2.5 ... 2.6 |
| Reference potential for $\mathrm{AO} 0+\ldots \mathrm{AO}+$ | Terminal 2.7 (AO-) for voltage output Terminal 2.9, 3.9 and 4.9 for current output |
| Output type |  |
| Unipolar | Current |
| Bipolar | Voltage |
| Galvanic isolation | Against PROFIBUS |
| Configurability | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}, 0 \mathrm{~mA} . .20 \mathrm{~mA}, 4 \mathrm{~mA} . . .20 \mathrm{~mA}$ (each output can be configured individually) |
| Output resistance (load), as current output | $0 \Omega \ldots 500 \Omega$ |
| Output loadability, as voltage output | $\pm 10 \mathrm{~mA}$ max. |
| Indication of the output signals | 1 LED per channel (brightness depends on the value of the analog signal) |
| Resolution | 12 bits including sign |
| Settling time for full range change (resistive load, output signal within specified tolerance) | Typ. 5 ms |
| Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range | Typ. 0.5 \%, max. 1 \% |
| Relationship between input signal and hex code | Table Output Ranges Voltage and Current ⓨ Chapter 5.2.6.6.1.12.3 "Output ranges voltage and current" on page 1168 |
| Unused outputs | Are configured as "unused" (default value) and can be left open-circuited |

## Technical data of the fast counter

| Parameter | Value |
| :--- | :--- |
| Used inputs | Terminal 3.0 (DIO), 3.1 (DI1) |
| Used outputs | Terminal 4.0 (DO0) |
| Counting frequency | Depending on operation mode: <br>  <br>  <br>  <br>  <br>  <br>  <br>  <br>  <br>  <br>  <br>  <br>  <br>  <br>  Mode 1-6: 7ax. max. 200 kHz |
| Mode 10: max. 35 kHz |  |

³ Chapter 6.8.2.12 "Fast counters in AC500 devices" on page 4536

## Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm

The dimensions are in mm and in brackets in inch.

## Ordering data

| Ordering No. | Scope of delivery | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 224 100 R0001 | CI541-DP, PROFIBUS DP communi- <br> cation interface module, 8 DI, 8 DO, <br> 4 Al and 2 AO | Active |
| 1SAP 424 100 R0001 | CI541-DP-XC, PROFIBUS DP com- <br> munication interface module, 8 DI, <br> 8 DO, 4 AI and 2 AO, XC version | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.6.6.2 CI542-DP

Features

- 8 digital inputs 24 V DC
- 8 digital outputs $24 \mathrm{~V} \mathrm{DC}, 0.5 \mathrm{~A}$ max.
- 8 configurable digital inputs/outputs $24 \mathrm{~V} D C, 0.5 \mathrm{~A}$ max.
- Module-wise galvanically isolated
- Fast counter
- XC version for usage in extreme ambient conditions available


1 I/O bus
2 Allocation between terminal number and signal name
38 yellow LEDs to display the signal states of the configurable digital inputs/outputs (DC0 ... DC7)
48 yellow LEDs to display the signal states of the digital inputs (DI8 ... DI15)
58 yellow LEDs to display the signal states of the digital outputs (DO8 ... DO15)
62 green LEDs to display the process supply voltage UP and UP3
73 red LEDs to display errors (CH-ERR1, CH-ERR2, CH-ERR3)
85 system LEDs: PWR/RUN, STA1 DP, STA2 DP, S-ERR, I/O-Bus
9 Label
102 rotary switches for setting the PROFIBUS ID
11 9-pin D-SUB connector to connect the PROFIBUS DP signals
12 Terminal unit
13 DIN rail
${ }_{x}^{*}+{ }_{*}^{*}$ Sign for XC version

## Intended purpose

The PROFIBUS DP communication interface module is used as decentralized I/O module in PROFIBUS networks. Depending on the used terminal unit the network connection is performed either via 9-pole female D-sub or via 10 terminals (screw-type or spring terminals) which are integrated in the terminal unit.
The inputs/outputs are galvanically isolated from the PROFIBUS network. There is no potential separation between the channels. The configuration of the configurable digital inputs/outputs is performed by software.
For usage in extreme ambient conditions (e.g. wider temperature and humidity range), a special XC version of the device is available.

## Diagnosis settings

The current CI54x does not run in combination with a V3 PLC if in the "General" tab the parameter "Diagnosis behavior" is set to "AC500 V3 compatible". How to change the setting in your $A B$ project is described below.


1. Double click in the "Device" tree on "Cl541_IO". $\Rightarrow$ The tab for the various settings opens.
2. Double click on the "General" tab.
3. Double click on the "Value" of the parameter "Diagnosis behavior".
4. Click on the small arrow. $\Rightarrow$ A submenu with two values opens.
5. Click on "AC500 V2 compatible" as setting.
6. Close the tab.

After changing the parameter to "AC500 V2 compatible" the CI54x get in "RUN".
If the Cl 54 x indicates a S500 diagnosis message, following AC500 diagnosis entry ("655374 CI54x communication interface module is sending not supported diagnosis format - Check configuration and FW revision of communication interface module") is shown in the diagnosis editor and history. This diagnosis message does not have impact to cyclic data exchange between the master and the CI 54 x .

In case of a parameter change from V2 to V3 the parameter at the Cl 54 x of V 3 has the same value than at the $\mathrm{Cl} 54 x$ below V 2 (that means AC 500 V 2 compatible).

## Connections

## General

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

The PROFIBUS DP communication interface module CI542-DP is plugged on the I/O terminal units TU509 \& Chapter 5.2.7.3 "TU509 and TU510 for communication interface modules" on page 1274 or TU510 $\Leftrightarrow$ Chapter 5.2.7.3 "TU509 and TU510 for communication interface modules" on page 1274 and accordingly TU517 $\Leftrightarrow$ Chapter 5.2.7.4 "TU517 and TU518 for communication interface modules" on page 1278 or TU518 $\Leftrightarrow$ Chapter 5.2.7.4 "TU517 and TU518 for communication interface modules" on page 1278. Properly seat the module and press until it locks in place. The terminal unit is mounted on a DIN rail or with 2 screws plus the additional accessory for wall mounting ${ }^{*}$ Chapter 5.2.8.2.5 "TA526-Wall mounting accessory" on page 1324.
The connection of the I/O channels is carried out using the 30 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

The terminals 2.8 and 3.8 as well as $2.9,3.9$ and 4.9 are electrically interconnected within the terminal unit and have always the same assignment, independent of the inserted module:
Terminals 2.8 and 3.8: Process supply voltage UP $=+24 \mathrm{~V}$ DC
Terminal 4.8: Process supply voltage UP3 $=+24 \mathrm{~V}$ DC
Terminals 2.9, 3.9 and 4.9: Process supply voltage $\mathrm{ZP}=0 \mathrm{~V}$

With a separate UP3 power supply, the digital outputs can be switched off externally. This way, an emergency-off functionality can be realized.

Do not connect any voltages externally to digital outputs!
This ist not intended usage.
Reason: Externally voltages at one or more terminals DC0...DC7 or DOO...DO7 may cause that other digital outputs are supplied through that voltage instead of voltage UP3 (reverse voltage).
This is also possible, if DC channels are used as inputs. For this, the source for the input signals should be the impressed UP3 of the device.
This limitation does not apply for the input channels DIO...DI7.

## CAUTION!

## Risk of malfunction by unintended usage!

If the function cut-off of the digital outputs is to be used by deactivation of the supply voltage UP3, be sure that no external voltage is conncted at the outputs DO0...DO7 and DC0...DC7.

## Possibilities of connection

## Assignment

Mounting on terminal units TU509 or TU510:
The assignment of the 9-pole female D-sub for the PROFIBUS DP signals.

| Serial Interface | Pin | Signal | Description |
| :---: | :---: | :---: | :---: |
|  | 1 | --- | Reserved |
|  | 2 | --- | Reserved |
|  | 3 | B | PROFIBUS DP signal B |
|  | 4 | --- | Reserved |
|  | 5 | DGND | Ground for 5 V power supply |
|  | 6 | VP (5 V) | 5 V power supply |
|  | 7 | --- | Reserved |
|  | 8 | A | PROFIBUS DP signal A |
|  | 9 | --- | Reserved |
|  | Shield | Cable shield | Functional earth |

## Bus termination

The line ends of the bus segment must be equipped with bus terminating resistors. Normally, these resistors are integrated in the interface connectors.



The grounding of the shield should take place at the control cabinet, see System-Data AC500 « Chapter 5.1.2 "System data AC500" on page 166.

Mounting on terminal units TU517 or TU518:
The assignment of the terminals 1.0-1.9:

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 1.0 | B | Data line B (receive and send line, posi- <br> tive) |
| 1.1 | B | Data line B (receive and send line, posi- <br> tive) |
| 1.2 | A | Data line A (receive and send line, nega- <br> tive) |
| 1.3 | A | Data line A (receive and send line, nega- <br> tive) |
| 1.4 | TermB | Bus termination data line B |
| 1.5 | TermB | Bus termination data line B |
| 1.6 | TermA | Bus termination data line A |


| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 1.7 | TermA | Bus termination data line A |
| 1.8 | DGND | Reference potential for data transmis- <br> sion |
| 1.9 | DGND | Reference potential for data transmis- <br> sion |

At the line ends of a bus segment, terminating resistors must be connected. If using TU517/ TU518, the bus terminating resistors can be enabled by connecting the terminals TermA and TermB to the data lines $A$ and $B$ (no external terminating resistors are required, see figure below).


If using TU517/TU518, note that the terminating resistors are not located inside the TU, but inside the communication interface module CI541-DP. I. e. when removing the device from the TU, the bus terminating resistors are not connected to the bus any more. The bus itself will not be disconnected if a device is removed.

If using TU517/TU518 the max. permitted transmission rate is limited to 1.5 MBaud.

## Technical data bus cable

| Parameter | Value |
| :--- | :--- |
| Type | Twisted pair (shielded) |
| Characteristic impedance | $135 \Omega \ldots . .165 \Omega$ |
| Cable capacitance | $<30 \mathrm{pF} / \mathrm{m}$ |
| Conductor diameter of the cores | $\geq 0.64 \mathrm{~mm}$ |
| Conductor cross section of the cores | $\geq 0.34 \mathrm{~mm}^{2}$ |
| Cable resistance per core | $\leq 55 \Omega / \mathrm{km}$ |
| Loop resistance (resistance of two cores) | $\leq 110 \Omega / \mathrm{km}$ |

## Cable length

The maximum possible cable length of a PROFIBUS subnet within a segment depends on the transmission rate (baud rate).

| Transmission rate | Maximum cable length |
| :--- | :--- |
| 9.6 kBaud to 93.75 kBaud | 1200 m |
| 187.5 kBaud | 1000 m |
| 500 kBaud | 400 m |
| 1.5 MBaud | 200 m |
| 3 MBaud to 12 MBaud | 100 m |

The assignment of the other terminals:

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 2.0 | DC0 | Signal of the configurable digital input/output DC0 |
| 2.1 | DC1 | Signal of the configurable digital input/output DC1 |
| 2.2 | DC2 | Signal of the configurable digital input/output DC2 |
| 2.3 | DC3 | Signal of the configurable digital input/output DC3 |
| 2.4 | DC4 | Signal of the configurable digital input/output DC4 |
| 2.5 | DC5 | Signal of the configurable digital input/output DC5 |
| 2.6 | DC6 | Signal of the configurable digital input/output DC6 |
| 2.7 | DC7 | Signal of the configurable digital input/output DC7 |


| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 2.8 | UP | Process voltage UP (24 V DC) |
| 2.9 | ZP | Process voltage ZP (0 V DC) |
| 3.0 | DI9 | Signal of the digital input DI8 |
| 3.1 | DI10 | Signal of the digital input DI9 |
| 3.2 | DI12 | Signal of the digital input DI10 |
| 3.3 | DI13 | Signal of the digital input DI11 |
| 3.4 | DI14 | Signal of the digital input DI12 |
| 3.5 | UP | Signal of the digital input DI13 |
| 3.6 | DO8 | Signal of the digital input DI14 |
| 3.7 | DO9 | Process voltage UP (24 V DC) |
| 3.8 | DO10 | Process voltage ZP (0 V DC) |
| 3.9 | DO11 | Signal of the digital output DO8 |
| 4.0 | DO12 | Signal of the digital output DO9 |
| 4.1 | DO13 | Signal of the digital output DO10 |
| 4.2 | DO14 | Signal of the digital output DO11 |
| 4.3 | DO15 | Signal of the digital output DO13 |
| 4.4 | Signal of the digital output DO14 |  |
| 4.5 | Signal of the digital output DO15 |  |
| 4.6 | Process voltage UP3 (24 V DC) |  |
| 4.7 | Process voltage ZP (0 V DC) |  |
| 4.8 | DP |  |

## WARNING!

## Removal/Insertion under power

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.
Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.
The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

## Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.


Fig. 238: Connection of the PROFIBUS DP communication interface module CI542-DP

## Connection of the digital inputs



Fig. 239: Connection of the digital input DI8 (Proceed with the digital inputs DI9 to DI15 in the same way)

The meaning of the LEDs is described in Displays ${ }^{\circledR}>$ Chapter 5.2.6.6.2.10 "State LEDs" on page 1193.

## Connection of the digital outputs



Fig. 240: Connection of the digital output DO8 (Proceed with the digital outputs DO9-DO15 in the same way)
The meaning of the LEDs is described in Displays "y Chapter 5.2.6.6.2.10 "State LEDs" on page 1193.

## Connection of the configurable digital inputs/outputs

## CAUTION!

If a DC channel is used as input, the source for the input signals should be the impressed UP3 of the device ${ }^{\star} \Rightarrow$ Chapter 5.2.6.6.2.4 "Connections" on page 1178.


Fig. 241: Connection of the configurable digital input/output DC0 and DC1 (Proceed with the configurable digital inputs/outputs DC2 to DC7 in the same way)
The meaning of the LEDs is described in Displays $\stackrel{y}{ }$ Chapter 5.2.6.6.2.10 "State LEDs" on page 1193.

## Internal data exchange

| Parameter | Value |
| :--- | :--- |
| Digital inputs (bytes) | 5 |
| Digital outputs (bytes) | 5 |
| Counter input data (words) | 4 |
| Counter output data (words) | 8 |

## Addressing

The module reads the position of the rotary switches only during power-up, i.e. changes of the switch position during operation will have no effect until the next module initialization.

## I/O configuration

The CI542-DP PROFIBUS DP bus configuration is handled by PROFIBUS DP master with the exception of the slave bus ID (via rotary switches) and the transmission rate (automatic detection).

The digital I/O channels and the fast counter are configured via software.
Details about configuration are described in Parameterization.

## Parameterization

## Parameters of the module

| Name | Value | Internal value | Internal value, type | Default |
| :---: | :---: | :---: | :---: | :---: |
| Module ID ${ }^{1}$ ) | Internal | 0x1C25 | WORD | 0x1C25 |
| Parameter length | Internal | 31 | BYTE | 31 |
| Reserved (1 byte) | 0 | 0 | BYTE | 0 |
| Error LED / Failsafe function ③ Table 261 "Set tings "Error LED / Failsafe function"" on page 1187 (see table ) | On | 0 | BYTE | 0 |
|  | Off by E4 | 1 |  |  |
|  | Off by E3 | 2 |  |  |
|  | On + failsafe | 16 |  |  |
|  | Off by E4 + failsafe | 17 |  |  |
|  | Off by E3 + failsafe | 18 |  |  |
| Reserved (20 bytes) | 0 | 0 | BYTE | 0 |
| Check supply | On | 0 | BYTE |  |
|  | Off | 1 |  | 1 |
| Fast counter | 0 | 0 | BYTE | 0 |
|  | : | : |  |  |
|  | $10^{2}$ ) | 10 |  |  |

${ }^{1}$ ) With a faulty ID, the module reports a "parameter error" and does not perform cyclic process data transmission.
${ }^{2}$ ) Counter operating modes, see 'Fast Counter' $\Longleftrightarrow$ Chapter 5.2.4.3.2.9 "Fast counter" on page 606.

Table 261: Settings "Error LED / Failsafe function"

| Setting | Description |
| :--- | :--- |
| On | Error LED (S-ERR) lights up at errors of all <br> error classes, Failsafe mode off |
| Off by E4 | Error LED (S-ERR) lights up at errors of error <br> classes E1, E2 and E3, Failsafe mode off |
| Off by E3 | Error LED (S-ERR) lights up at errors of error <br> classes E1 and E2, Failsafe mode off |
| On + Failsafe | Error LED (S-ERR) lights up at errors of all <br> error classes, Failsafe mode on *) |


| Setting | Description |
| :--- | :--- |
| Off by E4 + Failsafe | Error LED (S-ERR) lights up at errors of error <br> classes E1, E2 and E3, Failsafe mode on *) |
| Off by E3 + Failsafe | Error LED (S-ERR) lights up at errors of error <br> classes E1 and E2, Failsafe mode on *) |
| *) The parameter Behaviour DO at comm. error is only analyzed if the Failsafe mode is ON. |  |

## Group parameters for the digital part

| Name | Value | Internal value | Internal value, type | Default |
| :---: | :---: | :---: | :---: | :---: |
| Input delay | 0.1 ms <br> 1 ms <br> 8 ms <br> 32 ms | $\begin{aligned} & 0 \\ & 1 \\ & 2 \\ & 3 \end{aligned}$ | BYTE | $\begin{aligned} & 0.1 \mathrm{~ms} \\ & 0 \times 00 \end{aligned}$ |
| Detect short circuit at outputs | $\begin{aligned} & \text { Off } \\ & \text { On } \end{aligned}$ | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | BYTE | $\begin{aligned} & \text { On } \\ & 0 \times 01 \end{aligned}$ |
| Behaviour DO at comm. error ${ }^{1}$ ) | Off <br> Last value <br> Last value 5 sec <br> Last value 10 sec <br> Substitute value <br> Substitute value 5 sec <br> Substitute value 10 sec | $\begin{aligned} & \hline 0 \\ & 1 \\ & 6 \\ & 11 \\ & 2 \\ & 2 \\ & 7 \\ & 12 \end{aligned}$ | BYTE | $\begin{aligned} & \text { Off } \\ & 0 \times 00 \end{aligned}$ |
| Substitute value at output | 0 ... 65535 | 0000h ... FFFFh | WORD | $\begin{aligned} & 0 \\ & 0 \times 0000 \end{aligned}$ |
| Preventive voltage feedback monitoring for DC0 ... DC7 ${ }^{2}$ ) | $\begin{aligned} & \text { Off } \\ & \text { On } \end{aligned}$ | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | BYTE | $\begin{aligned} & \text { Off } \\ & 0 \times 00 \end{aligned}$ |
| Detect voltage overflow at outputs ${ }^{3}$ ) | $\begin{aligned} & \text { Off } \\ & \text { On } \end{aligned}$ | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | BYTE | $\begin{aligned} & \text { Off } \\ & 0 \times 00 \end{aligned}$ |

Remarks:

| ${ }^{1}$ ) | The parameter Behaviour DO at comm. error is apply to DC and DO channels <br> and only analyzed if the Failsafe-mode is ON. |
| :--- | :--- |
| ${ }^{2}$ ) | The state "externally voltage detected" appears, if the output of a channel <br> DCO ... DC7 should be switched on while an externally voltage is connected. In <br> this case the start up is disabled, as long as the externally voltage is connected. <br> The monitoring of this state and the resulting diagnosis message can be disabled <br> by setting the parameters to "OFF". |
| ${ }^{3}$ ) | The error state "voltage overflow at outputs" appears, if externally voltage at dig- <br> ital outputs DC0 .. DC7 and accordingly DOO ... DO7 has exceeded the process <br> supply voltage UP3 \& Chapter 5.2.6.6.2.4 "Connections" on page 1178. The <br> according diagnosis message "Voltage overflow on outputs " can be disabled <br> by setting the parameters on "OFF". This parameter should only be disabled in <br> exceptional cases for voltage overflow may produce reverse voltage. |

## Diagnosis

Structure of the diagnosis block via DPM_SLV_DIAG function block.
$\left.\begin{array}{|l|l|l|}\hline \text { Byte Number } & \text { Description } & \text { Possible Values } \\ \hline 1 & \begin{array}{l}\text { Data length (header } \\ \text { included) }\end{array} & 7 \\ \hline 2 & \begin{array}{l}\text { PROFIBUS DP V1 coding: } \\ \text { Vendor specific }\end{array} & 129 \\ \hline 3 & \text { Diagnosis Byte, slot number } & \begin{array}{l}\text { l1 = CI542-DP (e. g. error at integrated 8 DI / } \\ 8 \text { DO) } \\ 1=1 \text { st connected S500 I/O module }\end{array} \\ \hline 4 & \begin{array}{l}\text { Diagnosis Byte, module } \\ \text { number }\end{array} & \begin{array}{l}\text { According to the I/O bus specification passed } \\ \text { on by modules to the fieldbus master }\end{array} \\ \hline 5 & \text { Diagnosis Byte, channel } & \begin{array}{l}\text { According to the I/O bus specification passed } \\ \text { on by modules to the fieldbus master }\end{array} \\ \hline 6 & \text { Diagnosis Byte, error code } & \begin{array}{l}\text { According to the I/O bus specification } \\ \text { Bit } 7 \text { and bit } 6, \text { coded error class } \\ 0=\text { E1 }\end{array} \\ \hline 1=\text { E2 } \\ 2=\text { E3 } \\ 3=\text { E4 } \\ \text { Bit } 0 \text { to bit 5, coded error description }\end{array}\right]$

In cases of short circuit or overload, the digital outputs are turned off. The modules performs reactivation automatically. Thus an acknowledgement of the errors is not necessary. The error message is stored via the LED.

| E1 ... E4 | d1 | d2 | d3 | d4 | Identifier 000 ... | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err |  |  |
| $\begin{aligned} & \hline \text { Byte } 6 \\ & \text { Bit } \\ & 6 \ldots 7 \end{aligned}$ | - | Byte 3 | Byte 4 | Byte 5 | $\begin{aligned} & \text { Byte } 6 \\ & \text { Bit } \\ & 0 \text {... } 5 \end{aligned}$ |  |  |
| Class | $\begin{aligned} & \text { Inter- } \\ & \text { face } \end{aligned}$ | Device | Module | Channel | Error- <br> Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) |  |  |  |  |
| Module errors |  |  |  |  |  |  |  |
| 3 | - | 31 | 31 | 31 | 19 | Checksum error in the I/O module | Replace I/O module |
| 3 | - | 31 | 31 | 31 | 3 | Timeout in the I/O module |  |
| 3 | - | 31 | 31 | 31 | 40 | Different hard-/firmware versions in the module |  |
| 3 | - | 31 | 31 | 31 | 43 | Internal error in the module |  |
| 3 | - | 31 | 31 | 31 | 36 | Internal data exchange failure |  |
| 3 | - | 31 | 31 | 31 | 9 | Overflow diagnosis buffer | Restart |
| 3 | - | 31 | 31 | 31 | 26 | Parameter error | Check master |
| 3 | - | 31 | 31 | 31 | 11 | Process voltage UP too low | Check process supply voltage |
| 3 | - | 31 | 31 | 31 | 45 | Process voltage UP gone | Check process supply voltage |
| 3 | - | 31/1... 10 | 31 | 31 | 17 | No communication with I/O module | Replace I/O module |
| 3 | - | 1... 10 | 31 | 31 | 32 | Wrong I/O module type on socket | Replace I/O module / Check configuration |
| 4 | - | 1... 10 | 31 | 31 | 31 | At least one module does not support failsafe function | Check modules and parameterization |


| E1 ... E4 | d1 | d2 | d3 | d4 | Identifier 000 ... 0 | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err |  |  |
| $\begin{aligned} & \text { Byte } 6 \\ & \text { Bit } \\ & 6 \text {... } 7 \end{aligned}$ | - | Byte 3 | Byte 4 | Byte 5 | Byte 6 <br> Bit <br> 0 ... 5 |  |  |
| Class | Interface | Device | Module | Channel | Error- <br> Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) |  |  |  |  |
| 4 | - | 1... 10 | 31 | 5 | 8 | I/O module removed from hot swap terminal unit or defective module on hot swap terminal unit ${ }^{9}$ ) | Plug I/O module, replace I/O module |
| 4 | - | 1... 10 | 31 | 5 | 28 | Wrong I/O module plugged on hot swap terminal unit ${ }^{9}$ ) | Remove wrong I/O module and plug projected I/O module |
| 4 | - | 1... 10 | 31 | 5 | 42 | No communication with I/O module on hot swap terminal unit ${ }^{9}$ ) | Replace I/O module |
| 4 | - | 1... 10 | 31 | 5 | 54 | I/O module does not support hot swap $\left.{ }^{8}\right)^{9}$ ) | Power off system and replace I/O module |
| 4 | - | 1... 10 | 31 | 6 | 8 | Hot swap terminal unit configured but not found | Replace terminal unit by hot swap terminal unit |
| 4 | - | 1... 10 | 31 | 6 | 42 | No communication with hot swap terminal unit ${ }^{9}$ ) | Restart, if error persists replace terminal unit |
| 4 | - | 31 | 31 | 31 | 45 | Process voltage UP3 too low | Check process voltage |



[^21]| ${ }^{1}$ ) | In AC500 the following interface identifier applies: <br> "-" = Diagnosis via bus-specific function blocks; 0 ... 4 or $10=$ Position of the Communication Module;14 = I/O bus; $31=$ Module itself <br> The identifier is not contained in the CI542-DP diagnosis block. |
| :---: | :---: |
| ${ }^{2}$ ) | With "Device" the following allocation applies: $31=$ Module itself, $1 . .10=$ expansion module |
| $\left.{ }^{3}\right)$ | With "Module" the following allocation applies dependent of the master: <br> Module error: 31 = Module itself <br> Channel error: Module type ( 1 = AI, 2 = DO, 3 = AO) |
| ${ }^{4}$ ) | This message appears, if externally voltages at one or more terminals DC0 ... DC7 oder DO0 ... DO7 cause that other digital outputs are supplied through that voltage. <br> All outputs of the apply digital output groups will be turned off for 5 seconds. The diagnosis message appears for the whole output group. |
| ${ }^{5}$ ) | The voltage at digital outputs DC0 .. DC7 and accordingly DO0 ... DO7 has exceeded the process supply voltage UP3 ${ }^{*}$ 攵 Chapter 5.2.6.6.2.4 "Connections" on page 1178. Diagnosis message appears for the whole module. |
| ${ }^{6}$ ) | This message appears, if the output of a channel DC0 ... DC7 or DO0 ... DO7 should be switched on while an externally voltage is connected. In this case the start up is disabled, as long as the externally voltage is connected. Otherwise this could produce reverse voltage from this output to other digital outputs. This diagnosis message appears per channel. |
| ${ }^{7}$ ) | Short circuit: After a detected short circuit, the output is deactivated for 100 ms . Then a new start up will be executed. This diagnosis message appears per channel. |
| $\left.{ }^{8}\right)$ | In case of an I/O module doesn't support hot swapping, do not perform any hot swap operations (also not on any other terminal units (slots)) as modules may be damaged or I/O bus communication may be disturbed. |
| ${ }^{9}$ ) | Diagnosis for hot swap available as of version index F0. |

## State LEDs

The LEDs are located at the front of module. There are 2 different groups:

- The 5 system LEDs (PWR, STA1 DP, STA2 DP, S-ERR and I/O-Bus) show the operation state of the module and display possible errors.
- The 29 process LEDs (UP, UP3, inputs, outputs, CH-ERR1 to CH-ERR3) show the process supply voltage and the states of the inputs and outputs and display possible errors.

Table 262: States of the 5 system LEDs:

| LED | Color | OFF | ON | Flashing |
| :--- | :--- | :--- | :--- | :--- |
| PWR/RUN | Green | Process supply <br> voltage missing | Internal supply <br> voltage OK, <br> module ready for <br> communication <br> with I/O Con- <br> troller | Start-up / pre- <br> paring communi- <br> cation |
| STA1-DP | Green | --- | --- | --- |
|  | Yellow | PROFIBUS run- <br> ning | Invalid device <br> parameters |  |


| LED | Color | OFF | ON | Flashing |
| :--- | :--- | :--- | :--- | :--- |
| STA2-DP | Red | No error | Bus timeout | No communica- <br> tion to master |
| S-ERR | Red | No error | Internal error | -- |
| I/O-Bus | Green | No communica- <br> tion interface <br> modules con- <br> nected or com- <br> munication error | Communication <br> interface module <br> connected and <br> operational | --- |

Table 263: States of the 29 process LEDs:

| LED | Color | OFF | ON | Flashing |
| :--- | :--- | :--- | :--- | :--- |
| DC0 ... DC7 | Yellow | Input/Output is <br> OFF | Input/Output is <br> ON | -- |
| DI8 ... DI15 | Yellow | Input is OFF | Input is ON (the <br> input voltage is <br> even displayed if <br> the supply <br> voltage is OFF) | -- |
| DO8 ... DO15 | Yellow | Green | Output is OFF <br> voltage missing | Output is ON |
| UP | Process supply <br> voltage OK and <br> initialization fin- <br> ished | --- |  |  |
| UP3 | Green | Process supply <br> voltage missing | Process supply <br> voltage OK | --- |
| CH-ERR1 to CH- <br> ERR3 | Red | No error or <br> process supply <br> voltage missing | Internal error | Error on one <br> channel of the <br> corresponding <br> group |

## Technical data

## Technical data of the module

The system data of AC500 and S500 are applicable to the standard version $\Leftrightarrow$ Chapter 5.1.2 "System data AC500" on page 166.
The system data of AC500-XC are applicable to the XC version ${ }^{〔}$ Chapter 5.1.3 "System data AC500-XC" on page 169.
Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter |  | Value |
| :--- | :--- | :--- |
| Process supply voltages UP/UP3 |  |  |
|  | Rated value | 24 V DC (for inputs and outputs) |
|  | Max. load for the terminals | 10 A |
|  | Protection against reversed voltage | Yes |
|  | Rated protection fuse on UP/UP3 | 10 A fast |


| Parameter | Value |
| :---: | :---: |
| Galvanic isolation | PROFIBUS interface against the rest of the module |
| Inrush current from UP (at power up) | On request |
| Current consumption via UP (normal operation) | 0.2 A |
| Current consumption via UP3 | 0.06 A + 0.5 A max. per output |
| Connections | Terminals 2.8 and 3.8 for +24 V (UP) <br> Terminal 4.8 for +24 V (UP3) <br> Terminals 2.9, 3.9 and 4.9 for $0 \mathrm{~V}(\mathrm{ZP})$ |
| Max. power dissipation within the module | 6 W |
| Number of digital inputs | 8 |
| Number of digital outputs | 8 |
| Number of analog inputs | 4 |
| Number of analog outputs | 2 |
| Reference potential for all digital inputs and outputs | Negative pole of the supply voltage, signal name ZP |
| Setting of the PROFIBUS DP identifier | With 2 rotary switches at the front side of the module |
| Diagnose | See Diagnosis « ${ }^{\mu}$ Chapter 5.2.6.6.2.9 "Diagnosis" on page 1189 |
| Operation and error displays | 34 LEDs (totally) |
| Weight (without terminal unit) | Ca. 125 g |
| Mounting position | Horizontal <br> Or vertical with derating (output load reduced to $50 \%$ at $+40^{\circ} \mathrm{C}$ per group) |
| Cooling | The natural convection cooling must not be hindered by cable ducts or other parts in the control cabinet. |

## - NOTICE!

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and temporary overvoltage up to 30 V DC.

## Multiple overloads

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an internal smart high-side switch.

## Technical data of the digital inputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 8 |
| Distribution of the channels into groups | 1 group of 8 channels |


| Parameter | Value |
| :--- | :--- |
| Terminals of the channels DI0 ... DI7 | Terminals $3.0 \ldots 3.7$ |
| Reference potential for all inputs | Terminals $2.9 \ldots 4.9$ (negative pole of the <br> supply voltage, signal name ZP) |
| Indication of the input signals | 1 yellow LED per channel, the LED is ON when <br> the input signal is high (signal 1) |
| Input type (according EN 61131-2) | Type 1 |
| Input delay (0->1 or 1->0) | Typ. 0.1 ms, configurable from $0.1 \mathrm{~ms} \ldots 32 \mathrm{~ms}$ |
| Input signal voltage | 24 V DC |
|  | Signal 0 |
|  | Undefined Signal |
|  | Signal 1 |
| Ripple with signal 0 | $>+5 \mathrm{~V} \ldots<+15 \mathrm{~V}$ |
| Ripple with signal 1 | $+15 \mathrm{~V} \ldots+30 \mathrm{~V}$ |
| Input current per channel | Within $-3 \mathrm{~V} \ldots+5 \mathrm{~V}$ |
|  | Input voltage +24 V |
|  | Within $+15 \mathrm{~V} \ldots+30 \mathrm{~V}$ |
|  | Input voltage +5 V |
|  | Input voltage +30 V |
| Max. cable length | Typ. 5 mA |
|  | Shielded |
|  | Unshielded |
|  | $>2 \mathrm{~mA}$ |

## Technical data of the digital outputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 8 |
| Distribution of the channels into groups | 1 group of 8 channels |
| Terminals of the channels DO0 ... DO7 | Terminals $4.0 \ldots 4.7$ |
| Reference potential for all outputs | Terminals $2.9 \ldots 4.9$ (negative pole of the <br> supply voltage, signal name ZP) |
| Common power supply voltage | For all outputs terminal 4.8 (positive pole of the <br> supply voltage, signal name UP3) |
| Output voltage for signal 1 | UP3 (-0.8 V) |
| Output delay (0->1 or 1->0) | On request |
| Output current | 500 mA at UP3 = 24 V |
|  | Rated value per channel |
|  | Max. value (all channels together) |
| Leakage current with signal 0 | $<0.5 \mathrm{~mA}$ |
| Fuse for UP3 |  |
| Demagnetization with inductive DC load | Via internal varistors (see figure below this <br> table) |
| Output switching frequency |  |


| Parameter |  | Value |
| :--- | :--- | :--- |
|  | With resistive load | On request |
|  | With inductive loads | Max. 0.5 Hz |
|  | With lamp loads | 11 Hz max. at 5 W max. |
| Short-circuit-proof / overload-proof | Yes |  |
| Overload message (I > 0.7 A) | Yes, after ca. 100 ms |  |
| Output current limitation | Yes, automatic reactivation after short cir- <br> cuit/overload |  |
| Resistance to feedback against 24 V signals | Yes (software-controlled supervision) |  |
| Max. cable length |  |  |
|  | Shielded | 1000 m |
|  | Unshielded | 600 m |

The following drawing shows the circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off.


1 Digital output
2 Varistors for demagnetization when inductive loads are turned off

## Technical data of the configurable digital inputs/outputs

Each of the configurable I/O channels is defined as input or output by the user program. This is done by interrogating or allocating the corresponding channel.

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 8 inputs/outputs (with transistors) |
| Distribution of the channels into groups | 1 group for 8 channels |
| If the channels are used as inputs |  |
| Channels DC0 ... DC07 | Terminals 2.0 ... 2.7 |
| If the channels are used as outputs |  |
| Channels DC0 ... DC07 | Terminals 2.0 ... 2.7 |
| Indication of the input/output signals | 1 yellow LED per channel, the LED is ON when the input/output signal is high (signal 1) |
| Galvanic isolation | From the PROFIBUS network |

Technical data of the digital inputs/outputs if used as inputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 8 |
| Distribution of the channels into groups | 1 group of 8 channels |
| Terminals of the channels DC0 ... DC7 | Terminals $2.0 \ldots 2.7$ |
| Reference potential for all inputs | Terminals $2.9 \ldots 4.9$ (negative pole of the <br> supply voltage, signal name ZP) |
| Indication of the input signals | 1 yellow LED per channel, the LED is ON when <br> the input signal is high (signal 1 ) |
| Input type (according EN 61131-2) | Type 1 |
| Input delay (0->1 or 1->0) | Typ. 0.1 ms, configurable from $0.1 \mathrm{~ms} \ldots 32 \mathrm{~ms}$ |
| Input signal voltage | 24 V DC |
|  | Signal 0 |
|  | $-3 \mathrm{~V} \ldots+5 \mathrm{~V}$ |
|  | Signal 1 |
| Ripple with signal 0 | $>+5 \mathrm{~V} \ldots<+15 \mathrm{~V}$ |
| Ripple with signal 1 | $+15 \mathrm{~V} \ldots+30 \mathrm{~V}$ |
| Input current per channel | Within $-3 \mathrm{~V} \ldots+5 \mathrm{~V}$ |
|  | Input voltage +24 V |
| Input voltage +5 V | Within $+15 \mathrm{~V} \ldots+30 \mathrm{~V}$ |
|  | Input voltage +15 V |
| Input voltage +30 V | $>1 \mathrm{~mA}$ |
| Max. cable length | $>2 \mathrm{~mA}$ |
|  | Shielded |
| Unshielded | $<8 \mathrm{~mA}$ |

*) Due to the direct connection to the output, the demagnetizing varistor is also effective at the input (see figure) above. This is why the difference between UPx and the input signal may not exceed the clamp voltage of the varistor. The varistor limits the voltage to approx. 36 V . Following this, the input voltage must range from $-12 \mathrm{~V} \ldots+30 \mathrm{~V}$ when $\mathrm{UPx}=24 \mathrm{~V}$ and from $-6 \vee \ldots+30 \vee$ when UPx $=30 \mathrm{~V}$.

## Technical data of the digital inputs/outputs if used as outputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 8 |
| Distribution of the channels into groups | 1 group of 8 channels |
| Terminals of the channels DC0 ... DC7 | Terminals $2.0 \ldots 2.7$ |
| Reference potential for all outputs | Terminals $2.9 \ldots 4.9$ (negative pole of the <br> supply voltage, signal name ZP) |
| Common power supply voltage | For all outputs terminal 4.8 (positive pole of the <br> supply voltage, signal name UP3) |
| Output voltage for signal 1 | UP3 (-0.8 V) |
| Output delay (0->1 or $1->0)$ | On request |
| Output current |  |


| Parameter | Value |
| :---: | :---: |
| Rated value per channel | 500 mA at UP3 $=24 \mathrm{~V}$ |
| Max. value (all channels together) | 4 A |
| Leakage current with signal 0 | $<0.5 \mathrm{~mA}$ |
| Fuse for UP3 | 10 A fast |
| Demagnetization with inductive DC load | Via internal varistors (see figure below this table) |
| Output switching frequency |  |
| With resistive load | On request |
| With inductive loads | Max. 0.5 Hz |
| With lamp loads | 11 Hz max. at 5 W max. |
| Short-circuit-proof / overload-proof | Yes |
| Overload message ( $1>0.7$ A) | Yes, after ca. 100 ms |
| Output current limitation | Yes, automatic reactivation after short circuit/overload |
| Resistance to feedback against 24 V signals | Yes (software-controlled supervision) |
| Max. cable length |  |
| Shielded | 1000 m |
| Unshielded | 600 m |

The following drawing shows the circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off.


1 Digital input/output
2 For demagnetization when inductive loads are turned off

## Technical data of the fast counter

| Parameter | Value |
| :--- | :--- |
| Used inputs | Terminal 3.0 (DI0),Terminal 3.1 (DI1) |
| Used outputs | Terminal 4.0 (DO0) |
| Counting frequency | Depending on operation mode: |
|  | Mode 1- 6: max. 200 kHz |
|  | Mode 7: max. 50 kHz |
|  | Mode 9: max. 35 kHz |
|  | Mode 10: max. 20 kHz |

[^22]
## Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm

The dimensions are in mm and in brackets in inch.

Ordering Data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 224 200 R0001 | Cl542-DP, PROFIBUS DP communica- <br> tion interface module, 8 DI, 8 DO and <br> 8 DC | Active |
| 1SAP 424 200 R0001 | CI542-DP-XC, PROFIBUS DP com- <br> munication interface module, 8 DI, <br> 8 DO and 8 DC, XC version | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.6.7 PROFINET

### 5.2.6.7.1 Comparison of the CI5xx-PNIO modules

The PROFINET IO devices combine the advantages of decentralized I/O modules with the reaction time of AC500 mounted central I/O modules. The devices for PROFINET provide the extension -PNIO in the device name.

The communication module CM579-PNIO acts as I/O controller in a PROFINET network. It is connected to the processor module via an internal communication bus. Depending on the terminal base, several communication modules can be used for one processor module.

The communication interface modules $\mathrm{CI} 5 x x-$ PNIO act as I/O devices in a PROFINET network.
Additionally the communication module CM589-PNIO(-4) can be used to setup a AC500 PLC to act as I/O module in a PROFINET network.

The difference of the CI5xx-PNIO devices can be found in their input and output characteristics Chapter 5.2.6.7.1.1 "Characteristics of CI50x-PNIO" on page 1201.

## Characteristics of CI50x-PNIO

| Parameter | Value |
| :--- | :--- |
| Bus connection | 2 x RJ45 |
| Switch | Integrated |
| Technology | Hilscher NETX 100 |
| Transfer rate | $10 / 100$ Mbit/s (full-duplex) |
| Transfer method | According to Ethernet II, IEEE 802.3 |
| Ethernet | 100 base-TX, internal switch, 2x RJ45 socket |
| Expandability | Max. 10 S500 I/O modules |
| Adjusting elements | 2 rotary switches for generation of an explicit <br> name |
| Supported protocols | RTC - real time cyclic protocol, class 1 *) <br> RTA - real time acyclic protocol <br> DCP - discovery and configuration protocol <br> CL-RPC - connectionless remote procedure <br> Call <br> LLDP - link layer discovery protocol <br> MRP - MRP Client |
| Acyclic services | PNIO read / write sequence (max. 1024 bytes <br> per telegram) <br> Process-Alarm service |
| Supported alarm types | Process Alarm, Diagnostic Alarm, Return of <br> SubModule, Plug Alarm, Pull Alarm |
| Min. bus cycle | 1 ms |


| Parameter | Value |
| :--- | :--- |
| Conformance class | CC A |
| Protective functions (according to | Protected against: |
| IEC 61131-3) | - short circuit |
|  | - reverse supply |
|  | - overvoltage |
|  | - reverse polarity |
|  | Galvanic isolation from the rest of the module |

[^23]
## Input/Output characteristics of CI501-PNIO

The PROFINET communication interface module $\mathrm{Cl} 501-\mathrm{PNIO}$ is used as decentralized I/O module in PROFINET networks. The network connection is performed via 2 RJ45 connectors which are integrated in the terminal unit. The communication interface module contains 22 I/O channels with the following properties:

- 4 analog inputs (1.0 ... 1.3), configurable as:
- $-10 \mathrm{~V} . . .+10 \mathrm{~V}$
- $0 \vee \ldots+10 \mathrm{~V}$
- $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ (differential voltage)
- $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$
- $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$
- Pt100, Pt1000, Ni1000 (for each 2-wire and 3-wire)
- 24 V digital input function
- 2 analog outputs (1.5 ... 1.6), configurable as:
- $-10 \mathrm{~V} . .+10 \mathrm{~V}$
- $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$
- $4 \mathrm{~mA} . . .20 \mathrm{~mA}$
- 8 digital inputs 24 V DC in 1 group (2.0 ... 2.7)
- 8 digital transistor outputs 24 V DC ( 0.5 A max.) in 1 group (3.0 ... 3.7)
- Resolution of the analog channels: 12 bits

The inputs/outputs are galvanically isolated from the Ethernet network. There is no potential separation between the channels. The configuration of the analog inputs/outputs is performed by software.
For usage in enhanced ambient conditions (e.g. wider temperature and humidity range), a special XC version of the device is available.

## Input/Output characteristics of CI502-PNIO

- 8 digital inputs 24 V DC
- 8 digital transistor outputs 24 V DC, 0.5 A max.
- 8 configurable digital inputs/outputs 24 V DC, 0.5 A max.
- Module-wise galvanically isolated
- XC version for usage in extreme ambient conditions available


## Technical data of the serial interfaces of CI504-PNIO

| Parameter | Value |
| :--- | :--- |
| Number of serial interfaces | 3 |
| Connectors for serial interfaces | X11 for COM1 |
|  | X12 for COM2 |
|  | X13 for COM3 |
| Supported physical layers | RS-232 |
|  | RS-422 |
|  | RS-485 |
| Supported protocols | ASCII |
| Transmission rate | Configurable from 300 bit/s to $115.200 \mathrm{bit/s}$ |

### 5.2.6.7.2 CI501-PNIO

## Features

- 4 analog inputs, 2 analog outputs, 8 digital inputs, 8 digital outputs
- Resolution 12 bits including sign
- Module-wise galvanically isolated
- Fast counter
- XC version for usage in extreme ambient conditions available


1 I/O bus
2 Allocation between terminal number and signal name
36 yellow LEDs to display the signal states of the analog inputs/outputs (AIO ... AI3, AOO ... AO1)
48 yellow LEDs to display the signal states of the digital inputs (DIO ... DI7)
58 yellow LEDs to display the signal states of the digital outputs (DO0 ... DO7)
62 green LEDs to display the process supply voltage UP and UP3
73 red LEDs to display errors (CH-ERR1, CH-ERR2, CH-ERR3)
85 system LEDs: PWR/RUN, STA1 ETH, STA2 ETH, S-ERR, I/O-Bus
9 Label
102 rotary switches for setting the I/O device identifier
11 Ethernet interfaces (ETH1, ETH2) on the terminal unit
12 Terminal unit
13 DIN rail
Sign for XC version

## Intended purpose

The PROFINET communication interface modules $\mathrm{CI} 501-\mathrm{PNIO}$ and $\mathrm{Cl} 502-\mathrm{PNIO}$ are used as communication interface modules in PROFINET networks. The network connection is performed by Ethernet cables which are inserted in the RJ45 connectors in the terminal unit. An Ethernet switch in the communication interface module allows daisy chaining of the network.
For usage in enhanced ambient conditions (e.g. wider temperature and humidity range), a special XC version of the device is available.

## Functionality

The communication interface module contains 22 I/O channels with the following properties:

- 4 configurable analog inputs (2-wire / single-ended) or 2 configurable analog inputs (3-wire / differential) (1.0 ... 1.3)
- 2 analog outputs (1.5 ... 1.6)
- 8 digital inputs 24 V DC in 1 group (2.0 ... 2.7)
- 8 digital outputs 24 V DC, 0.5 A max. in 1 group (3.0 ... 3.7)

The inputs/outputs are galvanically isolated from the PROFINET network. There is no potential separation between the channels. The configuration of the analog inputs/outputs is performed by software.

| Parameter | Value |
| :--- | :--- |
| Interface | Ethernet |
| Protocol | PROFINET IO RT |
| Power supply | From the process supply voltage UP |
| Supply of the electronic circuitry of the I/O <br> expansion modules attached | Through the I/O bus interface (I/O bus) |
| Rotary switches | For setting the I/O device identifier for configu- <br> ration purposes (00h to FFh) |
| LED displays | For system displays, signal states, errors and <br> power supply |
| External supply voltage | Via terminals ZP, UP and UP3 (process supply <br> voltage 24 V DC) |
| Effect of incorrect input terminal connection | Wrong or no signal detected, no damage up to <br> 35 V |
| Required terminal unit | TU507 or TU508E <br> ETH and TU50pter 5.2.7.2 "TU507- <br> tion interface modules" on page 1270 |

## Connections

## General

The Ethernet communication interface module $\mathrm{CI} 501-\mathrm{PNIO}$ is plugged on the I/O terminal unit TU507-ETH or TU508-ETH $\Leftrightarrow$ Chapter 5.2.7.2 "TU507-ETH and TU508-ETH for Ethernet communication interface modules" on page 1270. Properly seat the module and press until it locks in place. The terminal unit is mounted on a DIN rail or with 2 screws plus the additional accessory for wall mounting $\Leftrightarrow$ Chapter 5.2.8.2.5 "TA526 - Wall mounting accessory" on page 1324.
The connection of the I/O channels is carried out using the 30 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

The terminals 1.8 and 2.8 as well as $1.9,2.9$ and 3.9 are electrically interconnected within the terminal unit and have always the same assignment, independent of the inserted module:
Terminals 1.8 and 2.8: Process supply voltage UP $=+24 \mathrm{~V}$ DC
Terminal 3.8: Process supply voltage UP3 $=+24 \mathrm{~V}$ DC
Terminals 1.9, 2.9 and 3.9: Process supply voltage $\mathrm{ZP}=0 \mathrm{~V}$

With a separate UP3 power supply, the digital outputs can be switched off externally. This way, an emergency-off functionality can be realized.

Do not connect any voltages externally to digital outputs!
Reason: External voltages at an output or several outputs may cause that other outputs are supplied through that voltage instead of voltage UP3 (reverse voltage). This is unintended usage.

## CAUTION!

## Risk of malfunction by unintended usage!

If the function cut-off of the digital outputs is to be used by deactivation of the supply voltage UP3, be sure that no external voltage is connected at the outputs DO0 ... DO7.

Table 264: Assignment of the other terminals

| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 1.0 | AI0+ | Positive terminal of analog input signal 0 |
| 1.1 | Al1+ | Positive terminal of analog input signal 1 |
| 1.2 | Al2+ | Positive terminal of analog input signal 2 |
| 1.3 | Al3+ | Positive terminal of analog input signal 3 |
| 1.4 | AO0+ | Negative terminal of analog input signals 0 to 3 |
| 1.5 | AO1+ | Positive terminal of analog output signal 0 |
| 1.6 | AI- | Positive terminal of analog output signal 1 |
| 1.7 | ZP | Negative terminal of analog output signals 0 and 1 |
| 1.8 | DI0 | Process voltage UP (24 V DC) |
| 1.9 | DI1 | Process voltage ZP (0 V DC) |
| 2.0 | DI2 | Signal of the digital input DI0 |
| 2.1 | DI3 | Signal of the digital input DI2 |
| 2.2 | DI4 | Signal of the digital input DI3 |
| 2.3 | DI5 | Signal of the digital input DI4 |
| 2.4 |  | Signal of the digital input DI5 |
| 2.5 |  |  |


| Terminal | Signal | Description |
| :--- | :--- | :--- |
| 2.6 | DI6 | Signal of the digital input DI6 |
| 2.7 | DI7 | Signal of the digital input DI7 |
| 2.8 | UP | Process voltage UP (24 V DC) |
| 2.9 | ZP | Process voltage ZP (0 V DC) |
| 3.0 | DO0 | Signal of the digital output DO0 |
| 3.1 | DO1 | Signal of the digital output DO1 |
| 3.2 | DO2 | Signal of the digital output DO2 |
| 3.3 | DO3 | Signal of the digital output DO3 |
| 3.4 | DO4 | Signal of the digital output DO4 |
| 3.5 | DO6 | Signal of the digital output DO5 |
| 3.6 | DO7 | Signal of the digital output DO6 |
| 3.7 | UP3 | Signal of the digital output DO7 |
| 3.8 | ZP | Process voltage UP3 (24 V DC) |
| 3.9 |  |  |

## WARNING!

## Removal/Insertion under power

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.
Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.
The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

## Risk of damaging the PLC modules!

Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.

For the open-circuit detection (wire break), each analog input channel is pulled up to "plus" by a high-resistance resistor. If nothing is connected, the maximum voltage will be read in then.

Generally, analog signals must be laid in shielded cables. The cable shields must be grounded at both sides of the cables. In order to avoid unacceptable potential differences between different parts of the installation, low resistance equipotential bonding conductors must be laid.


Fig. 242: Connection of the Ethernet bus module CI501-PNIO
Chapter 6.8.4.2 "PROFINET communication interface module" on page 4596

## Connection of the digital inputs



Fig. 243: Connection of the digital inputs (DIO ... DI7)
${ }^{\circledR}$ ¿ Chapter 5.2.6.7.2.10 "State LEDs" on page 1233

## Connection of the digital outputs



Fig. 244: Connection of the digital output (DOO ... DO7)

* Chapter 5.2.6.7.2.10 "State LEDs" on page 1233


## Connection of resistance thermometers in 2-wire configuration to the analog inputs

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the module CI501-PNIO provides a constant current source which is multiplexed over the max. 4 analog input channels.


Fig. 245: Connection of resistance thermometers in 2-wire configuration to the analog inputs (AIO ... Al3)

Table 265: Configurable measuring ranges

| Pt100 | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 2-wire configuration, 1 <br> channel used |
| :--- | :--- | :--- |
| Pt1000 | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 2-wire configuration, 1 <br> channel used |
| Ni1000 | $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ | 2-wire configuration, 1 <br> channel used |

« Chapter 5.2.6.7.2.8 "Parameterization" on page 1222

* ${ }^{2}$ Chapter 5.2.6.7.2.9 "Diagnosis" on page 1228

The module CI501-PNIO performs a linearization of the resistance characteristic.
To avoid error messages from unused analog input channels, configure them as "unused".

## Connection of resistance thermometers in 3-wire configuration to the analog inputs

When resistance thermometers (Pt100, Pt1000, Ni1000) are used, a constant current must flow through them to build the necessary voltage drop for the evaluation. For this, the module CI501-PNIO provides a constant current source which is multiplexed over the max. 4 analog input channels.


Fig. 246: Connection of resistance thermometers in 3-wire configuration to the analog inputs (AIO ... Al3)
With 3-wire configuration, 2 adjacent analog channels belong together (e. g. the channels 0 and 1). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0), the next higher address must be the odd address (channel 1).

The constant current of one channel flows through the resistance thermometer. The constant current of the other channel flows through one of the cores. The module calculates the measured value from the two voltage drops and stores it under the input with the higher channel number (e. g. I1).
In order to keep measuring errors as small as possible, it is necessary to have all the involved conductors in the same cable. All the conductors must have the same cross section.

Table 266: Configurable measuring ranges

| Pt100 | $-50^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ | 3-wire configuration, 2 chan- <br> nels used |
| :--- | :--- | :--- |
| Pt100 | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 3-wire configuration, 2 chan- <br> nels used |
| Pt1000 | $-50^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ | 3-wire configuration, 2 chan- <br> nels used |
| Ni1000 | $-50^{\circ} \mathrm{C} \ldots+150^{\circ} \mathrm{C}$ | 3-wire configuration, 2 chan- <br> nels used |

* Chapter 5.2.6.7.2.8 "Parameterization" on page 1222

② Chapter 5.2.6.7.2.9 "Diagnosis" on page 1228
The module CI501-PNIO performs a linearization of the resistance characteristic.
To avoid error messages from unused analog input channels, configure them as "unused".

## Connection of active-type analog sensors (Voltage) with galvanically isolated power supply to the analog inputs



Fig. 247: Connection of active-type analog sensors (voltage) with galvanically isolated power supply to the analog inputs (AIO ... AI3)

Table 267: Configurable measuring ranges

| Voltage | $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ | 1 channel used |
| :--- | :--- | :--- |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | 1 channel used |

Chapter 5.2.6.7.2.8 "Parameterization" on page 1222

* ${ }^{\text {² }}$ Chapter 5.2.6.7.2.9 "Diagnosis" on page 1228

To avoid error messages from unused analog input channels, configure them as "unused".

Connection of active-type analog sensors (Current) with galvanically isolated power supply to the analog inputs


Fig. 248: Connection of active-type analog sensors (current) with galvanically isolated power supply to the analog inputs (AIO ... AI3)

Table 268: Configurable measuring ranges

| Current | $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 1 channel used |
| :--- | :--- | :--- |
| Current | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 1 channel used |

\# Chapter 5.2.6.7.2.8 "Parameterization" on page 1222
\# Chapter 5.2.6.7.2.9 "Diagnosis" on page 1228
Unused input channels can be left open-circuited, because they are of low resistance.
To avoid error messages through unused analog input channels in measuring range $4 \mathrm{~mA} . . .20 \mathrm{~mA}$, these channels should be configured as "Not used".

## Connection of active-type analog sensors (Voltage) with no galvanically isolated power supply to the analog inputs



Fig. 249: Connection of active-type analog sensors (voltage) with no galvanically isolated power supply to the analog inputs (AIO ... AI3)

## CAUTION!

Risk of faulty measurements!
The negative pin at the sensors must not have too big a potential difference with respect to ZP (max. $\pm 1 \mathrm{~V}$ ).
Make sure that the potential difference never exceeds $\pm 1 \mathrm{~V}$ (also not with long cable lengths).

Table 269: Configurable measuring ranges

| Voltage | $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ | 1 channel used |
| :--- | :--- | :--- |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | 1 channel used |

* ${ }^{*}$ Chapter 5.2.6.7.2.8 "Parameterization" on page 1222
« Chapter 5.2.6.7.2.8 "Parameterization" on page 1222
(z) Chapter 5.2.6.7.2.9 "Diagnosis" on page 1228

To avoid error messages from unused analog input channels, configure them as "unused".

## Connection of passive-type analog sensors (Current) to the analog inputs



Fig. 250: Connection of passive-type analog sensors (current) to the analog inputs (AIO ... AI3)

Table 270: Configurable measuring ranges

| Current | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | 1 channel used |
| :--- | :--- | :--- |

${ }^{*}$ Chapter 5.2.6.7.2.8 "Parameterization" on page 1222

* $\boldsymbol{y}^{\prime}$ Chapter 5.2.6.7.2.9 "Diagnosis" on page 1228


## CAUTION!

## Risk of overloading the analog input!

If an analog current sensor supplies more than 25 mA for more than 1 second during initialization, this input is switched off by the module (input protection).

Use only sensors with fast initialization or without current peaks higher than 25 mA . If not possible, connect a 10 -volt zener diode in parallel to Alx+ and ZP.

Unused input channels can be left open-circuited, because they are of low resistance.
To avoid error messages through unused analog input channels in measuring range $4 \mathrm{~mA} . . .20 \mathrm{~mA}$, these channels should be configured as "Not used".

## Connection of active-type analog sensors (Voltage) to differential analog inputs

Differential inputs are very useful, if analog sensors are used which are remotely non-isolated (e.g. the minus terminal is remotely grounded).

The evaluation using differential inputs helps to considerably increase the measuring accuracy and to avoid ground loops.

With differential input configurations, two adjacent analog channels belong together (e.g. the channels 0 and 1). In this case, both channels are configured according to the desired operating mode. The lower address must be the even address (channel 0), the next higher address must be the odd address (channel 1). The converted analog value is available at the higher address (channel 1).
The analog value is calculated by subtraction of the input value with the higher address from the input value of the lower address.

The converted analog value is available at the odd channel (higher address).

## CAUTION!

## Risk of faulty measurements!

The negative pin at the sensors must not have too big a potential difference with respect to ZP (max. $\pm 1 \mathrm{~V}$ ).

Make sure that the potential difference never exceeds $\pm 1 \mathrm{~V}$.


Fig. 251: Connection of active-type analog sensors (voltage) to differential analog inputs (AIO ... Al3)

Table 271: Configurable measuring ranges

| Voltage | $0 \mathrm{~V} \ldots 10 \mathrm{~V}$ | With differential inputs, 2 <br> channels used |
| :--- | :--- | :--- |
| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | With differential inputs, 2 <br> channels used |

\# Chapter 5.2.6.7.2.8 "Parameterization" on page 1222
(3) Chapter 5.2.6.7.2.9 "Diagnosis" on page 1228

To avoid error messages from unused analog input channels, configure them as "unused".

## Use of analog inputs as digital inputs

Several (or all) analog inputs can be configured as digital inputs. The inputs are not galvanically isolated against the other analog channels.


Fig. 252: Connection of digital sensors to the analog inputs (AIO ... AI3)

Table 272: Configurable measuring ranges

| Digital input | 24 V | 1 channel used |
| :--- | :--- | :--- |
| Effect of incorrect input ter- <br> minal connection |  | Wrong or no signal detected, <br> no damage up to 35 V |

${ }^{4}$ Chapter 5.2.6.7.2.8 "Parameterization" on page 1222
« Chapter 5.2.6.7.2.9 "Diagnosis" on page 1228

## Connection of analog output loads (Voltage)



Fig. 253: Connection of analog output loads (voltage) to the analog outputs (AO0 ... AO1)

Table 273: Configurable measuring ranges

| Voltage | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ | Load $\pm 10 \mathrm{~mA}$ max. | 1 channel used |
| :--- | :--- | :--- | :--- |

を Chapter 5.2.6.7.2.8 "Parameterization" on page 1222

* ${ }^{*}$ Chapter 5.2.6.7.2.9 "Diagnosis" on page 1228

Unused analog outputs can be left open-circuited.

## Connection of analog output loads (Current)



Fig. 254: Connection of analog output loads (current) to the analog otputs (AOO and AO1)

Table 274: Configurable measuring ranges

| Current | $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | Load $0 \Omega \ldots 500 \Omega$ | 1 channel used |
| :--- | :--- | :--- | :--- |
| Current | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ | Load $0 \Omega \ldots 500 \Omega$ | 1 channel used |

』 ${ }^{\wedge}$ Chapter 5.2.6.7.2.8 "Parameterization" on page 1222
(3) Chapter 5.2.6.7.2.9 "Diagnosis" on page 1228

Unused analog outputs can be left open-circuited.

## Assignment of the Ethernet ports

The terminal unit for the communication interface module provides two Ethernet interfaces with the following pin assignment:

Pin assignment

| Interface | Pin | Signal | Description |
| :---: | :---: | :---: | :---: |
|  | 1 | TxD+ | Transmit data + |
|  | 2 | TxD- | Transmit data - |
|  | 3 | RxD+ | Receive data + |
|  | 4 | NC | Not connected |
|  | 5 | NC | Not connected |
|  | 6 | RxD- | Receive data - |
|  | 7 | NC | Not connected |


| Interface | Pin | Signal | Description |
| :--- | :--- | :--- | :--- |
|  | 8 | NC | Not connected |
|  | Shield | Cable shield | Functional earth |

In corrosive environment, please protect unused connectors using the TA535 accessory.
Not supplied with this device.
§ Further information about wiring and cable types

Internal data exchange

| Parameter | Value |
| :--- | :--- |
| Digital inputs (bytes) | 3 |
| Digital outputs (bytes) | 3 |
| Analog inputs (words) | 4 |
| Analog outputs (words) | 2 |
| Counter input data (words) | 4 |
| Counter output data (words) | 8 |

## Addressing

The module has 2 rotary switches to set an explicit name to the PROFINET IO device before commissioning. No engineering tool is needed in this case.
The device gets its name (including the fixed part of the device name) with the switch settings ( $01 \mathrm{~h} . . . \mathrm{FFh}$ ). This name can be used directly within the device configuration: "CI5xx-pn-yy"
"CI5xx-pn-yy" xx is the fixed part of the device name (e.g. CI501) and yy represents the position of the rotary switch (0..FFh). The rotary switch values must be entered in hexadecimal format. For example, to set the name to "Cl5xx-pn-08", set the upper rotary switch to "0" and the lower switch to " 8 ".

The module reads the position of the rotary switches only during power-up, i.e. changes of the switch position during operation will have no effect until the next module initialization.

## I/O configuration

The CI501-PNIO stores some PROFINET configuration parameters (I/O device identifier, I/O device type and IP address configuration). No more configuration data is stored.
The analog/digital I/O channels are configured via software.

* $\boldsymbol{y}^{\boldsymbol{y}}$ Chapter 5.2.6.7.2.8 "Parameterization" on page 1222


## Parameterization

## Parameters of the module

| Name | Value | Internal value | Internal value, type | Default |
| :---: | :---: | :---: | :---: | :---: |
| Module ID ${ }^{1}$ ) | Internal | 7000 | WORD | 7000 |
| Parameter length | Internal | 25 | BYTE | 25 |
| Error LED / Failsafe function see table Error LED / Failsafe function (y) Table 275 "Err or LED / Failsafe function" on page 1223 | On | 0 | BYTE | 0 |
|  | Off by E4 | 1 |  |  |
|  | Off by E3 | 3 |  |  |
|  | On + failsafe | 16 |  |  |
|  | Off by E4 + failsafe | 17 |  |  |
|  | Off by E3 + failsafe | 19 |  |  |
| Process cycle time ${ }^{2}$ ) | 1 ms process cycle time | 1 | BYTE | 1 ms |
|  | 2 ms process cycle time | 2 |  |  |
|  | 3 ms process cycle time | 3 |  |  |
|  | 4 ms process cycle time | 4 |  |  |
|  | 5 ms process cycle time | 5 |  |  |
|  | 6 ms process cycle time | 6 |  |  |
|  | 7 ms process cycle time | 7 |  |  |
|  | 8 ms process cycle time | 8 |  |  |
|  | 9 ms process cycle time | 9 |  |  |
|  | 10 ms process cycle time | 10 |  |  |
|  | 11 ms process cycle time | 11 |  |  |
|  | 12 ms process cycle time | 12 |  |  |
|  | 13 ms process cycle time | 13 |  |  |
|  | 14 ms process cycle time | 14 |  |  |
|  | 15 ms process cycle time | 15 |  |  |
|  | 16 ms process cycle time | 16 |  |  |
| Check supply | $\begin{array}{\|l\|} \hline \text { off } \\ \text { on } \end{array}$ | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | BYTE | 1 |


| Name | Value | Internal value | Internal value, type | Default |
| :---: | :---: | :---: | :---: | :---: |
| Input delay | 8 ms | 8 ms | BYTE | 8 ms |
| Fast counter | $\left.\begin{array}{ll} 0 \\ : \\ 10 & \\ 3 \end{array}\right)$ | $\begin{aligned} & 0 \\ & : \\ & 10 \end{aligned}$ | BYTE | 0 |
| Detect short circuit at outputs | On | 1 | BYTE | On |
| Behavior digital outputs at comm. error | Off | 0 | BYTE | Off |
| Substitute value digital outputs | 0 | $0 . .255$ | BYTE | 0 |
| Overvoltage behavior on output | Off | 0 | BYTE | Off |
| Behavior analog outputs atcomm. error | Off | 0 | BYTE | Off |
| I/O-Bus reset | Off | 0 | BYTE | Off |
|  | On | 1 | BYTE | Off |

## Remarks:

| ${ }^{1}$ ) | With a faulty ID, the modules reports a "parameter error" and does not perform cyclic process data transmission. |
| :---: | :---: |
| ${ }^{2}$ ) | As for device index C0 the parameter is no longer evaluated. |
| ${ }^{3}$ ) | Counter operating modes, see description of the Fast counter ${ }^{4} \boldsymbol{y}$ Chapter 5.2.4.3.2.9 "Fast counter" on page 606. |

Table 275: Error LED / Failsafe function

| Setting | Description |
| :--- | :--- |
| On | Error LED (S-ERR) lights up at errors of all <br> error classes, Failsafe-mode off |
| Off by E4 | Error LED (S-ERR) lights up at errors of error <br> classes E1, E2 and E3, Failsafe-mode off |
| Off by E3 | Error LED (S-ERR) lights up at errors of error <br> classes E1 and E2, Failsafe-mode off |
| On +Failsafe | Error LED (S-ERR) lights up at errors of all <br> error classes, Failsafe-mode on *) |
| Off by E4 + Failsafe | Error LED (S-ERR) lights up at errors of error <br> classes E1, E2 and E3, Failsafe-mode on *) |
| Off by E3 + Failsafe | Error LED (S-ERR) lights up at errors of error <br> classes E1 and E2, Failsafe-mode on *) |
| *) The parameters Behaviour AO at comm. error and Behaviour DO at comm. error are only |  |
| analyzed if the Failsafe-mode is ON. |  |

IO-BUS reset after PROFINET reconnection

IO-BUS reset after PROFINET reconnection controls the behavior of PROFINET CI modules in relation to connected I/O modules (both safety and non-safety I/O modules).

- IO-BUS reset after PROFINET reconnection = "On" resets and, thus, re-parameterizes all attached I/O modules. All internal I/O modules states are reset, including the related diagnosis information.
Note that if the parameter is set to "On" then:
- The bumpless re-start of non-safety I/O modules will not be supported. It means, for example, that non-safety output channels will go from fail-safe values to " 0 " values during the re-connection and re-parameterization time and after that go to new output values.
- Safety I/O modules will be re-parameterized and re-started as newly started modules, which may not require their PROFIsafe reintegration, depending on safety CPU state, in the safety application.
- IO-BUS reset after PROFINET reconnection = "Off" will not reset all attached I/O modules. It will re-parameterize I/O modules only if parameter change is detected during the reconnection. All internal I/O modules states are not reset, including the related diagnosis information.
Note that if the parameter is set to "Off" then:
- The bumpless re-start of non-safety I/O modules is supported (if no parameters are changed). It means, for example, that non-safety output channels will not go from failsafe values to " 0 " values during the re-connection and re-parameterization time, but directly from fail-safe values to new output values.
- Safety I/O modules will not be re-parameterized (if no parameters are changed). Thus, they may continue their operation, which may require their PROFIsafe reintegration in the safety application on the safety CPU, e.g., if PROFIsafe watchdog time for this safety I/O module has expired. Any reintegration of such safety I/O modules will be not only application specific but also PROFIsafe specific and depend on the safety I/O handling in the safety application.


## Group parameters for the analog part

| Name | Value | Internal value | Internal value, <br> type | Default |
| :--- | :--- | :--- | :--- | :--- |
| Analog data <br> format | Standard <br> Reserved | 255 | BYTE | 0 |
| Behaviour AO at <br> comm. error *) | Off | Last value | 1 | BYTE |
|  | Last value 5 s | 6 | 0 |  |
|  | Last value 10 s | 11 |  |  |
|  | Substitute value | 2 |  |  |
|  | Substitute value <br> 5 s <br> Substitute value <br> $10 ~ s ~$ | 12 |  |  |

## Channel parameters for the analog inputs (4x)

| Name | Value | Internal value | Internal value, type | Default |
| :---: | :---: | :---: | :---: | :---: |
| Input 0, Channel configuration | Table Operating modes of the analog inputs (4) Table 276 "Ch annel configuration" on page 1225 | Table Operating modes of the analog inputs *) Table 276 "Ch annel configuration" on page 1225 | BYTE | 0 |
| Input 0, Check channel | Table Channel montoring Table 277 "Ch annel monitoring" on page 1226 | Table Channel montoring ② Table 277 "Ch annel monitoring" on page 1226 | BYTE | 0 |
| : | : | : | . | : |
| : | : | : | : | : |
| Input 3, Channel configuration | Table Operating modes of the analog inputs (7) Table 276 "Ch annel configuration" on page 1225 | Table Operating modes of the analog inputs Table 276 "Ch annel configuration" <br> on page 1225 | BYTE | 0 |
| Input 3, Check channel | Table Channel montoring ③ Table 277 "Ch annel monitoring" on page 1226 | Table Channel montoring ③ Table 277 "Ch annel monitoring" on page 1226 | BYTE | 0 |

Table 276: Channel configuration

| Internal value | Operating modes of the analog inputs, individually configurable |
| :---: | :---: |
| 0 (default) | Not used |
| 1 | 0 V ... 10 V |
| 2 | Digital input |
| 3 | 0 mA ... 20 mA |
| 4 | $4 \mathrm{~mA} . . .20 \mathrm{~mA}$ |
| 5 | -10 V ... +10 V |
| 8 | 2-wire Pt100-50 ${ }^{\circ} \mathrm{C} \ldots+400{ }^{\circ} \mathrm{C}$ |
| 9 | 3-wire Pt100-50 ${ }^{\circ} \mathrm{C} \ldots+40{ }^{\circ} \mathrm{C}{ }^{*}$ ) |
| 10 | 0 V ... 10 V (voltage diff.) *) |
| 11 | -10 V ... +10 V (voltage diff.) *) |
| 14 | 2-wire Pt100-50 ${ }^{\circ} \mathrm{C} \ldots+70{ }^{\circ} \mathrm{C}$ |
| 15 | 3-wire Pt100-50 ${ }^{\circ} \mathrm{C} \ldots+70{ }^{\circ} \mathrm{C}$ *) |
| 16 | 2-wire Pt1000-50 ${ }^{\circ} \mathrm{C} \ldots+400{ }^{\circ} \mathrm{C}$ |
| 17 | 3-wire Pt1000-50 ${ }^{\circ} \mathrm{C} \ldots+400^{\circ} \mathrm{C}$ *) |
| 18 | 2-wire Ni1000-50 ${ }^{\circ} \mathrm{C} \ldots+150{ }^{\circ} \mathrm{C}$ |


| Internal value | Operating modes of the analog inputs, individually configurable |
| :--- | :--- |
| 19 | 3-wire Ni1000 $-50^{\circ} \mathrm{C} \ldots+150{ }^{\circ} \mathrm{C}{ }^{*}$ ) |
| ${ }^{*}$ ) In the operating modes with 3-wire configuration or with differential inputs, two adjacent |  |
| analog inputs belong together (e.g. the channels 0 and 1 ). In these cases, both channels |  |
| are configured in the desired operating mode. The lower address must be the even address |  |
| (channel 0). The next higher address must be the odd address (channel 1). The converted |  |
| analog value is available at the higher address (channel 1 ). |  |

Table 277: Channel monitoring

| Internal Value | Check Channel |
| :--- | :--- |
| 0 (default) | Plausibility, wire break, short circuit |
| 3 | Not used |

## Channel parameters for the analog outputs (2x)

| Name | Value | Internal value | Internal value, type | Default |
| :---: | :---: | :---: | :---: | :---: |
| Output 0, <br> Channel configuration | Table Operating modes of the analog outputs ②) Further information on page 1227 | Table Operating modes of the analog outputs \& Further information on page 1227 | BYTE | 0 |
| Output 0, Check channel | Table Channel monitoring ̌ Table 279 "Ch annel monitoring" on page 1227 | Table Channel monitoring ② Table 279 "Ch annel monitoring" on page 1227 | BYTE | 0 |
| Output 0, Substitute value | Table Substitute value をy Table 280 "Su bstitute value" on page 1227 | Table Substitute value ③ Table 280 "Su bstitute value" on page 1227 | WORD | 0 |
| Output 1, <br> Channel configuration | Table Operating modes of the analog outputs をy Further information on page 1227 | Table Operating modes of the analog outputs *) Further information on page 1227 | BYTE | 0 |
| Output 1, Check channel | Table Channel monitoring *s Table 279 "Ch annel monitoring" on page 1227 | Table Channel monitoring ⓨ Table 279 "Ch annel monitoring" on page 1227 | BYTE | 0 |
| Output 1, Substitute value | Table Substitute value <br> ③ Table 280 "Su bstitute value" on page 1227 | Table Substitute value <br> Table 280 "Su bstitute value" on page 1227 | WORD | 0 |

Table 278: Channel configuration

| Internal value | Operating modes of the analog outputs, individually configu- <br> rable |
| :--- | :--- |
| 0 (default) | Not used |
| 128 | $-10 \mathrm{~V} \ldots+10 \mathrm{~V}$ |
| 129 | $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ |
| 130 | $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}$ |

Table 279: Channel monitoring

| Internal value | Check channel |
| :--- | :--- |
| 0 | Plausibility, wire break, short circuit |
| 3 | None |

Table 280: Substitute value

| Intended behavior of output <br> channel when the control <br> system stops | Required setting of the <br> module parameter "Behav- <br> iour of outputs in case of a <br> communication error" | Required setting of the <br> channel parameter "Substi- <br> tute value" |
| :--- | :--- | :--- |
| Output OFF | Off | 0 |
| Last value infinite | Last value | 0 |
| Last value for 5 s and then <br> turn off | Last value 5 sec | 0 |
| Last value for 10 s and then <br> turn off | Last value 10 sec | 0 |
| Substitute value infinite | Substitute value | Depending on configuration |
| Substitute value for 5 s and <br> then turn off | Substitute value 5 sec | Depending on configuration |
| Substitute value for 10 s and <br> then turn off | Substitute value 10 sec | Depending on configuration |

## Group parameters for the digital part

| Name | Value | Internal value | Internal value, <br> type | Default |
| :--- | :--- | :--- | :--- | :--- |
| Input delay | 0.1 ms | 0 | BYTE | 0.1 ms |
|  | 1 ms | 1 |  | $0 \times 00$ |
|  | 8 ms | 2 | 3 |  |
| Detect short cir- <br> cuit at outputs | 32 ms | Off | On | 1 |


| Name | Value | Internal value | Internal value, <br> type | Default |
| :--- | :--- | :--- | :--- | :--- |
| Behaviour DO at <br> comm. error ${ }^{1}$ ) | Off <br> Last value <br> Last value 5 sec <br> Last value 10 sec <br> Substitute value <br> Substitute value | 11 <br> 5 <br> sec <br> Substitute value <br> 10 sec | 12 | BYTE |

## Diagnosis

Table 281: Structure of the diagnosis block via PNIO_DEV_ALARM function block

| Byte Number | Description | Possible Values |
| :--- | :--- | :--- |
| 1 | Diagnosis Byte, slot number | 31 = CI501-PNIO (e. g. error at inte- <br> grated 8 DI / 8 DO) <br> $1=1$ st connected S500 I/O module <br> $\ldots$ |
| $10=10$ th connected S500 I/O module |  |  |$|$| According to the I/O bus specification |
| :--- |
| passed on by modules to the fieldbus |
| master |


| Byte Number | Description | Possible Values |
| :--- | :--- | :--- |
| 4 | Diagnosis Byte, error code | According to the I/O bus specification <br> Bit 7 and bit 6, coded error class <br> $0=\mathrm{E} 1$ <br> $1=\mathrm{E} 2$ <br> $2=\mathrm{E} 3$ <br> $3=\mathrm{E} 4$ <br>  |
|  |  | Bit 0 to bit 5, coded error description |
| 5 | Diagnosis Byte, flags | According to the I/O bus specification <br> Bit 7: $1=$ coming error <br> Bit 6: $1=$ leaving error |

In cases of short circuit or overload, the digital outputs are turned off. The modules performs reactivation automatically. Thus an acknowledgement of the errors is not necessary. The error message is stored via the LED.

| E1 ... E4 | d1 | d2 | d3 | d4 | Identifier $000 \text {.. }$ | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err |  |  |
| $\begin{array}{\|l} \hline \text { Byte } 4 \\ \text { Bit } \\ 6 \text {... } 7 \end{array}$ | - | Byte 1 | Byte 2 | Byte 3 | $\begin{array}{\|l} \hline \text { Byte } 4 \\ \text { Bit } \\ 0 \text {... } 5 \end{array}$ |  |  |
| Class | Interface | Device | Module | Channel | Error- <br> Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | $\left.{ }^{3}\right)$ |  |  |  |  |
| Module errors |  |  |  |  |  |  |  |
| 3 | - | 31 | 31 | 31 | 19 | Checksum error in the I/O module | Replace I/O module |
| 3 | - | 31 | 31 | 31 | 3 | Timeout in the I/O module |  |
| 3 | - | 31 | 31 | 31 | 40 | Different hard-/firmware versions in the module |  |
| 3 | - | 31 | 31 | 31 | 43 | Internal error in the module |  |
| 3 | - | 31 | 31 | 31 | 36 | Internal data exchange failure |  |
| 3 | - | 31 | 31 | 31 | 9 | Overflow diagnosis buffer | Restart |
| 3 | - | 31 | 31 | 31 | 26 | Parameter error | Check master |


| E1 ... E4 | d1 | d2 | d3 | d4 | Identifier 000 ... | AC500Display | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err | $\begin{array}{\|l\|} \hline \text { PS501 } \\ \text { PLC } \\ \text { Browser } \end{array}$ |  |  |
|  | - | Byte 1 | Byte 2 | Byte 3 | $\begin{aligned} & \hline \text { Byte } 4 \\ & \text { Bit } \\ & 0 \text {... } 5 \end{aligned}$ | PNIO diagnosis block |  |  |
| Class | Interface | Device | Module | Channel | Error-Identifier | Error message |  | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) |  |  |  |  |  |
| 3 | - | 31 | 31 | 31 | 11 | Process voltage UP too low |  |  |
| 3 | - | 31 | 31 | 31 | 45 | No process voltage UP |  | Check process supply voltage |
| 3 | - | 31/1... 10 | 31 | 31 | 17 | No communication with I/O module |  | Replace I/O module |
| 3 | - | 1... 10 | 31 | 31 | 32 | Wrong I/O module type on socket |  | Replace <br> l/O <br> module / <br> Check <br> configu- <br> ration |
| 4 | - | 1... 10 | 31 | 31 | 31 | At least one module does not support failsafe function |  | Check modules and parame- terization |
| 4 | - | 1... 10 | 31 | 5 | 8 | I/O module removed from hot swap terminal unit or defective module on hot swap terminal unit ${ }^{9}$ ) |  | Plug I/O module, replace I/O module |
| 4 | - | 1... 10 | 31 | 5 | 28 | Wrong I/O module plugged on hot swap terminal unit ${ }^{9}$ ) |  | Remove <br> wrong <br> $1 / O$ <br> module <br> and plug <br> pro- <br> jected <br> $1 / 0$ <br> module |
| 4 | - | 1... 10 | 31 | 5 | 42 | No communication with I/O module on hot swap terminal unit ${ }^{9}$ ) |  | Replace I/O module |


| E1 ... E4 | d1 | d2 | d3 | d4 | Identifier 000 ... 0 | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err |  |  |
| Byte 4 <br> Bit <br> 6 ... 7 | - | Byte 1 | Byte 2 | Byte 3 | Byte 4 <br> Bit <br> 0 ... 5 |  |  |
| Class | Inter- face | Device | Module | Channel | Error- <br> Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) |  |  |  |  |
| 4 | - | 1... 10 | 31 | 5 | 54 | I/O module does not support hot swap $\left.{ }^{8}\right)^{9}$ ) | Power off system and replace I/O module |
| 4 | - | 1... 10 | 31 | 6 | 8 | Hot swap terminal unit configured but not found | Replace terminal unit by hot swap terminal unit |
| 4 | - | 1... 10 | 31 | 6 | 42 | No communication with hot swap terminal unit ${ }^{9}$ ) | Restart, if error persists replace terminal unit |
| 4 | - | 31 | 31 | 31 | 46 | Voltage feedback on activated digital outputs DO0...DO7 on UP3 ${ }^{4}$ ) | Check terminals |
| 4 | - | 31/1... 10 | 31 | 31 | 34 | No response during initialization of the I/O module | Replace I/O module |
| 4 | - | 31 | 31 | 31 | 11 | Process voltage UP3 too low | Check process supply voltage |
| 4 | 1... 6 | 255 | 2 | 0 | 45 | The connected Communication Module has no connection to the network | Check cabeling |
| 4 | - | 31 | 31 | 31 | 45 | No process voltage UP3 | Check process supply voltage |


| E1 ... E4 | d1 | d2 | d3 | d4 | Identifier $000 \ldots$ | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err |  |  |
| $\begin{array}{\|l} \hline \text { Byte } 4 \\ \text { Bit } \\ 6 \ldots 7 \end{array}$ | - | Byte 1 | Byte 2 | Byte 3 | $\begin{array}{\|l} \hline \text { Byte } 4 \\ \text { Bit } \\ 0 \text {... } 5 \end{array}$ |  |  |
| Class | Interface | Device | Module | Channel | Error- <br> Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | $\left.{ }^{3}\right)$ |  |  |  |  |
| 4 | - | 31 | 31 | 31 | 10 | Voltage overflow on outputs (above UP3 level) ${ }^{5}$ ) | Check terminals/ check process supply voltage |
| Channel error digital |  |  |  |  |  |  |  |
| 4 | - | 31 | 2 | 0... 7 | 46 | Externally voltage detected at digital output DO0...DO7 ${ }^{6}$ ) | Check terminals |
| 4 | - | 31 | 2 | 0... 7 | 47 | Short circuit at digital output ${ }^{7}$ ) | Check terminals |
| Channel error analog |  |  |  |  |  |  |  |
| 4 | - | 31 | 1 | 0... 3 | 48 | Analog value overflow or broken wire at an analog input | Check value or check terminals |
| 4 | - | 31 | 1 | 0... 3 | 7 | Analog value underflow at an analog input | Check value |
| 4 | - | 31 | 1 | 0... 3 | 47 | Short circuit at an analog input | Check terminals |
| 4 | - | 31 | 3 | 0... 1 | 4 | Analog value overflow at an analog output | Check output value |
| 4 | - | 31 | 3 | 0... 1 | 7 | Analog value underflow at an analog output | Check output value |

Remarks:

| ${ }^{1}$ ) | In AC500 the following interface identifier applies: <br> "-" = Diagnosis via bus-specific function blocks; 0 ... 4 or $10=$ Position of the communication module; $14=1 / \mathrm{O}$ bus; $31=$ Module itself <br> The identifier is not contained in the CI501-PNIO diagnosis block. |
| :---: | :---: |
| ${ }^{2}$ ) | With "Device" the following allocation applies: $31=$ Module itself; $1 \ldots 10=$ Expansion module |
| ${ }^{3}$ ) | With "Module" the following allocation applies: $31=$ Module itself Module type ( 1 = AI, 2 = DO, 3 = AO) |
| ${ }^{4}$ ) | This message appears, if externally voltages at one or more terminals DO0 ... DO7 cause that other digital outputs are supplied through that voltage ${ }^{\circledR}$ Chapter 5.2.6.7.2.4 "Connections" on page 1205. All outputs of the apply digital output groups will be turned off for 5 seconds. The diagnosis message appears for the whole output group. |
| ${ }^{5}$ ) | The voltage on digital outputs DO0 ... DO7 has overrun the process supply voltage UP3 ") Chapter 5.2.6.7.2.4 "Connections" on page 1205. Diagnosis message appears for the whole module. |
| ${ }^{6}$ ) | This message appears, if the output of a channel DO0 ... DO7 should be switched on while an externally voltage is connected. In this case the start up is disabled, as long as the externally voltage is connected. Otherwise this could produce reverse voltage from this output to other digital outputs. This diagnosis message appears per channel. |
| ${ }^{7}$ ) | Short circuit: After a detected short circuit, the output is deactivated for 100 ms . Then a new start up will be executed. This diagnosis message appears per channel. |
| ${ }^{8}$ ) | In case of an I/O module doesn't support hot swapping, do not perform any hot swap operations (also not on any other terminal units (slots)) as modules may be damaged or I/O bus communication may be disturbed. |
| ${ }^{9}$ ) | Diagnosis for hot swap available as of version index FO. |

## State LEDs

The LEDs are located at the front of module. There are 2 different groups:

- The 5 system LEDs (PWR, STA1 ETH, STA2 ETH, S-ERR and I/O-Bus) show the operation state of the module and display possible errors.
- The 27 process LEDs (UP, UP3, inputs, outputs, CH-ERR1 to CH-ERR3) show the process supply voltage and the states of the inputs and outputs and display possible errors.

Table 282: States of the 5 system LEDs

| LED | Color | OFF | ON | Flashing |
| :--- | :--- | :--- | :--- | :--- |
| PWR/RUN | Green | Process supply <br> voltage missing | Internal supply <br> voltage OK, <br> module ready for <br> communication <br> with I/O Con- <br> troller | Start-up / pre- <br> paring communi- <br> cation |
|  | Yellow | --- | --- | --- |
| STA1 ETH <br> (System LED <br> "BF") | Green | --- | Device config- <br> ured, cyclic data <br> exchange run- <br> ning | --- |


| LED | Color | OFF | ON | Flashing |
| :--- | :--- | :--- | :--- | :--- |
|  | Red | --- | --- | Device is not <br> configured |
| STA2 ETH <br> (System LED <br> "SF") | Green | --- | Got identification <br> request from I/O <br> controller |  |
|  | Red | Red | No system error | System error <br> (collective error) |
| S---- |  |  |  |  |
| I/O-Bus | Green | No error <br> modules con- <br> nected or com- <br> munication error | Expansion <br> modules con- <br> nected and <br> operational | --- |
| ETH1 | Green | No connection at <br> Ethernet inter- <br> face | Connected to <br> Ethernet inter- <br> face | ---- |
|  | Yellow | Green | No connection at <br> Ethernet inter- <br> face | Connected to <br> Ethernet inter- <br> face |
|  | Yellow | ---- |  |  |

Table 283: States of the 27 process LEDs

| LED | Color | OFF | ON | Flashing |
| :--- | :--- | :--- | :--- | :--- |
| AIO ... AI3 | Yellow | Input is OFF | Input is ON <br> (brightness <br> depends on the <br> value of the <br> analog signal) | -- |
| AO0 ... AO1 | Yellow | Output is OFF | Output is ON <br> (brightness <br> depends on the <br> value of the <br> analog signal) | -- |
| DI0 ... DI7 | Yellow | Input is OFF | Input is ON (the <br> input voltage is <br> even displayed if <br> the supply <br> voltage is OFF) | --- |
| DO0 ... DO7 | Yellow | Green | Output is OFF <br> Poltage missing | Output is ON |
| UP | Process supply <br> voltage OK and <br> initialization fin- <br> ished | -- |  |  |
| UP3 | Green | Process supply <br> voltage missing | Process supply <br> voltage OK | --- |
| CH-ERR1 to CH- <br> ERR3 | Red | No error or <br> process supply <br> voltage missing | Internal error | Error on one <br> channel of the <br> corresponding <br> group |

## Measuring ranges

Input ranges voltage, current and digital input

| Range | $0 \text { V ... +10 }$ | $-10 \mathrm{~V}$ | $0 \mathrm{~mA} \text {. }$ | $4 \mathrm{~mA} . . .20$ | Digital | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  | Decimal | Hex. |
| Overflow | > 11.7589 | > 11.7589 | > 23.5178 | > 22.8142 |  | 32767 | 7FFF |
| Measured value too high |  |  | 23.5178 $:$ 20.0007 |  |  | 32511 $:$ 27649 | $\begin{aligned} & \hline \text { 7EFF } \\ & \text { : } \\ & 6 \mathrm{CO1} \\ & \hline \end{aligned}$ |
| Normal range <br> Normal | 10.0000 $:$ 0.0004 | $\begin{aligned} & 10.0000 \\ & : \\ & 0.0004 \end{aligned}$ |  | 20.0000 $:$ 4.0006 | On | 27648 | $\begin{aligned} & \text { 6C00 } \\ & : \\ & 0001 \end{aligned}$ |
| measured | 0.0000 | 0.0000 | 0 | 4 | Off | 0 | 0000 |
| value too low | $\begin{array}{\|l\|} \hline-0.0004 \\ -1.7593 \end{array}$ | $\begin{aligned} & \hline-0.0004 \\ & : \\ & : \\ & -10,0000 \end{aligned}$ |  | $\begin{aligned} & 3.9994 \\ & 1.1858 \end{aligned}$ |  | -1 <br> -4864 <br> -27648 | $\begin{array}{\|l} \text { FFFF } \\ \text { ED00 } \\ : \\ 9400 \end{array}$ |
| Measured value too low |  | -10.0004 <br> -11.7589 |  |  |  | -27649 $:$ -32512 | $\begin{aligned} & \text { 93FF } \\ & : \\ & 8100 \\ & \hline \end{aligned}$ |
| Underflow | < 1.7593 | <-11.7589 | < 0.0000 | < 1.1858 |  | -32768 | 8000 |

The represented resolution corresponds to 16 bits.

Input ranges resistance temperature detector

| Range | $\begin{aligned} & \text { Pt100 / Pt1000 } \\ & -50 \ldots+70^{\circ} \mathrm{C} \end{aligned}$ | $\begin{array}{\|l} \hline \text { Pt100 I } \\ \text { Pt1000 } \\ -50 \ldots+400 \\ { }^{\circ} \mathrm{C} \end{array}$ | $\begin{aligned} & \text { Ni1000 } \\ & -50 \ldots+150 \\ & { }^{\circ} \mathrm{C} \mathrm{C} \end{aligned}$ | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  | Decimal | Hex. |
| Overflow | $>+80.0^{\circ} \mathrm{C}$ | $>+450.0{ }^{\circ} \mathrm{C}$ | > +160.0 ${ }^{\circ} \mathrm{C}$ | 32767 | 7FFF |
| Measured value too high | $+80.0{ }^{\circ} \mathrm{C}$ | $\begin{aligned} & +450.0^{\circ} \mathrm{C} \\ & \vdots \\ & +400.1^{\circ} \mathrm{C} \end{aligned}$ |  | $\begin{aligned} & 4500 \\ & : \\ & 4001 \end{aligned}$ | $\begin{aligned} & 1194 \\ & : \\ & \text { 0FA1 } \end{aligned}$ |
|  |  |  | $\begin{aligned} & +160.0^{\circ} \mathrm{C} \\ & : \\ & +150.1^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & 1600 \\ & : \\ & 1501 \end{aligned}$ | $\begin{aligned} & 0640 \\ & : \\ & \text { 05DD } \end{aligned}$ |
| Normal range |  | $\begin{aligned} & +400.0^{\circ} \mathrm{C} \\ & \vdots \\ & \vdots \\ & \vdots \\ & +0.1^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & +150.0^{\circ} \mathrm{C} \\ & : \\ & : \\ & +0.1^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & 800 \\ & : \\ & 701 \end{aligned}$ | $\begin{aligned} & \hline 0320 \\ & : \\ & \text { 02BD } \end{aligned}$ |


| Range | $\begin{aligned} & \text { Pt100 / Pt1000 } \\ & -50 \ldots+70^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \hline \mathrm{Pt} 100 \text { I } \\ & \mathrm{Pt} 1000 \\ & -50 \ldots+400 \\ & { }^{\circ} \mathrm{C} \mathrm{C} \end{aligned}$ | $\begin{aligned} & \hline \mathrm{Ni} 1000 \\ & -50 \ldots+150 \\ & { }^{\circ} \mathrm{C} \mathrm{C} \end{aligned}$ | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  | Decimal | Hex. |
|  |  | $0.0^{\circ} \mathrm{C}$ | $0.0{ }^{\circ} \mathrm{C}$ | $\begin{aligned} & \hline 4000 \\ & 1500 \\ & 700 \\ & : \\ & 1 \end{aligned}$ | $\begin{aligned} & \hline \text { 0FAO } \\ & \text { 05DC } \\ & \text { 02BC } \\ & : \\ & 0001 \end{aligned}$ |
|  |  | $-0.1^{\circ} \mathrm{C}$ <br> $-50.0^{\circ} \mathrm{C}$ | $-0.1^{\circ} \mathrm{C}$ <br> $-50.0^{\circ} \mathrm{C}$ | 0 | 0000 |
| Measured value too low | $<-60.0{ }^{\circ} \mathrm{C}$ | $\begin{aligned} & -50.1^{\circ} \mathrm{C} \\ & : \\ & -60.0^{\circ} \mathrm{C} \end{aligned}$ | $\begin{aligned} & -50.1^{\circ} \mathrm{C} \\ & : \\ & -60.0^{\circ} \mathrm{C} \end{aligned}$ | $-1$ -500 | $\begin{aligned} & \text { FFFF } \\ & \text { : } \\ & \text { FEOC } \end{aligned}$ |
| Underflow | <-60.0 ${ }^{\circ} \mathrm{C}$ | $<-60.0{ }^{\circ} \mathrm{C}$ | $<-60.0^{\circ} \mathrm{C}$ | $\begin{aligned} & \hline-501 \\ & : \\ & -600 \\ & \hline \end{aligned}$ | $\begin{aligned} & \hline \text { FEOB } \\ & : \\ & \text { FDA8 } \\ & \hline \end{aligned}$ |

## Output ranges voltage and current

| Range | -10...+10 V | 0... 20 mA | 4... 20 mA | Digital value |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  | Decimal | Hex. |
| Overflow | > 11.7589 V | $\begin{aligned} & >23.5178 \\ & \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & >22.8142 \\ & \mathrm{~mA} \end{aligned}$ | > 32511 | > 7EFF |
| Measured value too high | $\begin{aligned} & 11.7589 \mathrm{~V} \\ & : \\ & 10.0004 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 23.5178 \mathrm{~mA} \\ & : \\ & 20.0007 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 22.8142 \mathrm{~mA} \\ & : \\ & 20.0006 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 32511 \\ & : \\ & 27649 \end{aligned}$ | $\begin{aligned} & \text { 7EFF } \\ & : \\ & 6 \mathrm{C} 01 \end{aligned}$ |
| Normal range | $\begin{aligned} & 10.0000 \mathrm{~V} \\ & : \\ & 0.0004 \mathrm{~V} \end{aligned}$ | $\begin{aligned} & 20.0000 \mathrm{~mA} \\ & : \\ & 0.0007 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 20.0000 \mathrm{~mA} \\ & : \\ & 4.0006 \mathrm{~mA} \end{aligned}$ | $27648$ <br> 1 | $\begin{aligned} & \text { 6C00 } \\ & \text { : } \\ & 0001 \end{aligned}$ |
|  | 0.0000 V | 0.0000 mA | 4.0000 mA | 0 | 0000 |
|  | $-0.0004 \mathrm{~V}$ -10.0000 V | $\begin{aligned} & 0 \mathrm{~mA} \\ & : \\ & 0 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 3.9994 \mathrm{~mA} \\ & 0 \mathrm{~mA} \\ & 0 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & \hline-1 \\ & -6912 \\ & -27648 \end{aligned}$ | $\begin{aligned} & \text { FFFF } \\ & \text { E500 } \\ & 9400 \end{aligned}$ |
| Measured value too low | $-10.0004 \mathrm{~V}$ $-11.7589 \mathrm{~V}$ | $\begin{aligned} & 0 \mathrm{~mA} \\ & : \\ & 0 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & 0 \mathrm{~mA} \\ & : \\ & 0 \mathrm{~mA} \end{aligned}$ | $\begin{aligned} & -27649 \\ & : \\ & -32512 \end{aligned}$ | $\begin{aligned} & \text { 93FF } \\ & : \\ & 8100 \end{aligned}$ |
| Underflow | <-11.7589 V | 0 mA | 0 mA | <-32512 | < 8100 |

The represented resolution corresponds to 16 bits.

## Technical data

## Technical data of the module

The system data of AC500 and S500 are applicable to the standard version Chapter 5.1.2 "System data AC500" on page 166.

The system data of AC500-XC are applicable to the XC version ${ }^{\circledR}$ Chapter 5.1.3 "System data AC500-XC" on page 169.

Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter | Value |
| :---: | :---: |
| Process supply voltages UP/UP3 |  |
| Rated value | 24 V DC (for inputs and outputs) |
| Max. load for the terminals | 10 A |
| Protection against reversed voltage | Yes |
| Rated protection fuse on UP/UP3 | 10 A fast |
| Galvanic isolation | Ethernet interface against the rest of the module |
| Inrush current from UP (at power up) | On request |
| Current consumption via UP (normal operation) | 0.2 A |
| Current consumption via UP3 | 0.06 A + 0.5 A max. per output |
| Connections | Terminals 1.8 and 2.8 for +24 V (UP) <br> Terminal 3.8 for +24 V (UP3) <br> Terminals 1.9, 2.9 and 3.9 for 0 V (ZP) |
| Max. power dissipation within the module | 6 W |
| Number of digital inputs | 8 |
| Number of digital outputs | 8 |
| Number of analog inputs | 4 |
| Number of analog outputs | 2 |
| Input data length | 19 bytes |
| Output data length | 23 bytes |
| Reference potential for all digital inputs and outputs | Negative terminal of the supply voltage, signal name ZP |
| Setting of the I/O device identifier | With 2 rotary switches at the front side of the module |
| Diagnose | See Diagnosis and Displays ${ }^{4} \Rightarrow$ Chapter 5.2.6.7.2.9 "Diagnosis" on page 1228 |
| Operation and error displays | 32 LEDs (totally) |
| Weight (without terminal unit) | Ca. 125 g |
| Mounting position | Horizontal or vertical with derating (output load reduced to $50 \%$ at $+40^{\circ} \mathrm{C}$ per group) |
| Extended ambient temperature (XC version) | >+60 ${ }^{\circ} \mathrm{C}$ on request |
| Cooling | The natural convection cooling must not be hindered by cable ducts or other parts in the control cabinet. |

## NOTICE!

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and temporary overvoltage up to 30 V DC.

## Multiple overloads

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an internal smart high-side switch.

| Parameter | Value |
| :---: | :---: |
| Bus connection | $2 \times \mathrm{RJ} 45$ |
| Switch | Integrated |
| Technology | Hilscher NETX 100 |
| Transfer rate | 10/100 Mbit/s (full-duplex) |
| Transfer method | According to Ethernet II, IEEE 802.3 |
| Ethernet | 100 base-TX, internal switch, 2x RJ45 socket |
| Expandability | Max. 10 S500 I/O modules |
| Adjusting elements | 2 rotary switches for generation of an explicit name |
| Supported protocols | RTC - real time cyclic protocol, class $1^{*}$ ) <br> RTA - real time acyclic protocol <br> DCP - discovery and configuration protocol <br> CL-RPC - connectionless remote procedure Call <br> LLDP - link layer discovery protocol <br> MRP - MRP Client |
| Acyclic services | PNIO read / write sequence (max. 1024 bytes per telegram) <br> Process-Alarm service |
| Supported alarm types | Process Alarm, Diagnostic Alarm, Return of SubModule, Plug Alarm, Pull Alarm |
| Min. bus cycle | 1 ms |
| Conformance class | CC A |
| Protective functions (according to IEC 61131-3) | Protected against: <br> - short circuit <br> - reverse supply <br> - overvoltage <br> - reverse polarity <br> Galvanic isolation from the rest of the module |

[^24]
## Technical data of the digital inputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 8 |
| Distribution of the channels into groups | 1 group of 8 channels |
| Terminals of the channels DI0 ... DI7 | Terminals $2.0 \ldots 2.7$ |
| Reference potential for all inputs | Terminals $1.9 \ldots 3.9$ (negative terminal of the <br> supply voltage, signal name ZP) |
| Indication of the input signals | 1 yellow LED per channel, the LED is ON when <br> the input signal is high (signal 1) |
| Input type (according EN 61131-2) | Type 1 |
| Input delay (0->1 or 1->0) | Typ. 0.1 ms, configurable from $0.1 \mathrm{~ms} \ldots 32 \mathrm{~ms}$ |
| Input signal voltage | 24 V DC |
|  | 0-Signal |
|  | $-3 \mathrm{~V} \ldots+5 \mathrm{~V}$ |
|  | 1-Signal |
| Ripple with signal 0 | $>+5 \mathrm{~V} \ldots<+15 \mathrm{~V}$ |
| Ripple with signal 1 | $+15 \mathrm{~V} \ldots+30 \mathrm{~V}$ |
| Input current per channel | Within $-3 \mathrm{~V} \ldots+5 \mathrm{~V}$ |
|  | Input voltage +24 V |
| Input voltage +5 V | Within $+15 \mathrm{~V} \ldots+30 \mathrm{~V}$ |
|  | Input voltage +15 V |
| Input voltage +30 V | $>1 \mathrm{~mA}$ |
| Max. cable length | $>2 \mathrm{~mA}$ |
|  | Shielded |
| Unshielded | $<8 \mathrm{~mA}$ |
|  | 600 m |

## Technical data of the digital outputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 8 |
| Distribution of the channels into groups | 1 group of 8 channels |
| Terminals of the channels DO0 ... DO7 | Terminals $3.0 \ldots 3.7$ |
| Reference potential for all outputs | Terminals $1.9 \ldots 3.9$ (negative terminal of the <br> supply voltage, signal name ZP) |
| Common power supply voltage | For all outputs terminal 3.8 (positive terminal of <br> the supply voltage, signal name UP3) |
| Output voltage for signal 1 | UP3 (-0.8 V) |
| Output delay (0->1 or 1->0) | On request |
| Output current | 500 mA at UP3 = 24 V |
|  | Rated value per channel |
| Max. value (all channels together) | 4 A |
| Leakage current with signal 0 | $<0.5 \mathrm{~mA}$ |


| Parameter | Value |
| :---: | :---: |
| Fuse for UP3 | 10 A fast |
| Demagnetization with inductive DC load | Via internal varistors (see figure below this table) |
| Output switching frequency |  |
| With resistive load | On request |
| With inductive loads | Max. 0.5 Hz |
| With lamp loads | 11 Hz max. at 5 W max. |
| Short-circuit-proof / overload-proof | Yes |
| Overload message ( $1>0.7$ A) | Yes, after ca. 100 ms |
| Output current limitation | Yes, automatic reactivation after short circuit/overload |
| Resistance to feedback against 24 V signals | Yes (software-controlled supervision) |
| Max. cable length |  |
| Shielded | 1000 m |
| Unshielded | 600 m |

The following drawing shows the circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off.


1 Digital output
2 Varistors for demagnetization when inductive loads are turned off

## Technical data of the analog inputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 4 |
| Distribution of channels into groups | 1 group with 4 channels |
| Connection if channels $\mathrm{Al0}+\ldots \mathrm{Al3+}$ | Terminals $1.0 \ldots 1.3$ |
| Reference potential for $\mathrm{Al0}+\ldots \mathrm{Al3+}$ | Terminal 1.4 (AI-) for voltage and RTD meas- <br> urement <br> Terminal 1.9, 2.9 and 3.9 for current measure- <br> ment |
| Input type | Voltage $0 \mathrm{~V} \ldots 10 \mathrm{~V}$, current or Pt100/Pt1000/ <br> Ni1000 |
| Unipolar | Vipolar |
| Galvanic isolation | Against Ethernet network |


| Parameter | Value |
| :--- | :--- |
| Configurability | $0 \mathrm{~V} \ldots 10 \mathrm{~V},-10 \mathrm{~V} \ldots+10 \mathrm{~V}, 0 \mathrm{~mA} \ldots 20 \mathrm{~mA}$, <br> $4 \mathrm{~mA} \ldots 20 \mathrm{~mA} \mathrm{Pt100/1000}, \mathrm{Ni1000} \mathrm{(each} \mathrm{input}$ <br> can be configured individually) |
| Channel input resistance | Voltage: > $100 \mathrm{k} \Omega$ <br> Current: ca. $330 \Omega$ |
| Time constant of the input filter | Voltage: $100 \mu \mathrm{~s}$ <br> Current: $100 \mu \mathrm{~s}$ |
| Indication of the input signals | 1 LED per channel (brightness depends on the <br> value of the analog signal) |
| Conversion cycle | 1 ms (for 4 inputs +2 outputs); with RTDs Pt/ <br> Ni... 1 s |
| Resolution | Range $0 \mathrm{~V} \ldots 10 \mathrm{~V}: 12$ bits <br> Range $-10 \mathrm{~V} \ldots+10 \mathrm{~V}: 12$ bits including sign <br> Range $0 \mathrm{~mA} \ldots 20 \mathrm{~mA}: 12$ bits <br> Range $4 \mathrm{~mA} \ldots 20 \mathrm{~mA}: 12$ bits <br> Range RTD (Pt100, PT1000, Ni1000): +0.1 ${ }^{\circ} \mathrm{C}$ |
| Conversion error of the analog values <br> caused by non-linearity, adjustment error <br> at factory and resolution within the normal <br> range | Typ. $0.5 \%$, max. $1 \%$ |
| Relationship between input signal and hex <br> code | Are configured as "unused" (default value) <br> Unused inputs <br> Overvoltage protection |

## Technical data of the analog inputs, if used as digital inputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | Max. 4 |
| Distribution of channels into groups | 1 group of 4 channels |
| Connections of the channels $\mathrm{AlO+} \ldots \mathrm{Al} 3+$ | Terminals $1.0 \ldots 1.3$ |
| Reference potential for the inputs | Terminals $1.9,2.9$ and $3.9(\mathrm{ZP})$ |
| Indication of the input signals | 1 LED per channel |
| Input signal voltage | 24 V DC |
|  | Signal 0 |
|  | Undefined signal |
|  | Signal 1 |
| Input current per channel | $+5 \mathrm{~V} \ldots+13 \mathrm{~V}$ |
|  | Input voltage +24 V |
|  | Input voltage +5 V |
|  | Input voltage +15 V |
|  | Input voltage +30 V |
| Input resistance | Typ. 7 mA |

Technical data of the analog outputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 2 |
| Distribution of channels into groups | 1 group for 2 channels |
| Connection of the channels $\mathrm{AO} 0+\ldots \mathrm{AO}++$ | Terminals 1.5 ... 1.6 |
| Reference potential for $\mathrm{AO} 0+\ldots \mathrm{AO}+$ | Terminal 1.7 (AO-) for voltage output terminal 1.9, 2.9 and 3.9 for current output |
| Output type |  |
| Unipolar | Current |
| Bipolar | Voltage |
| Galvanic isolation | Against internal supply and other modules |
| Configurability | -10 V ... +10 V, 0 mA ... $20 \mathrm{~mA}, 4 \mathrm{~mA} . . .20 \mathrm{~mA}$ (each output can be configured individually) |
| Output resistance (load), as current output | $0 \Omega \ldots 500 \Omega$ |
| Output loadability, as voltage output | $\pm 10 \mathrm{~mA}$ max. |
| Indication of the output signals | 1 LED per channel (brightness depends on the value of the analog signal) |
| Resolution | 12 bits including sign |
| Settling time for full range change (resistive load, output signal within specified tolerance) | Typ. 5 ms |
| Conversion error of the analog values caused by non-linearity, adjustment error at factory and resolution within the normal range | Typ. 0.5 \%, max. 1 \% |
| Relationship between input signal and hex code | Table Output ranges voltage and current $\Leftrightarrow$ Chapter 5.2.6.7.2.11.3 "Output ranges voltage and current" on page 1236 |
| Unused outputs | Are configured as "unused" (default value) and can be left open-circuited |

## Technical data of the fast counter

| Parameter | Value |
| :--- | :--- |
| Used inputs | Terminal 2.0 (DIO), 2.1 (DI1) |
| Used outputs | Terminal 3.0 (DO0) |
| Counting frequency | Depending on operation mode: |
|  | Mode 1-6: max. 200 kHz |
|  | Mode 7: max. 50 kHz |
|  | Mode 9: max. 35 kHz |
|  | Mode 10: max. 20 kHz |

[^25]
## Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm

The dimensions are in mm and in brackets in inch.

Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 220600 R0001 | Cl501-PNIO (V3), PROFINET commu- <br> nication interface module, 8 DI, 8 DO, <br> 4 AI and 2 AO | Active |
| 1SAP 420 600 R0001 | Cl501-PNIO-XC (V3), PROFINET <br> communication interface module, 8 DI, <br> 8 DO, 4 AI and 2 AO, XC version | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.6.7.3 CI502-PNIO

Features

- 8 digital inputs 24 V DC
- 8 digital outputs $24 \mathrm{~V} \mathrm{DC}, 0.5 \mathrm{~A}$ max.
- 8 configurable digital inputs/outputs $24 \mathrm{~V} D C, 0.5 \mathrm{~A}$ max.
- Module-wise galvanically isolated
- Fast counter
- XC version for usage in extreme ambient conditions available


1 I/O bus
2 Allocation between terminal number and signal name
38 yellow LEDs to display the signal states of the digital configurable inputs/outputs (DC0 ... DC7)
48 yellow LEDs to display the signal states of the digital inputs (DI8 ... DI15)
58 yellow LEDs to display the signal states of the digital outputs (DO8 ... DO15)
62 green LEDs to display the process supply voltage UP and UP3
73 red LEDs to display errors (CH-ERR1, CH-ERR2, CH-ERR3)
85 system LEDs: PWR/RUN, STA1 ETH, STA2 ETH, S-ERR, I/O-Bus
9 Label
102 rotary switches for setting the I/O device identifier
11 Ethernet interfaces (ETH1, ETH2) on the terminal unit
12 Terminal unit
13 DIN rail


## Intended purpose

The PROFINET communication interface module $\mathrm{Cl} 502-\mathrm{PNIO}$ is used as communication interface module in PROFINET networks. The network connection is performed via 2 RJ45 connectors which are integrated in the terminal unit.

For usage in extreme ambient conditions (e.g. wider temperature and humidity range), a special XC version of the device is available.

## Functionality

The CI502 communication interface module contains 24 I/O channels with the following properties:

- 8 digital configurable inputs/outputs
- 8 digital inputs: 24 V DC
- 8 digital outputs: 24 V DC, 0.5 A max.

The inputs/outputs are galvanically isolated from the Ethernet network. There is no potential separation between the channels. The configuration of the analog inputs/outputs is performed by software.

| Parameter | Value |
| :---: | :---: |
| Interface | Ethernet |
| Protocol | PROFINET IO RT |
| Power supply | From the process supply voltage UP |
| Supply of the electronic circuitry of the I/O expansion modules attached | Through the I/O bus interface (I/O bus) |
| Rotary switches | For setting the IO device identifier for configuration purposes (00h to FFh) |
| Configurable digital inputs/outputs | 8 (configurable via software) |
| Digital inputs | 8 (24 V DC; delay time configurable via software) |
| Digital outputs | 8 (24 V DC, 0.5 A max.) |
| LED displays | For system displays, signal states, errors and power supply |
| External supply voltage | Via terminals ZP, UP and UP3 (process supply voltage 24 V DC) |
| Effect of incorrect input terminal connection | Wrong or no signal detected, no damage up to 35 V |
| Required terminal unit | TU507-ETH or TU508-ETH « Chapter 5.2.7.2 "TU507-ETH and TU508-ETH for Ethernet communication interface modules" on page 1270 |

## Connections

## General

The Ethernet communication interface module $\mathrm{Cl} 502-\mathrm{PNIO}$ is plugged on the I/O terminal unit TU507-ETH ${ }^{\mu}$ Chapter 5.2.7.2 "TU507-ETH and TU508-ETH for Ethernet communication interface modules" on page 1270 or TU508-ETH $\Leftrightarrow$ Chapter 5.2.7.2 "TU507-ETH and TU508ETH for Ethernet communication interface modules" on page 1270. Properly seat the module and press until it locks in place. The terminal unit is mounted on a DIN rail or with 2 screws plus the additional accessory for wall mounting $\Leftrightarrow$ Chapter 5.2.8.2.5 "TA526-Wall mounting accessory" on page 1324.

The connection of the I/O channels is carried out using the 30 terminals of the I/O terminal unit. I/O modules can be replaced without re-wiring the terminal units.

For a detailed description of the mounting, disassembly and connection of the module, please refer to the installation instructions.

The terminals 1.8 and 2.8 as well as $1.9,2.9$ and 3.9 are electrically interconnected within the terminal unit and have always the same assignment, independent of the inserted module:

Terminals 1.8 and 2.8: Process supply voltage UP $=+24 \mathrm{~V}$ DC
Terminal 3.8: Process supply voltage UP3 $=+24 \mathrm{~V}$ DC
Terminals 1.9, 2.9 and 3.9: Process supply voltage $\mathrm{ZP}=0 \mathrm{~V}$.
The assignment of the other terminals:

With a separate UP3 power supply, the digital outputs can be switched off externally. This way, an emergency-off functionality can be realized.

Do not connect any voltages externally to digital outputs!
This ist not intended usage.
Reason: Externally voltages at one or more terminals $D C 0 \ldots D C 7$ or DO0 ... DO7 may cause that other digital outputs are supplied through that voltage instead of voltage UP3 (reverse voltage).
This is also possible, if DC channels are used as inputs. For this, the source for the input signals should be the impressed UP3 of the device.

This limitation does not apply for the input channels DIO ... DI7.

## CAUTION!

## Risk of malfunction by unintended usage!

If the function cut-off of the digital outputs is to be used by deactivation of the supply voltage UP3, be sure that no external voltage is conncted at the outputs DO0 ... DO7 and DC0 ... DC7.

Table 284: Assignment of the other terminals

| Terminal | Signal | Description |
| :---: | :---: | :---: |
| 1.0 | DC0 | Signal of the configurable digital input/output DCO |
| 1.1 | DC1 | Signal of the configurable digital input/output DC1 |
| 1.2 | DC2 | Signal of the configurable digital input/output DC2 |
| 1.3 | DC3 | Signal of the configurable digital input/output DC3 |
| 1.4 | DC4 | Signal of the configurable digital input/output DC4 |
| 1.5 | DC5 | Signal of the configurable digital input/output DC5 |
| 1.6 | DC6 | Signal of the configurable digital input/output DC6 |
| 1.7 | DC7 | Signal of the configurable digital input/output DC7 |
| 1.8 | UP | Process voltage UP (24 V DC) |
| 1.9 | ZP | Process voltage ZP (0 V DC) |
| 2.0 | DI8 | Signal of the digital input DI8 |
| 2.1 | DI9 | Signal of the digital input DI9 |
| 2.2 | DI10 | Signal of the digital input DI10 |
| 2.3 | DI11 | Signal of the digital input DI11 |
| 2.4 | DI12 | Signal of the digital input DI12 |
| 2.5 | DI13 | Signal of the digital input DI13 |
| 2.6 | DI14 | Signal of the digital input DI14 |
| 2.7 | DI15 | Signal of the digital input DI15 |
| 2.8 | UP | Process voltage UP (24 V DC) |
| 2.9 | ZP | Process voltage ZP (0 V DC) |
| 3.0 | DO8 | Signal of the digital output DO8 |
| 3.1 | DO9 | Signal of the digital output DO9 |
| 3.2 | DO10 | Signal of the digital output DO10 |
| 3.3 | D011 | Signal of the digital output DO11 |
| 3.4 | DO12 | Signal of the digital output DO12 |
| 3.5 | DO13 | Signal of the digital output DO13 |
| 3.6 | DO14 | Signal of the digital output DO14 |
| 3.7 | DO15 | Signal of the digital output DO15 |
| 3.8 | UP3 | Process voltage UP3 (24 V DC) |
| 3.9 | ZP | Process voltage ZP (0 V DC) |

WARNING!

## Removal/Insertion under power

The devices are not designed for removal or insertion under power. Because of unforeseeable consequences, it is not allowed to plug or unplug devices with the power being ON.

Make sure that all voltage sources (supply and process voltage) are switched off before you

- connect or disconnect any signal or terminal block
- remove, mount or replace a module.

Disconnecting any powered devices while energized in a hazardous location could result in an electric arc, which could create a flammable ignition resulting in fire or explosion.
Make sure that power is removed and that the area has been thoroughly checked to ensure that flammable materials are not present prior to proceeding.
The devices must not be opened when in operation. The same applies to the network interfaces.

## NOTICE!

Risk of damaging the PLC modules!
Overvoltages and short circuits might damage the PLC modules.

- Make sure that all voltage sources (supply voltage and process supply voltage) are switched off before you begin with operations on the system.
- Never connect any voltages or signals to reserved terminals (marked with ---). Reserved terminals may carry internal voltages.


Fig. 255: Connection of the Ethernet communication interface module Cl502-PNIO
ⓨ Chapter 6.8.4.2 "PROFINET communication interface module" on page 4596

## Connection of the Digital inputs



Fig. 256: Connection of the digital inputs (DI8 ... DI15)
The meaning of the LEDs is described in 'Displays' « Chapter 5.2.6.7.3.10 "State LEDs" on page 1261.

## Connection of the Digital outputs



Fig. 257: Connection of the digital outputs (DO8 ... DO15)
The meaning of the LEDs is described in 'Displays' « Chapter 5.2.6.7.3.10 "State LEDs" on page 1261.

## Connection of the configurable digital inputs/outputs

The following figure shows the connection of the configurable digital input/output DC0 and DC1. DC0 is connected as an input and DC1 is connected as an output. Proceed with the configurable digital inputs/outputs DC2 ... DC7 in the same way.


## CAUTION!

If a DC channel is used as input, the source for the input signals should be the impressed UP3 of the device $\Leftrightarrow$ Chapter 5.2.6.7.3.4 "Connections" on page 1246.


Fig. 258: Connection of the configurable digital inputs/outputs (DC0 ... DC7)(DC0 as input, DC1 as output)
The meaning of the LEDs is described in 'Displays' ${ }^{\mu}$ 解 Chapter 5.2.6.7.3.10 "State LEDs" on page 1261.

## Assignment of the Ethernet ports

The terminal unit for the communication interface module provides two Ethernet interfaces with the following pin assignment:

Pin assignment

| Interface | Pin | Signal | Description |
| :---: | :---: | :---: | :---: |
|  | 1 | TxD+ | Transmit data + |
|  | 2 | TxD- | Transmit data - |
|  | 3 | RxD+ | Receive data + |
|  | 4 | NC | Not connected |
|  | 5 | NC | Not connected |
|  | 6 | RxD- | Receive data - |
|  | 7 | NC | Not connected |
|  | 8 | NC | Not connected |
|  | Shield | Cable shield | Functional earth |

In corrosive environment, please protect unused connectors using the TA535 accessory.
Not supplied with this device.

* Further information about wiring and cable types


## Internal data exchange

| Parameter | Value |
| :--- | :--- |
| Digital inputs (bytes) | 5 |
| Digital outputs (bytes) | 5 |
| Counter input data (words) | 4 |
| Counter output data (words) | 8 |

## Addressing

The module has 2 rotary switches to set an explicit name to the PROFINET IO device before commissioning. No engineering tool is needed in this case.
The device gets its name (including the fixed part of the device name) with the switch settings ( $01 \mathrm{~h} . . \mathrm{FFh}$ ). This name can be used directly within the device configuration: "Cl5xx-pn-yy"
"CI5xx-pn-yy" xx is the fixed part of the device name (e.g. CI501) and yy represents the position of the rotary switch (0..FFh). The rotary switch values must be entered in hexadecimal format. For example, to set the name to "CI5xx-pn-08", set the upper rotary switch to " 0 " and the lower switch to " 8 ".

The module reads the position of the rotary switches only during power-up, i.e. changes of the switch position during operation will have no effect until the next module initialization.

## I/O configuration

The CI502-PNIO stores some PROFINET configuration parameters (I/O device identifier, I/O device type and IP address configuration). No more configuration data is stored.
The digital I/O channels are configured via software.
Details about configuration are described in 'Parameterization' $\Leftrightarrow$ Chapter 5.2.6.7.3.8 "Parameterization" on page 1253.

## Parameterization

## Parameters of the module

| Name | Value | Internal value | Internal value, type | Default |
| :---: | :---: | :---: | :---: | :---: |
| Module ID ${ }^{1}$ ) | Internal | 7005 | WORD | 7005 |
| Parameter length | Internal | 8 | BYTE | 8 |
| Error LED / Failsafe function (Table Error LED / Failsafe function ${ }^{4}$ Further information on page 1253) | On | 0 | BYTE | 0 |
|  | Off by E4 | 1 |  |  |
|  | Off by E3 | 3 |  |  |
|  | On + failsafe | 16 |  |  |
|  | Off by E4 + failsafe | 17 |  |  |
|  | Off by E3 + failsafe | 19 |  |  |
| Process cycle time | 1 ms process cycle time | 1 | BYTE | 1 ms |
|  | 2 ms process cycle time | 2 |  |  |
|  | 3 ms process cycle time | 3 |  |  |
|  | 4 ms process cycle time | 4 |  |  |
|  | 5 ms process cycle time | 5 |  |  |
|  | 6 ms process cycle time | 6 |  |  |
|  | 7 ms process cycle time | 7 |  |  |
|  | 8 ms process cycle time | 8 |  |  |
|  | 9 ms process cycle time | 9 |  |  |
|  | 10 ms process cycle time | 10 |  |  |
|  | 11 ms process cycle time | 11 |  |  |
|  | 12 ms process cycle time | 12 |  |  |
|  | 13 ms process cycle time | 13 |  |  |
|  | 14 ms process cycle time | 14 |  |  |
|  | 15 ms process cycle time | 15 |  |  |
|  | 16 ms process cycle time | 16 |  |  |
| Check supply | $\begin{aligned} & \text { Off } \\ & \text { On } \end{aligned}$ | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | BYTE | 1 |


| Name | Value | Internal value | Internal value, <br> type | Default |
| :--- | :--- | :--- | :--- | :--- |
| Fast counter | 0 | a <br> $:$ <br> $\left.10^{2}\right)$ | 10 | BYTE |
| I/O-Bus reset | Off | 0 | 0 |  |
|  | On | 1 | BYTE | Off |

${ }^{1}$ ) With a faulty ID, the module reports a "parameter error" and does not perform cyclic process data transmission.
${ }^{2}$ ) Counter operating modes $\stackrel{\text { \& Chapter 5.2.4.3.2.9 "Fast counter" on page } 606}{ }$

Table 285: Table Error LED / Failsafe function

| Setting | Description |
| :--- | :--- |
| On | Error LED (S-ERR) lights up at errors of all <br> error classes, Failsafe-mode off |
| Off by E4 | Error LED (S-ERR) lights up at errors of error <br> classes E1, E2 and E3, Failsafe-mode off |
| Off by E3 | Error LED (S-ERR) lights up at errors of error <br> classes E1 and E2, Failsafe-mode off |
| On + Failsafe | Error LED (S-ERR) lights up at errors of all <br> error classes, Failsafe-mode on *) |
| Off by E4 + Failsafe | Error LED (S-ERR) lights up at errors of error <br> classes E1, E2 and E3, Failsafe-mode on *) |
| Off by E3 + Failsafe | Error LED (S-ERR) lights up at errors of error <br> classes E1 and E2, Failsafe-mode on *) |
| *) The parameter Behaviour DO at comm. error is only analyzed if the Failsafe-mode is ON. |  |

IO-BUS reset after PROFINET reconnection

IO-BUS reset after PROFINET reconnection controls the behavior of PROFINET Cl modules in relation to connected I/O modules (both safety and non-safety I/O modules).

- IO-BUS reset after PROFINET reconnection = "On" resets and, thus, re-parameterizes all attached I/O modules. All internal I/O modules states are reset, including the related diagnosis information.
Note that if the parameter is set to "On" then:
- The bumpless re-start of non-safety I/O modules will not be supported. It means, for example, that non-safety output channels will go from fail-safe values to " 0 " values during the re-connection and re-parameterization time and after that go to new output values.
- Safety I/O modules will be re-parameterized and re-started as newly started modules, which may not require their PROFIsafe reintegration, depending on safety CPU state, in the safety application.
- IO-BUS reset after PROFINET reconnection = "Off" will not reset all attached I/O modules. It will re-parameterize I/O modules only if parameter change is detected during the reconnection. All internal I/O modules states are not reset, including the related diagnosis information.
Note that if the parameter is set to "Off" then:
- The bumpless re-start of non-safety I/O modules is supported (if no parameters are changed). It means, for example, that non-safety output channels will not go from failsafe values to " 0 " values during the re-connection and re-parameterization time, but directly from fail-safe values to new output values.
- Safety I/O modules will not be re-parameterized (if no parameters are changed). Thus, they may continue their operation, which may require their PROFIsafe reintegration in the safety application on the safety CPU, e.g., if PROFIsafe watchdog time for this safety I/O module has expired. Any reintegration of such safety I/O modules will be not only application specific but also PROFIsafe specific and depend on the safety I/O handling in the safety application.


## Group parameters for the digital part

| Name | Value | Internal value | Internal value, <br> type | Default |
| :--- | :--- | :--- | :--- | :--- |
| Input delay | 0.1 ms <br> 1 ms <br> 8 ms <br> 32 ms | 0 | 1 | BYTE |
| Detect short cir- <br> cuit at outputs | Off <br> On | 2 | 0.1 ms |  |
| Behaviour DO at <br> comm. error ${ }^{1}$ ) | Off <br> Last value <br> Last value 5 sec <br> Last value 10 sec <br> Substitute value <br> Substitute value <br> 5 sec <br> Substitute value <br> 10 sec | 2 | 7 | 11 |


| Name | Value | Internal value | Internal value, type | Default |
| :---: | :---: | :---: | :---: | :---: |
| Preventive voltage feedback monitoring for DC0..DC7 2 ${ }^{2}$ ) | $\begin{aligned} & \text { Off } \\ & \text { On } \end{aligned}$ | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | BYTE | $\begin{aligned} & \mathrm{Off} \\ & 0 \times 00 \end{aligned}$ |
| Detect voltage overflow at outputs ${ }^{3}$ ) | $\begin{aligned} & \text { Off } \\ & \text { On } \end{aligned}$ | $\begin{aligned} & 0 \\ & 1 \end{aligned}$ | BYTE | $\begin{aligned} & \text { Off } \\ & 0 \times 00 \end{aligned}$ |

Remarks:

| ${ }^{1}$ ) | The parameter Behaviour DO at comm. error is apply to DC and DO channels <br> and only analyzed if the Failsafe-mode is ON. |
| :--- | :--- |
| ${ }^{2}$ ) | The state "externally voltage detected" appears, if the output of a channel <br> DCO ... DC7 should be switched on while an externally voltage is connected. In <br> this case the start up is disabled, as long as the externally voltage is connected. <br> The monitoring of this state and the resulting diagnosis message can be disabled <br> by setting the parameters to "OFF". |
| ${ }^{3}$ ) | The error state "voltage overflow at outputs" appears, if externally voltage at dig- <br> ital outputs DC0 ... DC7 and accordingly DOO ... DO7 has exceeded the process <br> supply voltage UP3 $\#$ Chapter 5.2.6.7.3.4 "Connections" on page 1246 (see <br> description in section). The according diagnosis message "Voltage overflow on <br> outputs " can be disabed by setting the parameters on "OFF". This parameter <br> should only be disabled in exceptional cases for voltage overflow may produce <br> reverse voltage. |

## Diagnosis

Structure of the Diagnosis Block via function block PNIO_DEV_ALARM.

| Byte Number | Description | Possible Values |
| :---: | :---: | :---: |
| 1 | Diagnosis Byte, slot number | ```31 = CI502-PNIO (e. g. error at integrated 8 DI / 8 DO) 1 = 1st connected S500 I/O module 10 = 10th connected S500 I/O module``` |
| 2 | Diagnosis Byte, module number | According to the I/O bus specification passed on by modules to the fieldbus master |
| 3 | Diagnosis Byte, channel | According to the I/O bus specification passed on by modules to the fieldbus master |


| Byte Number | Description | Possible Values |
| :--- | :--- | :--- |
| 4 | Diagnosis Byte, error code | According to the I/O bus specification |
|  |  | Bit 7 and bit 6, coded error class |
|  |  | $0=\mathrm{E} 1$ |
|  |  | $1=\mathrm{E} 2$ |
|  |  | $2=\mathrm{E} 3$ |
|  | $3=\mathrm{E} 4$ |  |
|  |  | Bit 0 to bit 5, coded error description |
| 5 | Diagnosis Byte, flags | According to the I/O bus specification |
|  |  | Bit $7: 1$ = coming error |
|  |  | Bit 6: $1=$ leaving error |

In cases of short circuit or overload, the digital outputs are turned off. The modules performs reactivation automatically. Thus an acknowledgement of the errors is not necessary. The error message is stored via the LED.

| E1 ... E4 | d1 | d2 | d3 | d4 | Identifier 000 ... | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err |  |  |
| $\begin{aligned} & \hline \text { Byte } 4 \\ & \text { Bit } \\ & 6 \text {... } 7 \end{aligned}$ | - | Byte 1 | Byte 2 | Byte 3 | Byte 4 <br> Bit <br> 0 ... 5 |  |  |
| Class | Interface | Device | Module | Channel | Error- <br> Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) |  |  |  |  |
| Module errors |  |  |  |  |  |  |  |
| 3 | - | 31 | 31 | 31 | 19 | Checksum error in the I/O module | Replace I/O module |
| 3 | - | 31 | 31 | 31 | 3 | Timeout in the I/O module |  |
| 3 | - | 31 | 31 | 31 | 40 | Different hard-/firmware versions in the module |  |
| 3 | - | 31 | 31 | 31 | 43 | Internal error in the module |  |
| 3 | - | 31 | 31 | 31 | 36 | Internal data exchange failure |  |
| 3 | - | 31 | 31 | 31 | 9 | Overflow diagnosis buffer | Restart |
| 3 | - | 31 | 31 | 31 | 26 | Parameter error | Check master |


| E1 ... E4 | d1 | d2 | d3 | d4 | Identifier 000 ... | AC500Display | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err | $\begin{array}{\|l\|} \hline \text { PS501 } \\ \text { PLC } \\ \text { Browser } \end{array}$ |  |  |
|  | - | Byte 1 | Byte 2 | Byte 3 | $\begin{aligned} & \hline \text { Byte } 4 \\ & \text { Bit } \\ & 0 \text {... } 5 \end{aligned}$ | PNIO diagnosis block |  |  |
| Class | Interface | Device | Module | Channel | Error-Identifier | Error message |  | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) |  |  |  |  |  |
| 3 | - | 31 | 31 | 31 | 11 | Process voltage UP too low |  | Check process supply voltage |
| 3 | - | 31 | 31 | 31 | 45 | Process voltage UP gone |  | Check process supply voltage |
| 3 | - | 31/1 ... 10 | 31 | 31 | 17 | No communication with I/O device |  | Replace I/O module |
| 3 | - | 1 ... 10 | 31 | 31 | 32 | Wrong I/O device type on socket |  | Replace I/O module / Check configuration |
| 4 | - | 1 ... 10 | 31 | 31 | 31 | At least one module does not support failsafe function |  | Check modules and parameterization |
| 4 | - | 1 ... 10 | 31 | 5 | 8 | I/O module removed from hot swap terminal unit or defective module on hot swap terminal unit ${ }^{9}$ ) |  | Plug I/O module, replace I/O module |
| 4 | - | 1 ... 10 | 31 | 5 | 28 | Wrong I/O module plugged on hot swap terminal unit ${ }^{9}$ ) |  | Remove <br> wrong <br> I/O <br> module <br> and plug <br> pro- <br> jected <br> I/O <br> module |
| 4 | - | 1 ... 10 | 31 | 5 | 42 | No communication with I/O module on hot swap terminal unit ${ }^{9}$ ) |  | Replace I/O module |


| E1 ... E4 | d1 | d2 | d3 | d4 | Identifier 000 ... | AC500Display | <- Display in |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err | $\begin{aligned} & \hline \text { PS501 } \\ & \text { PLC } \\ & \text { Browser } \end{aligned}$ |  |  |
| Byte 4 Bit <br> 6 ... 7 | - | Byte 1 | Byte 2 | Byte 3 | Byte 4 <br> Bit <br> 0 ... 5 | PNIO diagnosis block |  |  |
| Class | $\begin{aligned} & \text { Inter- } \\ & \text { face } \end{aligned}$ | Device | Module | Channel | Error- <br> Identifier | Error message |  | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | $\left.{ }^{3}\right)$ |  |  |  |  |  |
| 4 | - | 1 ... 10 | 31 | 5 | 54 | I/O module does not support hot swap $\left.{ }^{8}\right)^{9}$ ) |  |  <br> Power <br> off <br> system <br> and <br> replace <br> I/O <br> module |
| 4 | - | 1 ... 10 | 31 | 6 | 8 | Hot swap terminal unit configured but not found |  | Replace terminal unit by hot swap terminal unit |
| 4 | - | 1 ... 10 | 31 | 6 | 42 | No communication with hot swap terminal unit ${ }^{9}$ ) |  | Restart, if error persists replace terminal unit |
| 4 | 1...6 | 255 | 2 | 0 | 45 | The connected Communication Module has no connection to the network |  | Check cabeling |
| 4 | - | 31 | 31 | 31 | 45 | Process voltage UP3 too low |  | Check process voltage |
| 4 | - | 31 | 31 | 31 | 46 | Reverse voltage from digital outputs DO0..DO7 to UP3 ${ }^{4}$ ) |  | Check terminals |
| 4 | - | 31/1 ... 10 | 31 | 31 | 34 | No response during initialization of the I/O module |  | Replace <br> $1 / O$ <br> module |
| 4 | - | 31 | 31 | 31 | 11 | Process voltage UP3 too low |  | Check process supply voltage |
| 4 | - | 31 | 31 | 31 | 45 | Process voltage UP3 gone |  | Check process supply voltage |


| E1 ... E4 | d1 | d2 | d3 | d4 |  | $\|$AC500- <br> Display $<-$ Display | <- Display in |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Class | Comp | Dev | Mod | Ch | Err | $\begin{array}{\|l\|} \hline \text { PS501 } \\ \text { PLC } \\ \text { Browser } \end{array}$ |  |
| Byte 4 <br> Bit <br> 6 ... 7 | - | Byte 1 | Byte 2 | Byte 3 | Byte 4 <br> Bit <br> 0 ... 5 | PNIO diagnosis block |  |
| Class | Inter- face | Device | Module | Channel | Error-Identifier | Error message | Remedy |
|  | ${ }^{1}$ ) | ${ }^{2}$ ) | ${ }^{3}$ ) |  |  |  |  |
| 4 | - | 31 | 31 | 31 | 10 | Voltage overflow at outputs (above UP3 level) ${ }^{5}$ ) | Check terminals/ check process supply voltage |
| Channel error digital |  |  |  |  |  |  |  |
| 4 | - | 31 | 2 | 8 ... 15 | 46 | Externally voltage detected at digital output DO0 ... DO7 ${ }^{6}$ ) | Check terminals |
| 4 | - | 31 | 4 | $0 . . .7$ | 46 | Externally voltage detected at digital output DC0 ... DC7 ${ }^{6}$ ) | Check terminals |
| 4 | - | 31 | 4 | $0 . . .7$ | 47 | Short circuit at digital output DC0 ... DC7 ${ }^{7}$ ) | Check terminals |
| 4 | - | 31 | 2 | 8 ... 15 | 47 | Short circuit at digital output DOO ... DO77) | Check terminals |

Remarks:

| $\left.{ }^{1}\right)$ | In AC500 the following interface identifier applies: <br> "-" = Diagnosis via bus-specific function blocks; 0 ... 4 or 10 $=$ Position of the <br> Communication Module;14 = I/O-Bus; 31 = Module itself <br> The identifier is not contained in the CI502-PNIO diagnosis block. |
| :--- | :--- |
| $\left.{ }^{2}\right)$ | With "Device" the following allocation applies: $31=$ Module itself, $1 \ldots 10=$ <br> Expansion module |
| $\left.{ }^{3}\right)$ | With "Module" the following allocation applies dependent of the master: <br> Module error: $31=$ Module itself <br> Channel error: Module type $(1=\mathrm{AI}, 2=\mathrm{DO}, 3=\mathrm{AO})$ |


| ${ }^{4}$ ) | This message appears, if externally voltages at one or more terminals DC0 ... DC7 oder DO0 ... DO7 cause that other digital outputs are supplied through that voltage (voltage feedback, see description in 'Connections' \# Chapter 5.2.6.7.3.4 "Connections" on page 1246. All outputs of the apply digital output groups will be turned off for 5 seconds. The diagnosis message appears for the whole output group. |
| :---: | :---: |
| ${ }^{5}$ ) | The voltage at digital outputs DC0 ... DC7 and accordingly DO0 ... DO7 has exceeded the process supply voltage UP3 © Chapter 5.2.6.7.3.4 "Connections" on page 1246. Diagnosis message appears for the whole module. |
| ${ }^{6}$ ) | This message appears, if the output of a channel DC0 ... DC7 or DOO ... DO7 should be switched on while an externally voltage is connected. In this case the start up is disabled, as long as the externally voltage is connected. Otherwise this could produce reverse voltage from this output to other digital outputs. This diagnosis message appears per channel. |
| ${ }^{7}$ ) | Short circuit: After a detected short circuit, the output is deactivated for 2000 ms . Then a new start up will be executed. This diagnosis message appears per channel. |
| ${ }^{8}$ ) | In case of an I/O module doesn't support hot swapping, do not perform any hot swap operations (also not on any other terminal units (slots)) as modules may be damaged or I/O bus communication may be disturbed. |
| ${ }^{9}$ ) | Diagnosis for hot swap available as of version index F0. |

## State LEDs

The LEDs are located at the front of module. There are 2 different groups:

- The 5 system LEDs (PWR, STA1 ETH, STA2 ETH, S-ERR and I/O-Bus) show the operation state of the module and display possible errors.
- The 29 process LEDs (UP, UP3, inputs, outputs, CH-ERR1 to CH-ERR3) show the process supply voltage and the states of the inputs and outputs and display possible errors.

Table 286: States of the 5 system LEDs

| LED | Color | OFF | ON | Flashing |
| :---: | :---: | :---: | :---: | :---: |
| PWR/RUN | Green | Process supply voltage missing | Internal supply voltage OK, module ready for communication with IO Controller | Start-up / preparing communication |
|  | Yellow | --- | --- | --- |
| STA1 ETH <br> (System-LED "BF") | Green | --- | Device configured, cyclic data exchange running | --- |
|  | Red | --- | --- | Device is not configured |
| STA2 ETH <br> (System LED "SF") | Green | --- | --- | Got identification request from I/O controller |
|  | Red | No system error | System error (collective error) | --- |
| S-ERR | Red | No error | Internal error | -- |


| LED | Color | OFF | ON | Flashing |
| :--- | :--- | :--- | :--- | :--- |
| I/O-Bus | Green | No expansion <br> modules con- <br> nected or com- <br> munication error | Expansion <br> modules con- <br> nected and <br> operational | --- |
| ETH1 | Green | No connection at <br> Ethernet inter- <br> face | Connected to <br> Ethernet inter- <br> face | --- |
|  | Yellow | --- | Device is trans- <br> mitting telegrams | Device is trans- <br> mitting telegrams |
|  | Green | No connection at <br> Ethernet inter- <br> face | Connected to <br> Ethernet inter- <br> face | --- |
|  | Yellow | --- | Device is trans- <br> mitting telegrams | Device is trans- <br> mitting telegrams |

Table 287: States of the 29 process LEDs

| LED | Color | OFF | ON | Flashing |
| :--- | :--- | :--- | :--- | :--- |
| DC0 ... DC7 | Yellow | Input/Output is <br> OFF | Input/Output is <br> ON | -- |
| DI8 ... DI15 | Yellow | Input is OFF | Input is ON (the <br> input voltage is <br> even displayed if <br> the supply <br> voltage is OFF) | -- <br> DO8 ... DO15 <br> UP Yellow |
| Green | Output is OFF <br> voltage missing | Output is ON | Process supply <br> voltage OK and <br> initialization fin- <br> ished | -- |
| UP3 | Green | Process supply <br> voltage missing | Process supply <br> voltage OK | -- |
| CH-ERR1 to CH- <br> ERR3 | Red | No error or <br> process supply <br> voltage missing | Internal error | Error on one <br> channel of the <br> corresponding <br> group |

## Technical data

## Technical data of the module

The system data of AC500 and S500 are applicable to the standard version $\Leftrightarrow$ Chapter 5.1.2 "System data AC500" on page 166.

The system data of AC500-XC are applicable to the XC version $\Leftrightarrow$ Chapter 5.1.3 "System data AC500-XC" on page 169.
Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter | Value |
| :---: | :---: |
| Process supply voltages UP/UP3 |  |
| Rated value | 24 V DC (for inputs and outputs) |
| Max. load for the terminals | 10 A |
| Protection against reversed voltage | Yes |
| Rated protection fuse on UP/UP3 | 10 A fast |
| Galvanic isolation | Ethernet interface against the rest of the module |
| Inrush current from UP (at power up) | On request |
| Current consumption via UP (normal operation) | 0.15 A |
| Current consumption via UP3 | 0.06 A + 0.5 A max. per output |
| Connections | Terminals 1.8 and 2.8 for +24 V (UP) <br> Terminal 3.8 for +24 V (UP3) <br> Terminals 1.9, 2.9 and 3.9 for 0 V (ZP) |
| Max. power dissipation within the module | 6 W |
| Number of digital inputs | 8 |
| Number of digital outputs | 8 |
| Number of configurable digital inputs/outputs | 8 |
| Input data length | 12 bytes |
| Output data length | 20 bytes |
| Reference potential for all digital inputs and outputs | Negative terminal of the supply voltage, signal name ZP |
| Setting of the I/O device identifier | With 2 rotary switches at the front side of the module |
| Diagnosis | See Diagnosis and Displays $\stackrel{y}{c}$ Chapter 5.2.6.7.3.9 "Diagnosis" on page 1256 |
| Operation and error displays | 34 LEDs (totally) |
| Weight (without terminal unit) | Ca. 125 g |
| Mounting position | Horizontal or vertical with derating (output load reduced to $50 \%$ at $+40^{\circ} \mathrm{C}$ per group) |
| Extended ambient temperature (XC version) | > +60 ${ }^{\circ} \mathrm{C}$ on request |
| Cooling | The natural convection cooling must not be hindered by cable ducts or other parts in the control cabinet. |

## NOTICE!

All I/O channels (digital and analog) are protected against reverse polarity, reverse supply, short circuit and temporary overvoltage up to 30 V DC.

## Multiple overloads

No effects of multiple overloads on isolated multi-channel modules occur, as every channel is protected individually by an internal smart high-side switch.

| Parameter | Value |
| :---: | :---: |
| Bus connection | $2 \times \mathrm{RJ} 45$ |
| Switch | Integrated |
| Technology | Hilscher NETX 100 |
| Transfer rate | 10/100 Mbit/s (full-duplex) |
| Transfer method | According to Ethernet II, IEEE 802.3 |
| Ethernet | 100 base-TX, internal switch, $2 \times$ RJ45 socket |
| Expandability | Max. 10 S500 I/O modules |
| Adjusting elements | 2 rotary switches for generation of an explicit name |
| Supported protocols | RTC - real time cyclic protocol, class 1 *) <br> RTA - real time acyclic protocol <br> DCP - discovery and configuration protocol <br> CL-RPC - connectionless remote procedure Call <br> LLDP - link layer discovery protocol <br> MRP - MRP Client |
| Acyclic services | PNIO read / write sequence (max. 1024 bytes per telegram) <br> Process-Alarm service |
| Supported alarm types | Process Alarm, Diagnostic Alarm, Return of SubModule, Plug Alarm, Pull Alarm |
| Min. bus cycle | 1 ms |
| Conformance class | CC A |
| Protective functions (according to IEC 61131-3) | Protected against: <br> - short circuit <br> - reverse supply <br> - overvoltage <br> - reverse polarity <br> Galvanic isolation from the rest of the module |

[^26]
## Technical data of the digital inputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 8 |
| Distribution of the channels into groups | 1 group of 8 channels |
| Terminals of the channels DIO ... DI7 | Terminals $2.0 \ldots 2.7$ |
| Reference potential for all inputs | Terminals $1.9 \ldots 3.9$ (Negative terminal of the <br> supply voltage, signal name ZP) |
| Indication of the input signals | 1 yellow LED per channel, the LED is ON when <br> the input signal is high (signal 1$)$ |
| Input type (according EN 61131-2) | Type 1 |
| Input delay (0->1 or $1->0)$ | Typ. 0.1 ms, configurable from $0.1 \mathrm{~ms} \ldots 32 \mathrm{~ms}$ |


| Parameter |  | Value |
| :--- | :--- | :--- |
| Input signal voltage | 24 V DC |  |
|  | Signal 0 | $-3 \mathrm{~V} \ldots+5 \mathrm{~V}$ |
|  | Undefined Signal | $>+5 \mathrm{~V} \ldots<+15 \mathrm{~V}$ |
|  | Signal 1 | $+15 \mathrm{~V} \ldots+30 \mathrm{~V}$ |
| Ripple with signal 0 | Within $-3 \mathrm{~V} \ldots+5 \mathrm{~V}$ |  |
| Ripple with signal 1 | Within $+15 \mathrm{~V} \ldots+30 \mathrm{~V}$ |  |
| Input current per channel |  |  |
|  | Input voltage +24 V | Typ. 5 mA |
|  | Input voltage +5 V | $>1 \mathrm{~mA}$ |
|  | Input voltage +15 V | $>2 \mathrm{~mA}$ |
|  | Input voltage +30 V | $<8 \mathrm{~mA}$ |
| Max. cable length |  |  |
|  | Shielded | 1000 m |
|  | Unshielded | 600 m |

## Technical data of the digital outputs

| Parameter | Value |
| :---: | :---: |
| Number of channels per module | 8 |
| Distribution of the channels into groups | 1 group of 8 channels |
| Terminals of the channels DO0 ... DO7 | Terminals 3.0 ... 3.7 |
| Reference potential for all outputs | Terminals 1.9 ... 3.9 (negative terminal of the supply voltage, signal name ZP) |
| Common power supply voltage | For all outputs terminal 3.8 (positive terminal of the supply voltage, signal name UP3) |
| Output voltage for signal 1 | UP3 (-0.8 V) |
| Output delay (0->1 or 1->0) | On request |
| Output current |  |
| Rated value per channel | 500 mA at UP3 $=24 \mathrm{~V}$ |
| Max. value (all channels together) | 4 A |
| Leakage current with signal 0 | $<0.5 \mathrm{~mA}$ |
| Fuse for UP3 | 10 A fast |
| Demagnetization with inductive DC load | Via internal varistors (see figure below this table) |
| Output switching frequency |  |
| With resistive load | On request |
| With inductive loads | Max. 0.5 Hz |
| With lamp loads | 11 Hz max. at 5 W max. |
| Short-circuit-proof / overload-proof | Yes |
| Overload message ( l 0.7 A) | Yes, after ca. 100 ms |


| Parameter | Value |
| :---: | :---: |
| Output current limitation | Yes, automatic reactivation after short circuit/overload |
| Resistance to feedback against 24 V signals | Yes (software-controlled supervision) |
| Max. cable length |  |
| Shielded | 1000 m |
| Unshielded | 600 m |

The following drawing shows the circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off.


1 Digital output
2 Varistors for demagnetization when inductive loads are turned off

## Technical data of the configurable digital inputs/outputs

Each of the configurable I/O channels is defined as input or output by the user program. This is done by interrogating or allocating the corresponding channel.

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 8 inputs/outputs (with transistors) |
| Distribution of the channels into groups | 1 group for 8 channels |
| If the channels are used as inputs |  |
|  | Channels DC0 ... DC07 |
| If the channels are used as outputs |  |
| Channels DC0 ... DC07 |  |
| Indication of the input/output signals | Terminals 1.0 ... 1.7 |
| Galvanic isolation | 1 yellow LED per channel, the LED is ON when <br> the input/output signal is high (signal 1) |

## Technical data of the digital inputs/outputs if used as inputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 8 |
| Distribution of the channels into groups | 1 group of 8 channels |
| Terminals of the channels DC0 ... DC7 | Terminals $1.0 \ldots 1.7$ |
| Reference potential for all inputs | Terminals $1.9 \ldots 3.9$ (Negative terminal of the <br> supply voltage, signal name ZP) |


| Parameter | Value |
| :---: | :---: |
| Indication of the input signals | 1 yellow LED per channel, the LED is ON when the input signal is high (signal 1) |
| Input type (according EN 61131-2) | Type 1 |
| Input delay (0->1 or 1->0) | Typ. 0.1 ms , configurable from $0.1 \mathrm{~ms} . . .32 \mathrm{~ms}$ |
| Input signal voltage | 24 V DC |
| Signal 0 | -3 V ... +5 V |
| Undefined Signal | > +5 V ... < +15 V |
| Signal 1 | +15V ... +30 V |
| Ripple with signal 0 | Within -3V ... +5V |
| Ripple with signal 1 | Within +15 V ... +30 V |
| Input current per channel |  |
| Input voltage +24 V | Typ. 5 mA |
| Input voltage +5V | > 1 mA |
| Input voltage +15 V | $>2 \mathrm{~mA}$ |
| Input voltage +30 V | $<8 \mathrm{~mA}$ |
| Max. cable length |  |
| Shielded | 1000 m |
| Unshielded | 600 m |

*) Due to the direct connection to the output, the demagnetizing varistor is also effective at the input (see figure) above. This is why the difference between UPx and the input signal may not exceed the clamp voltage of the varistor. The varistor limits the voltage to approx. 36 V . Following this, the input voltage must range from $-12 \mathrm{~V} \ldots+30 \mathrm{~V}$ when UPx $=24 \mathrm{~V}$ and from $-6 \mathrm{~V} . . .+30 \mathrm{~V}$ when UPx $=30 \mathrm{~V}$.

## Technical data of the digital inputs/outputs if used as outputs

| Parameter | Value |
| :--- | :--- |
| Number of channels per module | 8 |
| Distribution of the channels into groups | 1 group of 8 channels |
| Terminals of the channels DC0 ... DC7 | Terminals $1.0 \ldots 1.7$ |
| Reference potential for all outputs | Terminals $1.9 \ldots 3.9$ (negative terminal of the <br> supply voltage, signal name ZP) |
| Common power supply voltage | For all outputs terminal 3.8 (positive terminal of <br> the supply voltage, signal name UP3) |
| Output voltage for signal 1 | UP3 (-0.8 V) |
| Output delay (0->1 or 1->0) | On request |
| Output current | 5 |
| Rated value per channel |  |
|  | Max. value (all channels together) |
| Leakage current with signal 0 | 4 A |
| Fuse for UP3 |  |
| Demagnetization with inductive DC load | < 0.5 mA |


| Parameter |  | Value |
| :--- | :--- | :--- |
| Output switching frequency |  |  |
|  | With resistive load | On request |
|  | With inductive loads | Max. 0.5 Hz |
|  | With lamp loads | 11 Hz max. at 5 W max. |
| Short-circuit-proof / overload proof | Yes |  |
| Overload message (I > 0.7 A) | Yes, after ca. 100 ms |  |
| Output current limitation | Yes, automatic reactivation after short cir- <br> cuit/overload |  |
| Resistance to feedback against 24 V signals | Yes (software-controlled supervision) |  |
| Max. cable length |  |  |
|  | Shielded | 1000 m |
|  | Unshielded | 600 m |

The following drawing shows the circuitry of a digital input/output with the varistors for demagnetization when inductive loads are switched off.


1 Digital input/output
2 For demagnetization when inductive loads are turned off

## Technical data of the fast counter

| Parameter | Value |
| :--- | :--- |
| Used inputs | Terminal 2.0 (DI8),Terminal 2.1 (DI9) |
| Used outputs | Terminal 3.0 (DO8) |
| Counting frequency | Depending on operation mode: |
|  | Mode 1- 6: max. 200 kHz |
|  | Mode 7: max. 50 kHz |
|  | Mode 9: max. 35 kHz |
|  | Mode 10: max. 20 kHz |

Chapter 6.8.2.12 "Fast counters in AC500 devices" on page 4536

## Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm

The dimensions are in mm and in brackets in inch.

## Ordering data

| Active | Active | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 220 700 R0001 | CI502-PNIO (V3), PROFINET commu- <br> nication interface module, 8 DI, 8 DO <br> and 8 DC | Active |
| 1SAP 420 700 R0001 | CI502-PNIO-XC (V3), PROFINET <br> communication interface module, 8 DI, <br> 8 DO and 8 DC, XC version | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.7 Terminal units for communication interface modules

### 5.2.7.1 Safety instructions

## Hot swap

System requirements for hot swapping of I/O modules:

- Types of terminal units that support hot swapping of I/O modules have the appendix TU5xx-H.
- I/O modules as of index FO.

The following I/O bus masters support hot swapping of attached I/O modules:

- Communication interface modules CI5xx as of index FO.
- Processor modules PM56xx-2ETH with firmware version as of V3.2.0.


## NOTICE!

## Risk of damage to I/O modules!

Hot swapping is only allowed for I/O modules.
Processor modules and communication interface modules must not be removed or inserted during operation.

## Conditions for hot swapping

- Digital outputs are not under load.
- Input/output voltages above safety extra low voltage/ protective extra low voltages (SELV/PELV) are switched off.
- Modules are completely plugged on the terminal unit with both snap fit engaged before switching on loads or input/output voltage.
$\star$ Further information about hot swap.


### 5.2.7.2 TU507-ETH and TU508-ETH for Ethernet communication interface modules

### 5.2.7.2.1 Features

- TU507-ETH, Ethernet terminal unit, 24 V DC, screw terminals
- TU508-ETH, Ethernet terminal unit, 24 V DC, spring terminals
- TU508-ETH-XC, Ethernet terminal unit, 24 V DC, spring terminals, XC version


1 I/O bus (10 pins, female) to connect the first terminal unit
2a Plug ( $2 \times 25$ pins) to connect the inserted Ethernet communication interface module
2b Plug ( $3 x 19$ pins) to connect the inserted Ethernet communication interface module
3 With a screwdriver, inserted in this place, the terminal unit and the adjacent terminal unit can be shoved from each other
42 holes for wall mounting
52 RJ45 interfaces with indication LEDs for connection with the Ethernet network
630 terminals for signals and process supply voltages (UP and UP3)
7 DIN rail

The Ethernet communication interface modules plug into the Ethernet terminal unit. When properly seated, they are secured with two mechanical locks. All the connections are made through the Ethernet terminal unit, which allows removal and replacement of the Ethernet communication interface modules without disturbing the wiring at the Ethernet terminal unit.
The Ethernet terminal units TU507-ETH and TU508-ETH are specifically designed for use with AC500/S500 Ethernet communication interface modules (e. g. CI501-PNIO).


## Extreme conditions

Terminal units for use in extreme ambient conditions have no $\stackrel{*+k}{*+k}$ sign for XC version.

The figure 4 in the Part no. 1SAP4... (label) identifies the XC version.

## Terminals

| Screw terminals |  |  | Spring terminals |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Conductor |  | Screwdriver | Conductor |  | Screwdriver (opens terminal) |
|  |  |  |  |  |  |

The terminals 1.8 and 2.8 as well as $1.9,2.9$ and 3.9 are electrically interconnected within the terminal unit and have always the same assignment, independent of the inserted module:

Terminals 1.8 and 2.8: Process supply voltage UP $=+24 \mathrm{~V}$ DC
Terminal 3.8: Process supply voltage UP3 $=+24 \mathrm{~V}$ DC
Terminals 1.9, 2.9 and 3.9: Process supply voltage $\mathrm{ZP}=0 \mathrm{~V}$

The assignment of the other terminals is dependent on the inserted communication interface module.

## NOTICE!

Risk of corrosion!
Unused connectors and slots may corrode if XC devices are used in salt-mist environments.

Protect unused connectors and slots with TA535 protective caps for XC devices Chapter 5.2.8.3.6 "TA535-Protective caps for XC devices" on page 1332.

### 5.2.7.2.2 Technical data

The system data of AC500 and S500 are applicable to the standard version ${ }^{*} \Rightarrow$ Chapter 5.1.2 "System data AC500" on page 166.

The system data of AC500-XC are applicable to the XC version ${ }^{〔}$ Chapter 5.1.3 "System data AC500-XC" on page 169.
Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter | Value |
| :--- | :--- |
| Number of I/O channels per module | Max. 24 (depending on the inserted communi- <br> cation interface module) |
| Distribution of the channels into groups | 3 groups of max. 8 channels each (1.0 ... 1.7, <br> $2.0 \ldots 2.7,3.0 \ldots 3.7)$, the allocation of the <br> channels is given by the inserted Ethernet bus <br> module |
| Network interface connector | 2 RJ45, 8-pole |
| Rated voltage | 24 V DC |
| Max. permitted total current | 10 A via the supply terminals (UP, UP3 and <br> ZP) |
| Ethernet | $10 / 100$ base-TX or 100 base-TX (depending <br> on CI5xx module plugged in), 2 RJ45 socket |
| Grounding | Direct connection to the grounded DIN rail or <br> via the screws with wall mounting |
| Screw terminals | Front terminal, conductor connection vertically <br> with respect to the printed circuit board |
| Spring-type terminals | Front terminal, conductor connection vertically <br> with respect to the printed circuit board |
| Weight | 200 g |
| Mounting position | Horizontal or vertical |

### 5.2.7.2.3 Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm

The dimensions are in mm and in brackets in inch.

### 5.2.7.2.4 Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 214 200 R0001 | TU507-ETH, Ethernet terminal unit, <br> 24 V DC, screw terminals | Active |
| 1SAP 214 000 R0001 | TU508-ETH, Ethernet terminal unit, <br> 24 V DC, spring terminals | Active |
| 1SAP 414 000 R0001 | TU508-ETH-XC, Ethernet terminal <br> unit, 24 V DC, spring terminals, <br> XC version | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.7.3 TU509 and TU510 for communication interface modules

### 5.2.7.3.1 Features

- TU509, terminal unit, 24 V DC, screw terminals
- TU510, terminal unit, 24 V DC, spring terminals
- TU510-XC, terminal unit, 24 V DC, spring terminals, XC version


1 I/O bus (10 pins, female) to connect the first terminal unit
2a Plug (2 25 pins) to connect the inserted communication interface module
2b Plug (3 19 pins) to connect the inserted communication interface module
3 With a screwdriver, inserted in this place, the terminal unit and the adjacent terminal unit can be shoved from each other
42 holes for wall mounting
5 D-sub 9 (female) for connection with the PROFIBUS network
630 terminals for signals and process supply voltages (UP and UP3)
7 DIN rail

The communication interface modules plug into the terminal unit. When properly plugged-in, they are secured with two mechanical locks. All the connections are established via the terminal unit, which allows removal and replacement of the communication interface modules without disturbing the wiring at the terminal unit.
The terminal units TU509 and TU510 are specifically designed for use with AC500/S500 communication interface modules (e. g. CI451-DP).


## Extreme conditions

Terminal units for use in extreme ambient conditions have no ${ }^{x_{+}+\ldots}$ sign for XC version.

The figure 4 in the Part no. 1SAP4... (label) identifies the XC version.

## Terminals

| Screw terminals |  |  | Spring terminals |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Conductor |  | Screwdriver | Conductor |  | Screwdriver (opens terminal) |
|  |  |  |  |  |  |

The terminals $2.8,3.8,2.9,3.9$ and 4.9 are electrically interconnected within the terminal unit and always have the same assignment, irrespective of the inserted module:

Terminals 2.8 and 3.8: process supply voltage UP $=+24 \mathrm{~V}$ DC
Terminal 4.8: process supply voltage UP3 $=+24 \mathrm{~V}$ DC
Terminals 2.9, 3.9 and 4.9: process supply voltage $\mathrm{ZP}=0 \mathrm{~V}$

The assignment of the other terminals depends on the inserted communication interface module (see communication interface modules for CANopen and PROFIBUS).

## NOTICE!

Risk of corrosion!
Unused connectors and slots may corrode if XC devices are used in salt-mist environments.

Protect unused connectors and slots with TA535 protective caps for XC devices Chapter 5.2.8.3.6 "TA535-Protective caps for XC devices" on page 1332.

### 5.2.7.3.2 Technical data

The system data of AC500 and S500 are applicable to the standard version ${ }^{\mu}$ Chapter 5.1.2 "System data AC500" on page 166.

The system data of AC500-XC are applicable to the XC version ${ }^{〔}$ Chapter 5.1.3 "System data AC500-XC" on page 169.
Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter | Value |
| :--- | :--- |
| Number of I/O channels per module | Max. 24 (depending on the inserted communi- <br> cation interface module) |
| Distribution of the channels into groups | 3 groups of max. 8 channels each (2.0 ... 2.7, <br> $3.0 \ldots 3.7,4.0 \ldots 4.7)$, the allocation of the <br> channels is given by the inserted communica- <br> tion interface module |
| Network interface connector | $9-$ pin D-sub connector, female |
| Rated voltage | 24 V DC |
| Max. permitted total current | 10 A via the supply terminals (UP, UP3 and <br> ZP $)$ |
| Grounding | Direct connection to the grounded DIN rail or <br> via the screws with wall mounting |
| Screw terminals | Front terminal, conductor connection vertically <br> with respect to the printed circuit board |
| Spring terminals | Front terminal, conductor connection vertically <br> with respect to the printed circuit board |
| Weight | 200 g |
| Mounting position | Horizontal or vertical |

### 5.2.7.3.3 Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm
$\square$

### 5.2.7.3.4 Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 211 000 R0001 | TU509, terminal unit, 24 V DC, screw <br> terminals | Active |
| 1SAP 210 800 R0001 | TU510, terminal unit, 24 V DC, spring <br> terminals | Active |
| 1SAP 410 800 R0001 | TU510-XC, terminal unit, 24 V DC, <br> spring terminals, XC version | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.7.4 TU517 and TU518 for communication interface modules

### 5.2.7.4.1 Features

- TU517, terminal unit, 24 V DC, screw terminals
- TU518, terminal unit, 24 V DC, spring terminals
- TU518-XC, terminal unit, 24 V DC, spring terminals, XC version


1 I/O bus (10 pins, female) to connect the first terminal unit
2a Plug (2 25 pins) to connect the inserted communication interface module
2b Plug ( 219 pins) to connect the inserted communication interface module
3 With a screwdriver, inserted in this place, the terminal unit and the adjacent I/O terminal unit can be shoved from each other
42 holes for wall mounting
510 terminals for connection with the bus system
630 terminals for signals and process supply voltages (UP and UP3)
7 DIN rail

The communication interface modules plug into the terminal unit. When properly plugged-in, they are secured with two mechanical locks. All the connections are established via the terminal unit, which allows removal and replacement of the communication interface modules without disturbing the wiring at the terminal unit.
The terminal units TU517 and TU518 are specifically designed for use with AC500/S500 communication interface modules (e. g. CI581-CN, CI541-DP):

- CANopen communication interface modules
- DeviceNet modules
- PROFIBUS DP communication interface modules



## Extreme conditions

Terminal units for use in extreme ambient conditions have no $\underset{x_{+}^{*}+{ }_{*}^{*}}{*}$ sign for XC version.

The figure 4 in the Part no. 1SAP4... (label) identifies the XC version.

## Terminals

| Screw terminals |  |  | Spring terminals |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Conductor |  | Screwdriver | Conductor |  | Screwdriver (opens terminal) |
|  |  |  |  |  |  |

The terminals $2.8,3.8,2.9,3.9$ and 4.9 are electrically interconnected within the terminal unit and always have the same assignment, irrespective of the inserted communication interface module:

- Terminals 2.8 and 3.8: process supply voltage UP $=+24 \mathrm{~V}$ DC
- Terminal 4.8: process supply voltage UP3 $=+24 \mathrm{~V}$ DC
- Terminals 2.9, 3.9 and 4.9: process supply voltage $\mathrm{ZP}=0 \mathrm{~V}$

The assignment of the other terminals depends on the inserted communication interface module (see communication interface modules for CANopen and PROFIBUS).

### 5.2.7.4.2 Technical data

The system data of AC500 and S500 are applicable to the standard version $\left.{ }^{*}\right\rangle$ Chapter 5.1.2 "System data AC500" on page 166.
The system data of AC500-XC are applicable to the XC version ${ }^{〔}$ Chapter 5.1.3 "System data AC500-XC" on page 169.

Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter | Value |
| :--- | :--- |
| Number of I/O channels per module | Max. 24 (depending on the inserted communi- <br> cation interface module) |
| Distribution of the channels into groups | 3 groups of max. 8 channels each (2.0 ... 2.7, <br> $3.0 \ldots 3.7,4.0 \ldots 4.7)$, the allocation of the <br> channels is given by the inserted communica- <br> tion interface module |
| Network interface connector | 10 screw or spring terminals (1.0 ... 1.9) |
| Rated voltage | 24 V DC |
| Max. permitted total current | 10 A via the supply terminals (UP, UP3 and <br> ZP) |
| Grounding | Direct connection to the grounded DIN rail or <br> via the screws with wall mounting |
| Screw terminals | Front terminal, conductor connection vertically <br> with respect to the printed circuit board |
| Spring terminals | Front terminal, conductor connection vertically <br> with respect to the printed circuit board |
| Weight | 200 g |
| Mounting position | Horizontal or vertical |

### 5.2.7.4.3 Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm

The dimensions are in mm and in brackets in inch.

### 5.2.7.4.4 Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 211 400 R0001 | TU517, terminal unit, 24 V DC, screw <br> terminals | Active |
| 1SAP 211 200 R0001 | TU518, terminal unit, 24 V DC, spring <br> terminals | Active |
| 1SAP 411 200 R0001 | TU518-XC, terminal unit, 24 V DC, <br> spring terminals, XC version | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.7.5 TU520-ETH for PROFINET communication interface modules

### 5.2.7.5.1 Features

- TU520-ETH, 2 RJ45 interfaces for connection to PROFIBUS network, 3 removable connectors for bus systems
- TU520-ETH-XC, 2 RJ45 interfaces for connection to PROFIBUS network, 3 removable connectors for bus systems, XC version


1 I/O bus (10 pins, female) to connect the first terminal unit
2a Plug (2 25 pins) to connect the inserted PROFINET communication interface module
2b Plug (3 19 pins) to connect the inserted PROFINET communication interface module
3 With a screwdriver, inserted in this place, the PROFINET I/O terminal unit and the adjacent I/O terminal unit can be shoved from each other
42 holes for wall mounting
53 removable connectors to connect the subordinated bus systems
62 RJ45 interfaces with indication LEDs for connection with the PROFINET network
76 spring terminals for process supply voltage (UP)
8 DIN rail

The PROFINET communication interface modules plug into the PROFINET IO terminal unit. When properly plugged-in, they are secured with two mechanical locks. All the connections are established via the PROFINET IO terminal unit, which allows removal and replacement of the communication interface modules without disturbing the wiring at the PROFINET IO terminal unit.

The PROFINET IO terminal unit TU520-ETH are specifically designed for use with AC500/S500 PROFINET communication interface modules (e. g. CI504-PNIO, CI506-PNIO).


## Extreme conditions

Terminal units for use in extreme ambient conditions have no ${ }_{x_{k}+\ldots}^{*}$ sign for XC version.

The figure 4 in the Part no. 1SAP4... (label) identifies the XC version.

## NOTICE!

Risk of corrosion!
Unused connectors and slots may corrode if XC devices are used in salt-mist environments.
Protect unused connectors and slots with TA535 protective caps for XC devices Chapter 5.2.8.3.6 "TA535-Protective caps for XC devices" on page 1332.

| Spring terminals | Conductor |  | Screwdriver (opens terminal) |
| :---: | :---: | :---: | :---: |

The terminals $1.0,2.0,3.0,1.1,2.1$ and 3.1 are electrically interconnected within the PROFINET IO terminal unit and always have the same assignment, irrespective of the inserted PROFINET communication interface module:

- Terminals 1.0, 2.0 and 3.0: process supply voltage UP $=+24 \mathrm{~V} D C$
- Terminals 1.1, 2.1 and 3.1: process supply voltage $\mathrm{ZP}=0 \mathrm{~V}$

The assignment of the bus system terminals depends on the inserted PROFINET communication interface module (see Ethernet communication interface modules overview).

### 5.2.7.5.2 Technical data

The system data of AC500 and S500 are applicable to the standard version $\Leftrightarrow$ Chapter 5.1.2 "System data AC500" on page 166.
The system data of AC500-XC are applicable to the XC version ${ }^{\leftrightarrows}$ Chapter 5.1.3 "System data AC500-XC" on page 169.

Only additional details are therefore documented below.
The technical data are also applicable to the XC version.

| Parameter | Value |
| :--- | :--- |
| Ethernet | $10 / 100$ base-TX or 100 base-TX (depending <br> on the plugged CI5xx module), 2 RJ45 socket |
| Number of bus system connectors | 3 (the type of bus system depends on <br> the PROFINET IO communication interface <br> module) |
| Rated voltage | 24 V DC |
| Max. permitted total current | 10 A via the supply terminals (UP and ZP) |
| Grounding | Direct connection to the grounded DIN rail or <br> via the screws with wall mounting |
| Spring terminals | Front terminal, conductor connection vertically <br> with respect to the printed circuit board |
| Weight | 200 g |
| Mounting position | Horizontal or vertical |

### 5.2.7.5.3 Dimensions



1 Din rail 15 mm
2 Din rail 7.5 mm


### 5.2.7.5.4 Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 214400 <br> R0001 | TU520-ETH, PROFINET I/O terminal <br> unit, 24 V DC, spring terminals | Active |
| 1SAP 414 400 <br> R0001 | TU520-ETH-XC, PROFINET I/O <br> terminal unit, 24 V DC, spring <br> terminals, XC version | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.8 Accessories

### 5.2.8.1 AC500-eCo

### 5.2.8.1.1 MC5102 - Micro memory card with adapter

- Solid state flash memory storage


1 Micro memory card
2 TA5350-AD micro memory card adapter, necessary for use in AC500 processor modules

The MC5102 micro memory card has no write protect switch.
The TA5350-AD micro memory card adapter has a write protect switch.
In the position "LOCK", the inserted micro memory card can only be read.

| Memory card type | AC500 V2 | AC500-XC <br> V2 | AC500 V3 | AC500-XC <br> V3 | AC500-eCo <br> V3 |
| :--- | :---: | :---: | :---: | :---: | :---: |
| MC5141 | x | x | x | x | - |
| MC5102 with TA5350-AD micro memory <br> card adapter | $\left.\mathrm{x}^{1}\right)^{2}$ | $\left.\left.\mathrm{x}^{1}\right)^{2}\right)^{2}$ | x | $\left.\mathrm{x}^{2}\right)$ | - |
| MC5102 without TA5350-AD micro <br> memory card adapter | - | - | - | - | x |

${ }^{1}$ ) As of firmware 2.5.x
${ }^{2}$ ) Temporary use of MC5102 is possible under normal environmental conditions, but MC5141 should be preferred.

The use of other micro memory cards is prohibited. ABB is not responsible nor liable for consequences resulting from use of unapproved memory cards.

Processor modules can be operated with and without (micro) memory card.
Processor modules are supplied without (micro) memory card. It must be ordered separately.

The micro memory card is used to store or backup application data and/or application programs or project source codes as well as to update the internal CPU firmware.
The micro memory card can only be used temporarily in standard and XC applications.
The memory card can be read/written on a PC with a SDHC compatible memory card reader when using TA5350-AD micro memory card adapter.
② Further information on using the micro memory card in AC500 PLCs

## Dimensions

Micro memory card


The dimensions are in mm and in brackets in inch.

Micro memory card adapter

2.1
(0.082)


Insert the micro memory card AC500 V3


Fig. 259: Insert micro memory card into PM56xx
1 Micro memory card
2 TA5350-AD micro memory card adapter
3 Memory card slot

1. Unpack the micro memory card and insert it into the supplied micro memory card adapter.
2. Insert the micro memory card adapter with integrated micro memory card into the memory card slot of the processor module until locked.

AC500-eCo V3


1 Micro memory card slot cover
2 Micro memory card
3 Micro memory card slot

1. Open the micro memory card slot cover by turning it upwards.
2. Carefully insert the micro memory card into the micro memory card slot as far as it will go. Observe orientation of card.
3. Close the micro memory card slot cover by turning it downwards.

## Remove the micro memory card

## NOTICE!

Disturbed PLC operation
Do not remove the micro memory card when it is working!
Otherwise the micro memory card and/or files on it might get corrupted and/or normal PLC operation might be disturbed.

- AC500 V3: Remove the micro memory card only when no black square ( is shown next to MC in the display.
- AC500-eCo V3: Remove the micro memory card only when the MC LED is not blinking.


Fig. 260: Remove micro memory card from PM56xx
1 Micro memory card
2 Micro memory card adapter
3 Memory card slot

1. To remove the micro memory card adapter with the integrated micro memory card, push on the micro memory card adapter until it moves forward.
2. By this, the micro memory card adapter is unlocked and can be removed.

## NOTICE!

## Disturbed PLC operation

Do not remove the memory card when it is working!
Otherwise the memory card and/or files on it might get corrupted and/or normal PLC operation might be disturbed.

- Remove the memory card only when no black square ( $\square$ ) is shown next to MC in the display.


1 Micro memory card slot cover
2 Micro memory card
3 Micro memory card slot

1. Open the micro memory card slot cover by turning it upwards.
2. Micro memory card can be removed from the micro memory card slot by gripping and pulling with two fingers.
3. Close the micro memory card slot cover by turning it downwards.

## Technical data

| Parameter | Value |
| :--- | :--- |
| Memory capacity | 8 GB |
| Total bytes written (TBW) | On request |
| Data retention |  |
|  | at beginning |
| when number of write processes has been <br> $90 \%$ of lifetime of each cell | 10 years at $+40^{\circ} \mathrm{C}$ |
| Write protect switch | year at $+40^{\circ} \mathrm{C}$ |
|  | Micro memory card |
|  | Micro memory card adapter |
| Weight | Yes |
| Dimensions | 0.25 g |

It is not possible to use 100 \% of a device's memory space. About $10 \%$ of the total available space must remain unused at any time to maintain normal device operation.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 180 100 R0002 | MC5102, micro memory <br> card with TA5350-AD micro <br> memory card adapter | Active |

> *) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.8.1.2 TA52xx(-x) - Terminal block sets

Intended pur- Removable terminal blocks are used for power supply and for I/O connectors on AC500-eCo V3 pose processor modules PM50x2.
For option boards there are different removable terminal blocks in spring version.

For the AC500-eCo V3 Basic CPUs a 3-pin terminal block for power supply and a 13-pin terminal block for I/O connectors are used.
For the AC500-eCo V3 Standard CPUs and Pro CPUs a 3-pin terminal block for power supply, a 13 -pin terminal block and a 12 -pin terminal block for I/O connectors are used.

For all CPUs there is a screw and a spring variant available.

| Basic CPU |  | Standard and Pro CPUs |  |
| :---: | :---: | :---: | :---: |
| Spring terminal TA5211-TSPF-B | Screw terminal TA5211-TSCL-B | Spring terminal TA5212-TSPF | Screw terminal <br> TA5212-TSCL |
|  |  |  |  |

Various removable spring terminal blocks are available for option boards.
The following spare parts are available (depending on the number of pins).

| Spring terminals |  |  |  |
| :---: | :---: | :---: | :---: |
| TA5220-SPF5 | TA5220-SPF6 | TA5220-SPF7 | TA5220-SPF8 |
|  |  |  |  |

## CAUTION!

Risk of injury and damaging the product!
Improper installation and maintenance may result in injury and can damage the product!

- Installation and maintenance have to be performed according to the technical rules, codes and relevant standards, e.g. EN 60204-1.
- Read product documentation carefully before wiring. Improper wiring or wrong terminal block from other devices can damage the product!
- Only by qualified personnel.


## CAUTION!

Risk of injury and damaging the module when using unapproved terminal blocks!
Only use terminal blocks approved by ABB to avoid injury and damage to the module.

## Terminal block set for PM50x2

Processor modules PM50x2 CPU are not delivered with terminal blocks.
Screw terminal block set:

- TA5211-TSCL-B (1SAP187400R0001) for PM5012-x-ETH
- TA5212-TSCL (1SAP187400R0004) for PM5032-x-ETH, PM5052-x-ETH, PM5072-T-2ETH(W), PM5082-T-2ETH

Spring terminal block set:

- TA5211-TSPF-B (1SAP187400R0002) for PM5012-x-ETH
- TA5212-TSPF (1SAP187400R0005) for PM5032-x-ETH, PM5052-x-ETH, PM5072-T-2ETH(W), PM5082-T-2ETH


## Dimensions

3-pin terminal block for power supply

## Screw terminal



## Spring terminal



13-pin terminal block for I/O connectors

## Screw terminal



## Spring terminal



12-pin terminal block for I/O connectors

## Screw terminal



## Spring terminal


x-PIN terminal blocks for option boards

Only these x-pin terminal blocks are available for the option boards.
TA5220-SPFx, with $\boldsymbol{x}=5 \ldots 8$


This results in these dimensions for the available spring terminal blocks.

| Description | Pin | Length [mm] | Wide [mm] | Height [mm] |
| :---: | :---: | :---: | :---: | :---: |
| TA5220-SPF5 | 5 | 18.2 | 7.7 | 22.9 |
| TA5220-SPF6 | 6 | 21.7 | 7.7 | 22.9 |
| TA5220-SPF7 | 7 | 25.2 | 7.7 | 22.9 |
| TA5220-SPF8 | 8 | 28.7 | 7.7 | 22.9 |

## Assembly



## Disassembly



Technical data Table 288: Screw terminal block for power supply

| Parameter | Value |
| :---: | :---: |
| Type |  |
| TA5211-TSCL-B | Removable 3-pin terminal block: screw front/cable side 5.00 mm pitch |
| TA5212-TSCL |  |
| Usage | Power supply for AC500-eCo V3 processor modules |
| Conductor cross section |  |
| Solid (copper) | $0.5 \mathrm{~mm}^{2} \ldots 2.5 \mathrm{~mm}^{2}$ |
| Flexible (copper) | $0.5 \mathrm{~mm}^{2} \ldots 2.5 \mathrm{~mm}^{2}$ |
| Stripped conductor end | 7 mm |


| Parameter |  |
| :--- | :--- |
| Fastening torque | Value |
| Dimensions |  |
| 0.5 Nm |  |
| Weight | $15 \mathrm{~mm} \times 12.4 \mathrm{~mm} \times 26.05 \mathrm{~mm}$ |
|  | TA5211-TSCL-B |
|  | TA5212-TSCL |

Table 289: Spring terminal block for power supply

| Parameter | Value |
| :---: | :---: |
| Type |  |
| TA5211-TSPF-B | Removable 3-pin terminal block: spring front/cable front 5.00 mm pitch |
| TA5212-TSPF |  |
| Usage | Power supply for AC500-eCo V3 processor modules |
| Conductor cross section |  |
| Solid (copper) | $0.5 \mathrm{~mm}^{2} \ldots 2.5 \mathrm{~mm}^{2}$ |
| Flexible (copper) | $0.5 \mathrm{~mm}^{2} \ldots 2.5 \mathrm{~mm}^{2}$ |
| Stripped conductor end | 11 mm |
| Dimensions |  |
| 3-pin terminal block | $15 \mathrm{~mm} \times 15 \mathrm{~mm} \times 25.95 \mathrm{~mm}$ |
| Weight |  |
| TA5211-TSPF-B | 150 g (2 terminal blocks) |
| TA5212-TSPF | 200 g (3 terminal blocks) |

Table 290: Screw terminal block for onboard I/Os

| Parameter | Value |
| :---: | :---: |
| Type |  |
| TA5211-TSCL-B | Removable 13-pin terminal block: screw front/cable side 5.00 mm pitch |
| TA5212-TSCL | Removable 13-pin and 12-pin terminal block: screw front/cable side 5.00 mm pitch |
| Usage | Onboard I/Os for AC500-eCo V3 processor modules |
| Conductor cross section |  |
| Solid (copper) | $0.5 \mathrm{~mm}^{2} \ldots 2.5 \mathrm{~mm}^{2}$ |
| Flexible (copper) | $0.5 \mathrm{~mm}^{2} \ldots 2.5 \mathrm{~mm}^{2}$ |
| Stripped conductor end | 7 mm |
| Fastening torque | 0.5 Nm |
| Dimensions |  |
| 13-pin terminal block | $65 \mathrm{~mm} \times 12.4 \mathrm{~mm} \times 26.05 \mathrm{~mm}$ |
| 12-pin terminal block | $60 \mathrm{~mm} \times 12.4 \mathrm{~mm} \times 26.05 \mathrm{~mm}$ |
| Weight |  |


| Parameter |  | Value |
| :--- | :--- | :--- |
| TA5211-TSCL-B | $150 \mathrm{~g}(2$ terminal blocks $)$ |  |
|  | TA5212-TSCL | 200 g (3 terminal blocks) |

Table 291: Spring terminal block for onboard I/Os

| Parameter | Value |
| :---: | :---: |
| Type |  |
| TA5211-TSPF-B | Removable 13-pin terminal block: spring front/cable front 5.00 mm pitch |
| TA5212-TSPF | Removable 13-pin and 12-pin terminal block: spring front/cable front 5.00 mm pitch |
| Usage | Onboard I/Os for AC500-eCo V3 processor modules |
| Conductor cross section |  |
| Solid (copper) | $0.5 \mathrm{~mm}^{2} \ldots 2.5 \mathrm{~mm}^{2}$ |
| Flexible (copper) | $0.5 \mathrm{~mm}^{2}$... $2.5 \mathrm{~mm}^{2}$ |
| Stripped conductor end | 11 mm |
| Dimensions |  |
| 13-pin terminal block | $65 \mathrm{~mm} \times 15 \mathrm{~mm} \times 25.95 \mathrm{~mm}$ |
| 12-pin terminal block | $60 \mathrm{~mm} \times 15 \mathrm{~mm} \times 25.95 \mathrm{~mm}$ |
| Weight |  |
| TA5211-TSPF-B | 150 g (2 terminal blocks) |
| TA5212-TSPF | 200 g (3 terminal blocks) |

Table 292: Spring terminal block for option boards

| Parameter | Value |
| :---: | :---: |
| Type |  |
| TA5220-SPF5 | Removable 5-pin terminal block: spring front, cable front 3.50 mm pitch |
| TA5220-SPF6 | Removable 6-pin terminal block: spring front, cable front 3.50 mm pitch |
| TA5220-SPF7 | Removable 7-pin terminal block: spring front, cable front 3.50 mm pitch |
| TA5220-SPF8 | Removable 8-pin terminal block: spring front, cable front 3.50 mm pitch |
| Usage | Connectors for AC500-eCo V3 option boards |
| Conductor cross section |  |
| Solid (copper) | $0.2 \mathrm{~mm}^{2} \ldots 1.5 \mathrm{~mm}^{2}$ |
| Flexible (copper) | $0.2 \mathrm{~mm}^{2} \ldots 1.5 \mathrm{~mm}^{2}$ |
| Stripped conductor end | $8 \mathrm{~mm} . .110 \mathrm{~mm}$ |
| Dimensions |  |
| TA5220-SPF5 | $18.2 \mathrm{~mm} \times 7.7 \mathrm{~mm} \times 22.9 \mathrm{~mm}$ |
| TA5220-SPF6 | $21.7 \mathrm{~mm} \times 7.7 \mathrm{~mm} \times 22.9 \mathrm{~mm}$ |


| Parameter |  | Value |
| :--- | :--- | :--- |
|  | TA5220-SPF7 | $25.2 \mathrm{~mm} \times 7.7 \mathrm{~mm} \times 22.9 \mathrm{~mm}$ |
|  | TA5220-SPF8 | $28.7 \mathrm{~mm} \times 7.7 \mathrm{~mm} \times 22.9 \mathrm{~mm}$ |
| Weight | 150 g |  |
|  | TA5220-SPF5 | 170 g |
| TA5220-SPF6 | 180 g |  |
| TA5220-SPF7 | 200 g |  |
|  | TA5220-SPF8 |  |

## Ordering data

Table 293: Terminal block sets for AC500-eCo V3 Basic

| Part no. | Description |
| :---: | :---: |
| 1SAP 187400 R0001 | TA5211-TSCL-B: screw terminal block set for AC500-eCo V3 CPU Basic screw front, cable side 5.00 mm pitch <br> - 1 removable 3-pin terminal block for power supply <br> - 1 removable 13-pin terminal block for I/O connectors |
| 1SAP 187400 R0002 | TA5211-TSPF-B: spring terminal block set for AC500-eCo V3 CPU Basic spring front, cable front 5.00 mm pitch <br> - 1 removable 3-pin terminal block for power supply <br> - 1 removable 13-pin terminal block for I/O connectors |

Table 294: Terminal block sets for AC500-eCo V3 Standard and Pro

| Part no. | Description |
| :---: | :---: |
| 1SAP 187400 R0004 | TA5212-TSCL: screw terminal block set for AC500-eCo V3 Standard and Pro CPU <br> screw front, cable side 5.00 mm pitch <br> - 1 removable 3-pin terminal block for power supply <br> - 1 removable 13-pin terminal block for I/O connectors <br> - 1 removable 12-pin terminal block for I/O connectors |
| 1SAP 187400 R0005 | TA5212-TSPF: spring terminal block set for AC500-eCo V3 Standard and Pro CPU spring front, cable front 5.00 mm pitch <br> - 1 removable 3-pin terminal block for power supply <br> - 1 removable 13 -pin terminal block for I/O connectors <br> - 1 removable 12 -pin terminal block for I/O connectors |

Table 295: Spare parts

| Part no. | Description |
| :--- | :--- |
| 1SAP 187 400 R0012 | TA5220-SPF5: spring terminal block, removable, 5-pin, spring front, <br> cable front, 6 pieces per packing unit |
| 1SAP 187 400 R0013 | TA5220-SPF6: spring terminal block, removable, 6-pin, spring front, <br> cable front, 6 pieces per packing unit |
| 1SAP 187 400 R0014 | TA5220-SPF7: spring terminal block, removable, 7-pin, spring front, <br> cable front, 6 pieces per packing unit |
| 1SAP 187 400 R0015 | TA5220-SPF8: spring terminal block, removable, 8-pin, spring front, <br> cable front, 6 pieces per packing unit |

### 5.2.8.1.3 TA5300-CVR - Option board slot cover

Intended pur- TA5300-CVR option board slot covers for PM50xx processor modules are necessary to protect pose not used option board slots.


## CAUTION!

## Risk of injury and damaging the product!

Always plug in the option board slot cover when the option board is not inserted.
If the option board slot cover is lost, please order the replacement TA5300-CVR (1SAP187500R0001).

Never power up the CPU with uncovered option board slot, otherwise it may cause serious injury and/or damage the product.

The AC500-eCo V3 processor modules are delivered with option board slot cover(s).
The option board slot cover has to be removed before inserting an option board.
The TA5300-CVR option board slot covers are available as spare parts.

## Dimensions



The dimensions are in mm and in brackets in inch.

Inserting of the option board slot cover


1. Press on the option board slot cover to insert it in the not used option board slot of the processor module PM50xx.
2. The option board slot cover must click into the not used option board slot.

## Removing of the

 option board slot cover

1. Press the side of the inserted option board slot cover.
2. At the same time, pull the option board slot cover out of the option board slot of the processor module PM50xx.

Technical data The system data of AC500-eCo V3 apply ${ }^{4}$ ) Chapter 5.1.1 "System data AC500-eCo" on page 159

Only additional details are therefore documented below.

| Parameter | Value |
| :--- | :--- |
| Weight | 47 g |
| Dimensions | $42.1 \mathrm{~mm} \times 30.8 \mathrm{~mm} \times 23.55$ |

Ordering data

| Part no. | Description | Product life cycle phase <br> $*$ |
| :--- | :--- | :--- |
| 1SAP 187 500 R0001 | TA5300-CVR: option board slot cover, <br> removable plastic part, 6 pieces per <br> packing unit | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.8.1.4 TA5400-SIM - Input simulator

- TA5400-SIM input simulator for 6 digital inputs 24 V DC
- For usage with AC500-eCo V3 processor modules


1 Contacts for connecting the input simulator to the terminal block for I/O connectors
26 switches for the digital inputs DIO ... DI5 (0 means opened switch, 1 means closed switch)
3 Screw terminal block for power supply
4 Screw terminal block(s) for I/O connectors

Intended purpose

## TA5400-SIM

The TA5400-SIM input simulator is only intended for testing and training purposes for AC500-eCo V3 processor modules PM50x2.
Continuous operation in a productive system is not permitted.
The TA5400-SIM input simulator may only be used with screw-type terminal blocks.

The TA5400-SIM input simulator must not be used with spring-type terminal blocks.

## Environmental conditions for testing and training purposes

In order not to impair the functionality of the product, avoid any kind of disturbing environmental influences:

- mechanical disturbances
- climatic influences

Make sure that the parameters are within the normal range:

- temperature
- air pressure
- humidity
- altitude

The TA5400-SIM input simulator can simulate 6 digital 24 V DC input signals to the digital inputs IO ... I5 of onboard I/Os.

With the TA5400-SIM input simulator, the digital 24 V DC inputs $10 \ldots 15$ can be turned OFF and ON separately:

- If the lever of the switch is on the right side (1), the input is ON.
- If the lever of the switch is on the left side (0), the input is OFF.


## Dimensions



The dimensions are in mm and in brackets in inch.

## Electrical diagram

The diagram below shows the connection of the TA5400-SIM input simulator.


## NOTICE!

Risk of damage to the TA5400-SIM input simulator!
Do not remove the terminal block while the TA5400-SIM input simulator is connected.

Do not apply mechanical forces to the input simulator when it is connected to the terminal block.

In both cases the input simulator could be damaged.

## Assembly

Insertion of the input simulator

1. Make sure that the power supply of the processor module is turned off.

## CAUTION!

Risk of damaging the PLC modules!
The PLC modules can be damaged by overvoltages and short circuits.
Make sure, that all voltage sources (supply and process voltage) are switched off before you start working on the system.

Never connect voltages > 24 V DC to the terminal block of the TA5400SIM input simulator.

## Risk of damaging the input simulator and/or PLC modules!

CAUTION!

The TA5400-SIM input simulator may only be used with AC500-eCo V3 processor modules PM50x2.

Never use the input simulator with other devices.
The input simulator may only be used with screw-type terminal blocks.
The input simulator is only intended for testing and training purposes.
Never use it within productive sytems.
2. Make sure that all clamps of the onboard I/Os are totally open.
3. Insert the TA5400-SIM input simulator into the screw terminal block as shown in the figure.

4. Tighten all screws of the onboard I/O clamps.
5. Make sure all switches are in OFF state (0).
6. Connect 24 V DC to the power supply of the TA5400-SIM ( $L+$ and M). Tighten the screws.
7. Connect the processor module power supply wires (24 V DC) ${ }^{\wedge}$ "Pin assignment" on page 184.

## Disassembly

Removal of the input simulator

1. Make sure that the power supply of the processor module is turned off.

## CAUTION!

## Risk of damaging the PLC modules!

The PLC modules can be damaged by overvoltages and short circuits.
Make sure that all voltage sources (supply and process voltage) are switched off before you start working on the system.
2. Disconnect the TA5400-SIM power supply wires ( 24 V DC) with a flat-blade screwdriver from the terminal block for power supply ( $L+$ and $M$ ).
3. Loosen all screws of the onboard I/Os.
4. Remove the input simulator by pulling it to the left side.

Technical data The system data of AC500-eCo V3 apply ${ }^{4}$ ) Chapter 5.1.1 "System data AC500-eCo" on page 159
Only additional details are therefore documented below.

Table 296: Technical data of the module

| Parameter |  |
| :--- | :--- |
| Process supply voltage |  |
| Connections | Terminal (L+) for +24 V DC and terminal (M) for <br> 0 V DC |
| Rated value | 24 V DC |
| Max. ripple | $5 \%$ |
| Weight | 18 g |
| Mounting position | Horizontal or vertical |

Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 187 600 R0001 | TA5400-SIM, input simulator <br> for PM50x2 | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.8.2 AC500 and AC500-XC

### 5.2.8.2.1 MC5102 - Micro memory card with adapter

- Solid state flash memory storage


1 Micro memory card
2 TA5350-AD micro memory card adapter, necessary for use in AC500 processor modules


The MC5102 micro memory card has no write protect switch.
The TA5350-AD micro memory card adapter has a write protect switch. In the position "LOCK", the inserted micro memory card can only be read.

| Memory card type | AC500 V2 | AC500-XC <br> V2 | AC500 V3 | AC500-XC <br> V3 | AC500-eCo <br> V3 |
| :--- | :---: | :---: | :---: | :---: | :---: |
| MC5141 | x | x | x | x | - |
| MC5102 with TA5350-AD micro memory <br> card adapter | $\mathrm{x}^{1}$ ) | $\left.\left.\mathrm{x}^{1}\right)^{2}\right)^{2}$ | x | $\mathrm{x}^{2}$ ) | - |
| MC5102 without TA5350-AD micro <br> memory card adapter | - | - | - | - | x |

${ }^{1}$ ) As of firmware 2.5.x
${ }^{2}$ ) Temporary use of MC5102 is possible under normal environmental conditions, but MC5141 should be preferred.

The use of other micro memory cards is prohibited. $A B B$ is not responsible nor liable for consequences resulting from use of unapproved memory cards.

Purpose
Processor modules can be operated with and without (micro) memory card.
Processor modules are supplied without (micro) memory card. It must be ordered separately.

The micro memory card is used to store or backup application data and/or application programs or project source codes as well as to update the internal CPU firmware.

The micro memory card can only be used temporarily in standard and XC applications.
The memory card can be read/written on a PC with a SDHC compatible memory card reader when using TA5350-AD micro memory card adapter.
« Further information on using the micro memory card in AC500 PLCs

## Dimensions

Micro memory card




The dimensions are in mm and in brackets in inch.

Micro memory card adapter

2.1


The dimensions are in mm and in brackets in inch.

Insert the micro
memory card
AC500 V3


Fig. 261: Insert micro memory card into PM56xx
1 Micro memory card
2 TA5350-AD micro memory card adapter
3 Memory card slot

1. Unpack the micro memory card and insert it into the supplied micro memory card adapter.
2. Insert the micro memory card adapter with integrated micro memory card into the memory card slot of the processor module until locked.

AC500-eCo V3


1 Micro memory card slot cover
2 Micro memory card
3 Micro memory card slot

1. Open the micro memory card slot cover by turning it upwards.
2. Carefully insert the micro memory card into the micro memory card slot as far as it will go. Observe orientation of card.
3. Close the micro memory card slot cover by turning it downwards.

## Remove the micro memory card

## NOTICE!

Disturbed PLC operation
Do not remove the micro memory card when it is working!
Otherwise the micro memory card and/or files on it might get corrupted and/or normal PLC operation might be disturbed.

- AC500 V3: Remove the micro memory card only when no black square ( is shown next to MC in the display.
- AC500-eCo V3: Remove the micro memory card only when the MC LED is not blinking.


Fig. 262: Remove micro memory card from PM56xx
1 Micro memory card
2 Micro memory card adapter
3 Memory card slot

1. To remove the micro memory card adapter with the integrated micro memory card, push on the micro memory card adapter until it moves forward.
2. By this, the micro memory card adapter is unlocked and can be removed.

## NOTICE!

## Disturbed PLC operation

Do not remove the memory card when it is working!
Otherwise the memory card and/or files on it might get corrupted and/or normal PLC operation might be disturbed.

- Remove the memory card only when no black square ( $\square$ ) is shown next to MC in the display.


1 Micro memory card slot cover
2 Micro memory card
3 Micro memory card slot

1. Open the micro memory card slot cover by turning it upwards.
2. Micro memory card can be removed from the micro memory card slot by gripping and pulling with two fingers.
3. Close the micro memory card slot cover by turning it downwards.

## Technical data

| Parameter | Value |
| :--- | :--- |
| Memory capacity | 8 GB |
| Total bytes written (TBW) | On request |
| Data retention |  |
|  | at beginning |
| when number of write processes has been <br> $90 \%$ of lifetime of each cell | 10 years at $+40^{\circ} \mathrm{C}$ |
| Write protect switch | year at $+40^{\circ} \mathrm{C}$ |
|  | Micro memory card |
|  | Micro memory card adapter |
| Weight | Yes |
| Dimensions | 0.25 g |

It is not possible to use 100 \% of a device's memory space. About $10 \%$ of the total available space must remain unused at any time to maintain normal device operation.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 180 100 R0002 | MC5102, micro memory <br> card with TA5350-AD micro <br> memory card adapter | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.8.2.2 MC5141-Memory card

- Solid state flash memory storage


1 MC5141 memory card

The memory card has a write protect switch.
In the position "LOCK", the memory card can only be read.

| Memory card type | AC500 V2 | AC500-XC <br> V2 | AC500 V3 | AC500-XC <br> V3 | AC500-eCo <br> V3 |
| :--- | :---: | :---: | :---: | :---: | :---: |
| MC5141 | x | x | x | x | - |
| MC5102 with TA5350-AD micro memory <br> card adapter | $\left.\mathrm{x}^{1}\right)^{2}$ | $\left.\left.\mathrm{x}^{1}\right)^{2}\right)^{2}$ | x | $\left.\mathrm{x}^{2}\right)$ | - |
| MC5102 without TA5350-AD micro <br> memory card adapter | - | - | - | - | x |

${ }^{1}$ ) As of firmware 2.5.x
${ }^{2}$ ) Temporary use of MC5102 is possible under normal environmental conditions, but MC5141 should be preferred.

The use of other memory cards is prohibited. ABB is not responsible nor liable for consequences resulting from use of unapproved memory cards.

Purpose
Processor modules can be operated with and without (micro) memory card.
Processor modules are supplied without (micro) memory card. It must be ordered separately.

The memory card is used to store or backup application data and/or application programs or project source codes as well as to update the internal CPU firmware.
The memory card is intended for long-term use in standard and XC application.
The memory card can be read/written on a PC with a SDHC compatible memory card reader.』 Further information on using the memory card in AC500 PLC

## Dimensions

Insert the memory card

## AC500 V3

1. Unpack the memory card.
2. Insert the memory card into the memory card slot of the processor module until locked.


Fig. 263: Insert memory card into PM56xx
1 Memory card
2 Memory card slot

Remove the memory card

AC500 V3

## NOTICE!

## Disturbed PLC operation

Do not remove the memory card when it is working!
Otherwise the memory card and/or files on it might get corrupted and/or normal PLC operation might be disturbed.

- Remove the memory card only when no black square ( ) is shown next to MC in the display.

1. To remove the memory card, push on the memory card until it moves forward.
2. By this, the memory card is unlocked and can be removed.


Fig. 264: Remove memory card from PM56xx
1 Memory card
2 Memory card slot

## Technical data

| Parameter | Value |
| :--- | :--- |
| Memory capacity | 2 GB |
| Total bytes written (TBW) | On request |
| Data retention |  |
|  | at beginning |
| when number of write processes has been <br> $90 \%$ of lifetime of each cell | 10 years at $+40^{\circ} \mathrm{C}$ |
| Write protect switch | Yes, at the edge of the memory card |
| Weight | 2 g |
| Dimensions | $24 \mathrm{~mm} \times 32 \mathrm{~mm} \times 2.1 \mathrm{~mm}$ |

It is not possible to use $100 \%$ of a device's memory space. About $10 \%$ of the total available space must remain unused at any time to maintain normal device operation.

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 180 100 R0041 | MC5141, memory card | Active |

*) Modules in lifecycle Classic are available from stock but not recommended
for planning and commissioning of new installations.

### 5.2.8.2.3 TA521 - Battery

- Manganese dioxide lithium battery, 3 V, 560 mAh
- Non-rechargeable


Purpose The TA521 battery is the only applicable battery for the AC500 processor modules ${ }^{\wedge} \Rightarrow$ Chapter 5.2.1.2.2 "PM56xx-2ETH for AC500 V3 products" on page 313. It cannot be recharged.

The processor modules are supplied without lithium battery. It must be ordered separately. The TA521 lithium battery is used for data (SRAM) and RTC buffering while the processor module is not powered.
※ Chapter 6.8.2.3.2 "AC500 battery" on page 4445
The CPU monitors the discharge degree of the battery. A warning is issued before the battery condition becomes critical (about 2 weeks before). Once the warning message appears, the battery should be replaced as soon as possible.

Handling instructions

- Do not short-circuit or re-charge the battery! It can cause excessive heating and explosion.
- Do not disassemble the battery!
- Do not heat up the battery and not put into fire! Risk of explosion.
- Store the battery in a dry place.
- Replace the battery with supply voltage ON in order not to risk data being lost.
- Recycle exhausted batteries meeting the environmental standards.


Battery lifetime The battery lifetime is the time, the battery can store data while the processor module is not powered. As long as the processor module is powered, the battery will only be discharged by its own leakage current.

To avoid a short battery discharge, the battery should always be inserted or replaced while the process module is under power, then the battery is correctly recognized and will not shortly discharged.

## Insertion

To ensure propper operation and to prevent data loss, the battery insertion or replacement must be always done with the system under power. Without battery and power supply there is no data buffering possible.

## WARNING!

## Risk of fire or explosion!

Use of incorrect Battery may cause fire or explosion.


1. Open the battery compartment with the small locking mechanism, press it down and slip down the door. The door is attached to the front face of the processor module and cannot be removed.
2. Remove the TA521 battery from its package and hold it by the small cable. Remove then the small connector from the socket, do this best by lifting it out with a screwdriver.
3. Insert the battery connector into the small connector port of the compartment. The connector is keyed to find the correct polarity (red = positive pole = above).
4. Insert first the cable and then the battery into the compartment, push it until it reaches the bottom of the compartment.
5. Arrange the cable in order not to inhibit the door to close.
6. Pull-up the door and press until the locking mechanism snaps.

| In order to prevent data losses or problems, the battery should be replaced after |
| :--- |
| 3 years of utilisation or at least as soon as possible after receiving the "low |
| battery warning" indication. |
| Do not use a battery older than 3 years for replacement, do not keep batteries <br> too long in stock. |

## Replacement of the battery

To ensure propper operation and to prevent data loss, the battery insertion or replacement must be always done with the system under power. Without battery and power supply there is no data buffering possible.

1. Open the battery compartment with the small locking mechanism, press it down and slip down the door. The door is attached to the front view of the processor module and cannot be removed.
2. Remove the old TA521 battery from the battery compartment by pulling it by the small cable. Remove then the small connector from the socket, do this best by lifting it out with a screwdriver.

3. Follow the previous instructions to insert a new battery.

## CAUTION!

## Risk of explosion!

Do not open, re-charge or disassemble lithium batteries. Attempting to charge lithium batteries will lead to overheating and can cause explosions.
Protect them from heat and fire and store them in a dry place.
Never short-circuit or operate lithium batteries with the polarities reversed. The batteries are likely to overheat and explode. Avoid unintentional short circuiting do not store batteries in metal containers and do not place them on metallic surfaces. Escaping lithium is a health hazard.

In order to prevent data losses or problems, the battery should be replaced after 3 years of utilisation or at least as soon as possible after receiving the "low battery warning" indication.
Do not use a battery older than 3 years for replacement, do not keep batteries too long in stock.

## Technical data

| Parameter | Value |
| :--- | :--- |
| Nominal voltage | 3 V |
| Nominal capacity | 560 mAh |
| Temperature range (index below C 0 ) | Operating: $0^{\circ} \mathrm{C} \ldots+60^{\circ} \mathrm{C}$ <br> Storage: $-20^{\circ} \mathrm{C} \ldots+60^{\circ} \mathrm{C}$ <br>  <br>  <br> Transport: $-20^{\circ} \mathrm{C} \ldots+60^{\circ} \mathrm{C}$ |


| Parameter | Value |
| :--- | :--- |
| Temperature range (index C 0 and above) | Operating: $-40^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$ <br> Storage: $-40^{\circ} \mathrm{C} \ldots+85^{\circ} \mathrm{C}$ <br>  <br>  <br> Transport: $-40^{\circ} \mathrm{C} \ldots+85^{\circ} \mathrm{C}$ |
| Battery lifetime | Typ. 3 years at $+25^{\circ} \mathrm{C}$ |
| Self-discharge | $2 \%$ per year at $+25^{\circ} \mathrm{C}$ <br> $5 \%$ per year at $+40^{\circ} \mathrm{C}$ <br>  <br>  <br> $20 \%$ per year at $+60^{\circ} \mathrm{C}$ |
| Protection against reverse polarity | Yes, by mechanical coding of the plug. |
| Insulation | The battery is completely insulated. |
| Connection | Red $=$ positive pole $=$ above at plug, black $=$ <br> negative pole, |
| Weight | 7 g |
| Dimensions | Diameter of the button cell: 24.5 mm <br> Thickness of the button cell: 5 mm |

## Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 180300 R0001 | TA521, lithium battery | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.8.2.4 TA524 - Dummy communication module


$\begin{array}{ll}1 & \text { Type } \\ 2 & \text { Label }\end{array}$

Purpose TA524 is used to cover an unused communication module slot of a terminal base. It protects the terminal base from dust and inadvertent touch.
ゃ Chapter 5.2.3.1 "TB56xx for AC500 V3 products" on page 377

Handling TA524 is mounted in the same way as a common communication module.
instructions
*) Chapter 4.3.5 "Mounting and demounting the communication module" on page 109

Technical data

| Parameter | Value |
| :--- | :--- |
| Weight | 50 g |
| Dimensions | $135 \mathrm{~mm} \times 28 \mathrm{~mm} \times 62 \mathrm{~mm}$ |

Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 180 600 R0001 | TA524, dummy communica- <br> tion module | Active |

> *) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.8.2.5 TA526 - Wall mounting accessory



Purpose If a terminal base or a terminal unit should be mounted with screws, the wall mounting accessories TA526 must be inserted at the rear side first. This plastic parts prevent bending of terminal bases and terminal units while screwing up.
$\leftrightarrow$ Handling of the wall mounting accessory with terminal units
\& Handling of the wall mounting accessory with terminal bases

Technical data

| Parameter | Value |
| :--- | :--- |
| Weight | 5 g |
| Dimensions | $67 \mathrm{~mm} \times 35 \mathrm{~mm} \times 5,5 \mathrm{~mm}$ |

Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 180 800 R0001 | TA526, wall mounting acces- <br> sory | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.8.3 S500 and S500-XC

### 5.2.8.3.1 CP-E - Economic range



The power supplies feature series and parallel connection as well as a true redundant setup via a redundancy module.

- Wide-range input voltage
- Mounting on DIN rail
- High efficiency of up to $90 \%$
- Low power dissipation and low heating
- Wide ambient temperature range from $-40^{\circ} \mathrm{C} \ldots+70^{\circ} \mathrm{C}$
- No-load-proof, overload-proof, continuous short-circuit-proof
- Power factor correction (depending on the type)
- Approved in accordance with all relevant international standards

Table 297: Ordering data

| Order No. | Type | Input | Output | Overload capacity | Module width [mm] |
| :---: | :---: | :---: | :---: | :---: | :---: |
| 1SVR427030R0000 | $\begin{aligned} & \text { CP-E } \\ & 24 / 0.75 \end{aligned}$ | $\begin{aligned} & 100-240 \mathrm{~V} \\ & \text { AC or } \\ & 120-370 \mathrm{~V} \\ & \mathrm{DC} \end{aligned}$ | $\begin{aligned} & 24 \mathrm{~V} D, \\ & 0.75 \mathrm{~A} \end{aligned}$ | - | 22.5 |
| 1SVR427031R0000 | $\begin{aligned} & \text { CP-E } \\ & 24 / 1.25 \end{aligned}$ | $\begin{aligned} & 100 \text { V AC ... } \\ & 240 \text { V AC or } \\ & 90 \mathrm{~V} \text { DC ... } \\ & 375 \mathrm{~V} \text {. } \end{aligned}$ | $\begin{aligned} & 24 \mathrm{~V} \mathrm{DC}, \\ & 1.25 \mathrm{~A} \end{aligned}$ | - | 40.5 |
| 1SVR427032R0000 | CP-E 24/2.5 | $\begin{aligned} & 100 \text { V AC ... } \\ & 240 \text { V AC or } \\ & 90 \text { V DC ... } \\ & 375 \text { V DC } \end{aligned}$ | $\begin{aligned} & 24 \mathrm{~V} D C, 2.5 \\ & \mathrm{~A} \end{aligned}$ | - | 40.5 |
| 1SVR427034R0000 | CP-E 24/5.0 | 115/230 V <br> AC auto select or 210 <br> V DC ... 370 <br> V DC | 24 V DC, 5 A | - | 63.2 |
| 1SVR427035R0000 | $\begin{aligned} & \hline \text { CP-E } \\ & 24 / 10.0 \end{aligned}$ | 115/230 V AC auto select or 210 <br> V DC ... 370 <br> V DC | $\begin{aligned} & 24 \mathrm{~V} \text { DC, } 10 \\ & \mathrm{~A} \end{aligned}$ | - | 83 |
| 1SVR427036R0000 | $\begin{aligned} & \text { CP-E } \\ & 24 / 20.0 \end{aligned}$ | $\begin{aligned} & 115 \text { V AC ... } \\ & 230 \text { V AC or } \\ & 120 \text { V DC ... } \\ & 370 \text { V DC } \end{aligned}$ | $\begin{aligned} & 24 \mathrm{~V} \text { DC, } 20 \\ & \mathrm{~A} \end{aligned}$ | - | 175 |

### 5.2.8.3.2 CP-C. 1 - High performance range



The power supplies feature series and parallel connection as well as a true redundant setup via a redundancy module.

The CP-C. 1 power supplies are ABB's high performance and most advanced range. With excellent efficiency, high reliability and innovative functionality it is prepared for the most demanding industrial applications. These power supplies have a $50 \%$ integrated power reserve and operate at an efficiency of up to $94 \%$. They are equipped with overheat protection and active power factor correction. Combinded with a broad AC and DC input range and extensive worldwide approvals the CP-C. 1 power supplies are the preferred choice for professional DC applications.

- Typical efficiency of up to $94 \%$
- Power reserve design delivers up to $150 \%$ of the nominal output current
- Signaling outputs for DC OK and power reserve mode
- High power density leads to very compact and small devices
- No-load-proof, overload-proof, continuous short-circuit-proof
- Active power factor correction (PFC)

Table 298: Ordering data

| Order No. | Type | Input | Output | Overload capacity | Module width [mm] |
| :---: | :---: | :---: | :---: | :---: | :---: |
| 1SVR360563R1001 | $\begin{aligned} & \text { CP-C. } 1 \\ & 24 / 5.0 \end{aligned}$ | $\begin{aligned} & 110 \text { V AC ... } \\ & 240 \text { V AC or } \\ & 90 \text { V DC ... } \\ & 300 \text { V DC } \end{aligned}$ | 24 V DC, 5 A | +50 \% | 40 |
| 1SVR360663R1001 | $\begin{aligned} & \text { CP-C. } 1 \\ & 24 / 10.0 \end{aligned}$ | $\begin{aligned} & 110 \text { V AC ... } \\ & 240 \text { V AC or } \\ & 90 \text { V DC ... } \\ & 300 \text { V DC } \end{aligned}$ | $\begin{aligned} & 24 \mathrm{~V} D C, 10 \\ & \mathrm{~A} \end{aligned}$ | +50 \% | 60 |
| 1SVR360763R1001 | $\begin{aligned} & \text { CP-C. } 1 \\ & \text { 24/20.0 } \end{aligned}$ | $\begin{aligned} & 110 \text { V AC ... } \\ & 240 \text { V AC or } \\ & 90 \text { V DC ... } \\ & 300 \text { V DC } \end{aligned}$ | $\begin{aligned} & 24 \mathrm{~V} \text { DC, } 20 \\ & \mathrm{~A} \end{aligned}$ | +30 \% | 82 |

5.2.8.3.3 TA523-Pluggable label mounting

For labelling the channels of S500 I/O modules.


1 Pluggable label mounting TA523
2 Plastic labels to be inserted into the holder

Purpose $\quad$ The pluggable label mounting is used to hold 4 plastic labels, on which the meaning of the I/O channels of I/O modules can be written down. The holder is transparent so that after snapping it onto the module the LEDs shine through.

Handling instructions

The plastic labels can be printed out from TA523.doc.

Technical data

| Parameter | Value |
| :--- | :--- |
| Use | For labelling channels of I/O modules |
| Mounting | Snap-on to the module |
| Weight | 20 g |
| Dimensions | $82 \mathrm{~mm} \times 67 \mathrm{~mm} \times 13 \mathrm{~mm}$ |

Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 180 500 R0001 | TA523, pluggable label <br> mounting (10 pieces) | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.8.3.4 TA525 - Plastic labels

Accessory to label AC500 and S500 modules.


1 Module without plastic label TA525
2 Module with plastic label TA525

Purpose The plastic labels are suitable for labelling AC500 and S500 modules (CPUs, communication modules and I/O modules). The small plastic parts can be written on with a standard waterproof pen.

Handling instructions

The plastic labels are inserted under a slight pressure. For disassembly, a small screwdriver is inserted at the lower edge of the module.

| Parameter | Value |
| :--- | :--- |
| Use | For labelling AC500 and S500 modules |
| Mounting | Insertion under a slight pressure |


| Parameter | Value |
| :--- | :--- |
| Disassembly | With a small screwdriver |
| Scope of delivery | 10 pieces |
| Weight | 1 g per piece |
| Dimensions | $8 \mathrm{~mm} \times 20 \mathrm{~mm} \times 5 \mathrm{~mm}$ |


| Ordering data | Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- | :--- |
| 1SAP 180 700 R0001 | TA525, Set of 10 white plastic <br> labels | Active |  |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.8.3.5 TA526 - Wall mounting accessory



Purpose If a terminal base or a terminal unit should be mounted with screws, the wall mounting accessories TA526 must be inserted at the rear side first. This plastic parts prevent bending of terminal bases and terminal units while screwing up.
$\left.{ }^{4}\right)$ Handling of the wall mounting accessory with terminal units
« Handling of the wall mounting accessory with terminal bases

Technical data

| Parameter | Value |
| :--- | :--- |
| Weight | 5 g |
| Dimensions | $67 \mathrm{~mm} \times 35 \mathrm{~mm} \times 5,5 \mathrm{~mm}$ |

Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 180 800 R0001 | TA526, wall mounting acces- <br> sory | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.8.3.6 TA535 - Protective caps for XC devices

Purpose Accessory to cover unused connectors of XC devices in salt mist environments.
One TA535 package includes different cap types for the following connectors:

- RJ45 connectors
- 9-pole D-sub connector
- FieldBusPlug connector

Protection should be done for all unused slots of -XC devices.

Ordering data

| Part no. | Description | Product life cycle phase *) |
| :--- | :--- | :--- |
| 1SAP 182 300 R0001 | TA535, Protective Caps for <br> XC devices | Active |

*) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

### 5.2.9 Storage devices

### 5.2.9.1 Overview

AC500 PLCs offer a variety of storage devices. The following table gives a short overview and a description on these storage devices:

IEC access means that the storage device can be accessed by function blocks of an IEC program.

FTP access means that the device can be accessed via FTP server on the PLC (if available).

| Component | Description | IEC access | FTP access | Processor <br> modules |
| :--- | :--- | :--- | :--- | :--- |
| userdisk <br> home/userdisk <br> (customer data) | User disk for <br> custom data <br> (flash) <br> Internal persis- <br> tent mass <br> storage placed in <br> the internal flash <br> device <br> Can be used for <br> any application <br> purpose | Yes | Yes | All |
| PLCLogic <br> home/PLCLogic <br> (customer data) | Internal persis- <br> tent mass <br> storage placed in <br> the internal flash <br> device <br> Used for configu- <br> ration data, user <br> application (boot <br> project), WebVisu <br> files, etc. | Yes | Yes | All |
|  | Battery-buffered <br> device, non-vola- <br> tile RAM <br> Used for retain/ <br> persistent and <br> \%M variables | Yes | No | All |
| SRAM | System RAM <br> disk (Temp direc- <br> tory) for storing <br> the firmware | Yes | No | All |
| For internal firm- <br> ware use only! | Yes | Pes |  |  |
| system | Internal persis- <br> tent mass <br> storage device <br> Can be used for <br> any application <br> purpose | Yes | Pes <br> memory card <br> (removable) <br> Removable per- <br> sistent mass <br> storage device <br> Can be used for <br> any application <br> purpose | Yes |

### 5.2.9.2 Functionalities

| Filesystem Name | As of CPU firmware | Description |
| :---: | :---: | :---: |
| userdisk | V3.0.0 | Boot project (size depends on PLC type) WebVisu files for web server <br> Symbol file for OPC server and CP600 panels <br> User data via CAA_File_xxx.lib *) <br> Files via Automation Builder file download <br> Files via FTP server |
|  | V3.1.0 | Save persistent data |
| SRAM | V3.1.0 | Save retain and persistent data |
| system | V3.0.0 | Load / save boot project |
|  |  | Firmware update |
|  |  | Internal system files |
| flash disk | V3.1.0 | User data via CAA_File_xxx.lib *) <br> Files via Automation Builder file download Files via FTP server |
| sdcard | V3.0.0 | Firmware update, <br> User data via CAA_File_xxx.lib *) <br> Files via Automation Builder file download <br> Files via FTP server |
|  | V3.1.0 | Save persistent data <br> Boot project (size depends on PLC type) |

*) Examples for the filename with path (sFileName for FILE.Open) specified by the user ('mydir' is optional, but must be an existing directory):

- 'userdisk/myfile.txt'
- 'sdcard/mydir/myfile.txt'
- 'flashdisk/myfile.txt'

The maximum number of files opened at the same time is limited to 1007.
The max. length of the user string (path and filename) is 241 characters.

Unlike the PLC's memory areas like \%M or Retain, where 1 byte actually consumes 1 byte, all storage device utilize a file system.

That means there is a difference between a files size and its size on the disk.
On disks the files are stored in so-called clusters which are a group of disk sectors. "Size on disk" refers to the amount of cluster(s) a file is taking up, while "file size" is an actual byte count of the file data. So you will usually find that the size on disk is larger than the file size. This is not an error, but a result of the disk organization via a file system. Since sector and cluster sizes vary depending on a disk's size and the used file system, the ratios between the size on disk and the file size also vary between the various storage devices.

### 5.2.9.3 Memory sizes

AC500-eCo V3 processor modules

| PLC type | system RAM disk | userdisk <br> PlcLogic <br> ... | Retain, \%M area | flash disk | memory card |
| :---: | :---: | :---: | :---: | :---: | :---: |
| PM5012-x- <br> ETH | Dynamically /max. 7.6 MB | 30 MB | 8 kB <br> Retain and persistent 4 kB (of which 88 byte are reserved for allocation table and are not available to the user) <br> \%M 4 kB | None | 4) Chapter 5.2.8.2.1 "MC5102 Micro memory card with adapter" on page 1309 |
| PM5032-x- <br> ETH |  |  | $32 \text { kB }$ <br> Retain and per- |  |  |
| PM5052-x- <br> ETH |  |  | sistent 16 kB (of which 88 byte are reserved for allocation table and are not available to the user) <br> \%M 16 kB |  |  |
| PM5072- <br> T-2ETH(W) <br> PM5082- <br> T-2ETH |  |  | $100 \text { kB }$ <br> Retain and persistent 36 kB (of which 88 byte are reserved for allocation table and are not available to the user) <br> \%M 64 kB |  |  |

AC500 V3 processor modules

| PLC type | system RAM disk | userdisk PlcLogic - : = | SRAM <br> Retain, \%M area | flash disk | memory card |
| :---: | :---: | :---: | :---: | :---: | :---: |
| $\begin{aligned} & \text { PM5630-2ET } \\ & \text { H } \end{aligned}$ | Dynamically /max. 7.6 MB | $\begin{aligned} & 40 \mathrm{MB} \\ & 30 \mathrm{MB} \text { (as of } \\ & \mathrm{V} 3.4 .0 \text { ) } \end{aligned}$ | 256 kB <br> Retain and persistent 128 kB (of which 68 byte are reserved for allocation table and are not available to the user) <br> \%M 128 kB | None | 4 Chapter 5.2.8.2.2 <br> "MC5141 - <br> Memory card" on page 1315 <br> 4. Chapter 5.2.8.2.1 <br> "MC5102 Micro memory card with adapter" on page 1309 |
| $\begin{aligned} & \text { PM5650-2ET } \\ & \text { H } \end{aligned}$ | Dynamically /max. 16 MB | $\begin{aligned} & 246 \mathrm{MB} \text { (as } \\ & \text { of V3.0.x) } \\ & 381 \mathrm{MB} \text { (as } \\ & \text { of V3.1) } \\ & 285.75 \text { (as of } \\ & \text { V3.4.0) } \end{aligned}$ |  |  |  |


| PLC type | system RAM disk | userdisk <br> PlcLogic | SRAM <br> Retain, \%M area | flash disk | memory card |
| :---: | :---: | :---: | :---: | :---: | :---: |
| $\begin{array}{\|l} \text { PM5670-2ET } \\ \mathrm{H} \end{array}$ | Dynamically /max. 69 MB | $\begin{array}{\|l\|} \hline 858 \mathrm{MB} \\ 643.50 \mathrm{MB} \\ \text { (as of V3.4.0) } \end{array}$ | 1536 MB <br> 1 MB retain and persistent (of which 68 byte are reserved for allocation table and are not available to the user) <br> 512 kB \%M |  |  |
| $\begin{array}{\|l} \hline \text { PM5675-2ET } \\ \mathrm{H} \end{array}$ |  |  |  | 8 GB |  |

It is not possible to use 100 \% of a device's memory space. About 10 \% of the total available space must remain unused at any time to maintain normal device operation.

### 5.2.9.4 Storage device details

### 5.2.9.4.1 SRAM

The SRAM is a battery-buffered, nonvolatile RAM and is used for the retain/persistent and the \%M variables. If a battery is inserted into the processor module, the data stored in the SRAM will not get lost during a power-down cycle.
During PLC startup, the SRAM will be deleted automatically if no or an empty battery is inserted into the processor module. In this case the information
ABBInitSram_SetupMemory : SRAM cleared
and the warning
Retain size in config changed, or retain area got corrupted
are written into the log file.
$\boldsymbol{y}$ Chapter 6.8.2.1 "Handling of remanent variables for AC500 V3 products" on page 4412

### 5.2.9.4.2 Flash disk

The flash disk is an internal persistent mass storage device and can be used for any application purpose.

It has a memory capacity of 8 GB (preformatted).
The flash disk is capable of high data throughput, however, the actual values to be achieved depend on the use cases. If the performance seems to get insufficient, check the following:

- If the PLCs CPU load is high, reduce overall CPU load of the PLC to have more performance for file operations.
- If the device has low free space, cleanup the disk.

Please consider the cluster size of 4 kB in your application design to achieve optimal usage of the flash disks space and access performance. For example, 10 files with 10 byte each require $10 * 4 \mathrm{kB}$ disk space, while 1 file with 100 byte requires only 4 kB .

Number of max. Technically, the flash chip used in the V3 flash disk has 20000 erase cycles (Write cycles). write cycles

Due to the produced write overhead, the optimum achievable number of write cycles is 10000 (for typical payload sizes of 256 kB ).

Example The write overhead is indicated by the write amplification factor (WAF).

$$
\text { WAF }=\frac{\text { Flash Write (in Bytes) }}{\text { Host Write (in Bytes) }}
$$

Table 299: Rule of thumb for assessing the flash lifetime for an application:

| Typical payload sizes | WAF | Max. write cycles |
| :---: | :---: | :---: |
| 256 kB | 2 | 10000 |
| 128 kB | 4 | 5000 |
| 64 kB | 8 | 2500 |
| $\ldots$ | $\ldots$ | $\ldots$ |
| 1024 Byte | 512 | $<40$ |
| 512 Byte | 1024 | $<20$ |

For monitoring the status

It is recommended to use the respective function blocks to monitor the status of the flash disk.
«y Chapter 6.8.2.2.1.4 "Health monitoring" on page 4424
Since FW version 3.3.0, there is also a diagnosis event supported when the user flash memory reaches the end of its life cycle.

Lifetime of flash disk will also depend on the operating environment.
E.g. high ambient temperatures will impose stress on the user flash memory and reduce the total overwrites achievable.

- Max. write speed is $20 \mathrm{MB} / \mathrm{s}$ (continuous write of sequential data)
- Read cycles are unlimited.


### 5.2.9.4.3 Memory card

The memory card is a removable persistent mass storage device and can be used for any application purpose. Both firmware updates and boot project updates can be run from the memory card.
\& Chapter 6.3.1.4.5 "Firmware installation and update with memory card" on page 1426

| Size | Product specific <br> 乡 Chapter 5.2.9.3 "Memory sizes " <br> on page 1335 |
| :--- | :--- |

### 5.2.9.4.4 Further information

$\leadsto$ Further details on the storage device sizes
*) Further details on the FTP access
«) Further details on the PLC shell commands

### 5.3 Status LEDs, display and control elements

Depending on the device type, various operating elements provided on the front panel can be used to control the devices of the PLC system and/or to change the operating mode.

Operating elements:

- Status LEDs:

Indicates the availability of devices and components such as communication modules, communication interface modules or function modules. Functionality and diagnosis of the status LEDs depends on the specific module and is described in the device description of the appropriate module. Possible status: on/off/blinking

- I/O LEDs:

Displays the status of the inputs and outputs.

- Display:

Available for some processor modules. It can be used for simple configurations and for reading out diagnosis information.
« Chapter 6.8.2.5 "Display, LEDs and function keys on the front panel" on page 4452
๕ Chapter 6.9.1.3 "Diagnosis in CPU display" on page 4607
(y) Chapter 4.8 "Troubleshooting" on page 149

- Function keys and switches:

Allows to change the current operating modes/status manually ${ }^{4} \Rightarrow$ Chapter 6.8.2.5.5 "Description of the function keys" on page 4457.

## 6 Configuration and programming

### 6.1 Cyber security

### 6.1.1 General

Cyber security This product is designed to be connected to and to communicate information and data via a disclaimer network interface. It is your sole responsibility to provide and continuously ensure a secure connection between the product and your network or any other network (as the case may be). You shall establish and maintain any appropriate measures (such as but not limited to the installation of firewalls, application of authentication measures, encryption of data, installation of anti-virus programs, etc.) to protect the product, the network, its system and the interface against any kind of security breaches, unauthorized access, interference, intrusion, leakage and/or theft of data or information. ABB Ltd and its affiliates are not liable for damages and/or losses related to such security breaches, any unauthorized access, interference, intrusion, leakage and/or theft of data or information.

Although ABB provides functionality testing on the products and updates that we release, you should institute your own testing program for any product updates or other major system updates (to include but not limited to code changes, configuration file changes, third party software updates or patches, hardware exchanges, etc.) to ensure that the security measures that you have implemented have not been compromised and system functionality in your environment is as expected. This also applies to the operating system. Security measures (such as but not limited to the installation of latest patches, installation of firewalls, application of authentication measures, installation of anti-virus programs, etc.) are in your responsibility. You have to be aware that operating systems provide a considerable number of open ports that should be monitored carefully for any threats.

It has to be considered that online connections to any devices are not secured. It is your responsibility to assure that connections are established to the correct device (and e.g. not to an unknown device pretending to be a known device type). Furthermore you have to take care that confidential data exchanged with the PLC is either compiled or encrypted.

Security related Security details for industrial automation is provided on ABB website in a whitepaper. deployment guidelines for industrial automation
Signed firmware The firmware update files for the AC500 V3 PLC are digitally signed releases by ABB. During updates the update process, these signatures are validated by a hardware security component in the PLC. This way, the AC500 V3 PLC will only update with valid, authentic firmware, signed by ABB.

Open ports and As part of the ABB security concept the AC500 V3 PLC comes with minimal services opened by services default. Only the services needed for initial setup and programming are open before any user application is downloaded $\stackrel{y}{c}$ Chapter 5.2.1.2.1.3 "Ethernet protocols and ports for AC500 V3 products" on page 307.

Only used services/ports should be enabled (e.g. to enable the functionality of an FTPS server).

| Encrypted and | An application can be encrypted and signed in order to protect a running application in an |
| :--- | :--- |
| signed applica- | AC500 V3 PLC and to protect a configured project. How to set-up the user management, the |
| tions | communication and the boot application in order to prevent unauthorized access is explained in |
| the application note $A C 500$ V3-Encrypt and sign your application. |  |

Secure commu- Whenever possible, use an encrypted communication between AC500 V3 devices and third nication party devices, such as HMI devices. This is necessary to protect passwords and other data.

## Secure shell access for ABB service

The AC500 V3 PLC contains a secure shell service to access core logging data in case of problems which need a deeper analysis. This service is inactive by default, which means that no one can access this privileged shell in the normal operating state.

To activate this service, local access to the PLC is necessary and activation is only valid until the next power cycle of the PLC. Once activated, the service run on TCP port 22. Each PLC also protects the secure shell access by an individual password.

Frequently For more information around cyber security please see our AC500 cyber security FAQ. asked questions

### 6.1.2 Defense in depth

The defense in depth approach implements multi-layer IT security measures. Each layer provides its special security measures. All deployed security mechanisms in the system must be updated regularly. It is also important to follow the system vendor's recommendations on how to configure and use these mechanisms. As a basis, the components must include security functions such as:

- Virus protection
- Firewall protection
- Strong and regularly changed passwords
- User management
- Using VPN tunnels for connections between networks

Additional security components such as routers and switches with integrated firewalls should be available. A defined user and rights concept managing access to the controllers and their networks is mandatory. Finally, the manufacturer of the components should be able to quickly discover weaknesses and provide patches.

Only used services/ports should be enabled (e.g. to enable the functionality of an FTPS server).

## References: CODESYS Security Whitepaper

Security zones IT resources vary in the extent to which they can be trusted. A common security architecture is therefore based on a layered approach that uses zones of trust to provide increasing levels of security according to increasing security needs. Less-trusted zones contain more-trusted zones and connections between the zones are only possible through secure interconnections such as firewalls Fig. 265. All resources in the same zone must have the same minimum level of trust. The inner layers, where communication interaction needs to flow freely between nodes, must have the highest level of trust. This is the approach described in the IEC 62443 series of standards.

Firewalls, gateways, and proxies are used to control network traffic between zones of different security levels, and to filter out any undesirable or dangerous material. Traffic that is allowed to pass between zones should be limited to what is absolutely necessary because each type of service call or information exchange translates into a possible route that an intruder may be able to exploit. Different types of services represent different risks. Internet access, incoming e-mail and instant messaging, for example, represent very high risks.

Cooperate network
Available for all employees

## Company intranet

Available to local employees


Firewall

> Automation system
> Available to operators and process engineers only

Firewall

## Fig. 265: Security zones

Fig. 265 shows three safety zones, but the number of zones is not necessarily decisive, there can also be more or less - depending on the safety requirement. The use of multiple zones allows access between zones of different trust levels to be controlled to protect a trusted resource from attack by a less trusted one.

High-security zones should be kept small and independent. They need to be physically protected, i.e. physical access to computers, network equipment and network cables must be limited by physical means to authorized persons only. A high-security zone should obviously not depend on resources in a less secure zone for its security. Therefore, it should form its own domain that is administered from the inside, and not depend on, e.g., a domain controller in a less secure network.

Even if a network zone is regarded as trusted, an attack is still possible: by a user or compromised resource that is inside the trusted zone, or by an outside user or resource that succeeds to penetrate the secure interconnection. Trust therefore depends also upon the types of measures taken to detect and prevent compromise of resources and violation of the security policy.
References: Security for industrial automation and control systems

### 6.1.3 Secure operation

The controller must be located in a protected environment in order to avoid accidental or intended access to the controller or the application.
A protected environment can be:

- Locked control cabinets without connection from outside
- No direct internet connection
- Use firewalls and VPN to separate different networks
- Separate different production areas with different access controls

To increase security, physical access protection measures such as fences, turnstiles, cameras or card readers can be added.

Follow these rules for the protected environment:

- Keep the trusted network as small as possible and independent from other networks.
- Protect the cross-communication of controllers and the communication between controllers and field devices via standard communication protocols (fieldbus systems) using appropriate measures.
- Protect such networks from unauthorized physical access.
- Use fieldbus systems only in protected environments. They are not protected by additional measures, such as encryption. Open physical or data access to fieldbus systems and their components is a serious security risk.
- Physically protect all equipment, i.e., ensure that physical access to computers, network equipment and cables, controllers, I/O systems, power supplies, etc., is limited to authorized persons
- When connecting a trusted network zone to outer networks, make sure that all connections are through properly configured secure interconnections only, such as a firewall or a system of firewalls, which is configured for "deny by default", i.e., blocks everything except traffic that is explicitly needed to fulfill operational requirements.
- Allow only authorized users to log on to the system, and enforce strong passwords that are changed regularly.
\& Chapter 6.3.1.2.1.1 "General" on page 1409
- Continuously maintain the definitions of authorized users, user groups, and access rights, to properly reflect the current authorities and responsibilities of all individuals at all times. Users should not have more privileges than they need to do their job.
- Do not use the system for e-mail, instant messaging, or internet browsing. Use separate computers and networks for these functions if they are needed.
- Do not allow installation of any unauthorized software in the system.
- Restrict temporary connection of portable computers, USB memory sticks and other removable data carriers. Computers that can be physically accessed by regular users should have ports for removable data carriers disabled.
- If portable computers need to be connected, e.g., for service or maintenance purposes, they should be carefully scanned for viruses immediately before connection.
- All CDs, DVDs, USB memory sticks and other removable data carriers, and files with software or software updates, should also be checked for viruses before being introduced into the trusted zone.
- Continuously monitor the system for intrusion attempts.
- Define and maintain plans for incident response, including how to recover from potential disasters.
- Regularly review the organization as well as technical systems and installations with respect to compliance with security policies, procedures and practices.
A protected local control cabinet could look like in figure 266, page 1343. This network is not connected to any external network. Security is primarily a matter of physically protecting the automation system and preventing unauthorized users from accessing the system and from connecting or installing unauthorized hardware and software.


Fig. 266: Isolated automation system
Servers and workplaces that are not directly involved in the control and monitoring of the process should preferably be connected to a subnet that is separated from the automation system network by means of a router/firewall. This makes it possible to better control the network load and to limit access to certain servers on the automation system network. Note that servers and workplaces on this subnet are part of the trusted zone and thus need to be subject to the same security precautions as the nodes on the automation system network.


Fig. 267: Plant information network connected to an automation system
For the purposes of process control security, a general-purpose information system (IS) network should not be considered a trusted network, not the least since such networks are normally further connected to the Internet or other external networks. The IS network is therefore a different lower-security zone, and it should be separated from the automation system by means of a firewall. The IS and automation system networks should form separate domains.


Fig. 268: Automation system and IS network

### 6.1.4 Hardening

System hardening means to eliminate as many security risks as possible. Hardening your system is an important step to protect your personal data and information. This process intends to eliminate attacks by patching vulnerabilities and turning off inessential services. Hardening a system involves several steps to form layers of protection.
Commissioning phase

- Protect the hardware from unauthorized access
- Be sure the hardware is based on a secure environment
- Disable unused software and services (network ports)
- Install firewalls
- Disallow file sharing among programs
- Install virus and spyware protection
- Use containers or virtual machines
- Create strong passwords by applying a strong password policy
- Create and keep backups
- Use encryption when possible
- Disable weak encryption algorithms
- Separate data and programs
- Enable and use disk quotas
- Strong logical access control
- Adjust default settings, especially passwords

Verification phase

- Verification of antivirus - Check antivirus is active and updated
- Verification of the identification - Check that test and unauthorized accounts are removed
- Verification of intrusion detection systems - Check malicious traffic is blocked
- Verification of audit logging - Check audit log is enabled
- You can use the checklist out of the cyber security white paper

Operation phase

- Keep software up-to-date, especially by applying security patches
- Keep antivirus up and running
- Keep antivirus definitions up-to-date
- Delete unused user accounts
- Lock an active session whenever it is unattended, e.g., lock the screen of the PC or of the control panel (HMI)
Decommissioning phase
- Delete all credentials stored in the device like certificates and user data $\xi^{*}$ Chapter 4.10 "Decommissioning" on page 157.
References: Hardening in Wikipedia (2021)


### 6.1.5 Certificates factory default - no encryption

As of Automation Builder Release 2.6.0, the encryption of the AC500 communication policy is set to "No encryption" by default. The reason for that is that the PLC clock defaults to 01/01/1970 when shipped from the factory, any pre-installed certificate would already be expired when the user attempts to connect.

## Preliminary

 work$\checkmark$ Connection of the PLC to the Automation Builder and internet access.
$\checkmark$ The PLC must be equipped with a battery to keep the date in case of power off. (4) Chapter 5.2.8.2.3 "TA521-Battery" on page 1319
$\checkmark$ The PLC must be set to the current date and time. ${ }^{\wedge} \Rightarrow$ Chapter 6.3.4.9.3.5.6 "Time syncronisation" on page 1746

1. If necessary reboot the PLC.
2. Connect the PLC to the Automation Builder again.

## Create an CAsigned certifi- <br> cate

The basis for an official trusted signed certificate is the creation of a Certificate Signing Request.
As an example, the creation of a certificate for a FTP server is described.

1. Double-click to the PLC main node and select the "PLC Shell" tab. Command line input is on the bottom. Find the index number for the FTP server and type [cert-getapplist] in the below command line.

2. Type [cert-createcsr <index number> encoding=Base64] in the command line e.g. here: [cert-createcsr 1 encoding=Base64] and press [ENTER].

For some users it may be necessary to swap "Base64" for "ASN1" if the user's PKI needs this format.
3. Change to the "Log" tab.
4. Click the $₫$ refresh button.
$\Rightarrow$ Wait until the log entry of CSR file created pops up.

|  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: |
| Communication Settings <br> PLC Settings | (\%) Components - |  |  |  |
|  | Severity | Time Stamp | Description | Component |
|  |  | 04.08.2022 09:45:27 |  | CmpopenssL |
| Version information | - | 04.08.2022 09:42:27 |  | CmpopenssL |
|  | - | 04.08.2022 07:44:56 | [1] CSR file created, subject='commonName=FTPServer@PM5670-2ETH unstucturedName=Vendor: ABB AG unstucturedName=Device:' | CmpopenssL |
| Statistic | - | 04.08.2022 07:44:51 | [5] CSR fie created, subjecte'commonName=PM5670-2ETH unstructuredName=Vendor: ABB AG unstucturedName=Device: AC500 PM5' | CmpOPenssL |
|  | - | 04.08.2022 07:43:55 | [4] CSR file created, subject='commonName=PM5670-2ETH' | CmpOPenssL |
| Files | - | 04.08.2022 07:43:45 | [3] CSR file created, subject='commonName=PM5670-2ETH unstructuredName=Vendor: ABB AG unstructuredName=Device: AC500 PMS' | CmpopenssL |
|  | - |  | [2] CSR file created, subject=commonName=MQTTClient@PM5670-2ETH' | CmpOPenssL |
| Log | - | 04.08.2022 07743:08 04.08.2022 07:22:28 | [1] CSR fie created, subject='commonName=FTPServer @PM5670-2ETH unstucturedName=Vendor: ABE AG unstucturedName=Device:' | CmpOPenssL |

5. Change to the "Files" tab.
6. Click the $\$$ refresh button.
7. Open the "cert" folder in the PLC.
8. Open the "export" folder.
9. Copy the certificate request file [1_IoDrvFTPServer.csr] to any folder on your PC.

10. This file, here [1_loDrvFTPServer.csr] must then be signed by your Public Key Infrastructure ( PKI ).

The signing process itself is outside the scope of this documentation. There are sufficient sources on the internet for building your own PKI (e.g. PKI tutorial) - or get in touch with your IT department.

Integrate a cer- After the signed certificate comes back from the PKI and is stored on the PC, it has to be tificate imported into the controller.

## Be sure to import into the "Own Certificates" folder.

1. Change to the "Security Screen".
2. Select tab "Devices", the folder "Own Certificates" and press the ${ }^{0}$ import button.

3. Select the certificate ([1_IoDrvFTPServer.cer]) to be imported in the opening file manager.
4. Press the "Open" button.
$\Rightarrow$ The new certificate will be shown as "loDrvFTPServer" in the information column. The previous self-signed certificate will lose its name and is therefore not valid anymore.

5. Check the validation and "Issued by" of the new certificate by activating the top PLC branch.
6. Download and reboot the PLC (repower or use "PLC Shell" and command [reboot].)

Encrypted and An application can be encrypted and signed in order to protect a running application in an signed applica- AC500 V3 PLC and to protect a configured project. How to set-up the user management, the tions

We recommend changing the communication policy accordingly so that only encrypted connections are allowed. communication and the boot application in order to prevent unauthorized access is explained in the application note AC500 V3 - Encrypt and sign your application.

## Enforced signing

As of Automation Builder 2.6 .0 there is also the possibility to activate the "Enforced signing" mode in the "Change Runtime Security Policy". Then the controller accepts ONLY signed downloads.
"Tab Communication Settings $\rightarrow$ Device $\rightarrow$ Change Runtime Security Policy... $\rightarrow$ Code Signing
$\rightarrow$ New policy $\rightarrow$ Enforced signing"


White Paper - More information about certificates and cyber security can be found in the white paper AC500 cyber security AC500 cyber security.

### 6.1.6 Open Ports and Services

### 6.1.6.1 Default open Ethernet ports of AC500 V3 and AC500-eCo V3 CPUs

| Protocol | Port |
| :--- | :--- |
| ABB NetConfig ${ }^{1}$ ) | UDP 24576 |
| Online access with driver 3S Tcp/lp BlkDrvTcp (no scan) | TCP 11740 |
| OPC UA server ${ }^{2}$ ) | TCP 4840 |
| Remarks: <br> 1): The port 24576 for ABB NetConfig protocol can be disabled via PLC configuration by <br> deleting the protocol node from configuration tree of Ethernet interfaces ETH1 and ETH2. <br> 2): The port 4840 for OPC UA server is closed by default as of SystemFW V3.1.0. |  |

### 6.1.6.2 Open ports and services of the Automation Builder (Engineering PC)

| Port | Protocol | Description |
| :--- | :--- | :--- |
| 1217 | TCP | CODESYS Gateway V3 |
| 1210 | TCP | CODESYS Gateway V2 |
| 1211 | TCP | CODESYS Gateway V2 |


| Port | Protocol | Description |
| :--- | :--- | :--- |
| 22350 | TCP/UDP | CodeMeter License Server <br> (runtime) - license |
| 22352 | HTTP | CodeMeter License Server <br> (runtime) - WebAdmin |
| 22353 | HTTPS | CodeMeter License Server <br> (runtime) - WebAdmin |
| 11040 | HTTP | Python editor server |

### 6.2 Engineering software Automation Builder

### 6.2.1 Introduction

ABB $\quad$ ABB Automation Builder is the integrated software suite for machine builders and system inte-

Automation
Builder - One
holistic suite grators wanting to automate their machines and systems in a productive way. Combining the tools required for configuring, programming, debugging and maintaining automation projects from a common intuitive interface, Automation Builder addresses the largest single cost element of most of today's industrial automation projects: software. ABB Automation Builder covers the engineering of ABB PLCs, Safety PLCs, control panels, drives, motion and robots.

## Automation

 Builder ReadMeBefore starting Automation Builder configuration read the version specific information provided in the Automation Builder readme file. It describes new features and functions as well as workarounds on known problems. The readme file is stored in the installation directory of Automation Builder, however can be downloaded as well from ABB website http://new.abb.com/plc/automationbuilder.

### 6.2.2 Automation Builder updates

An update notification dialog will be shown during Automation Builder startup in case there are any updates available for the installed version.

The user has the possibility to check for updates manually: "Help $\rightarrow$ Check for Updates"
${ }^{4}$ ) Chapter 4.8.3 "Execution of a hotfix" on page 151

### 6.2.3 Use of open source components

ABB Automation Builder and AC500 V3 firmware are using open source components. A detailed overview of these open source components and the corresponding licenses are available from Automation Builder via "Help $\rightarrow$ About... $\rightarrow$ License Terms".

There are separate lists available for Automation Builder and AC500 V3 firmware.

### 6.2.4 Managing your licenses

### 6.2.4.1 Identifying the installed license

The title bar of the Automation Builder shows the version and the license information (Basic, Standard or Premium).

Be aware of the following rule for this information:
The info in the title bar is taken in this order from the first found license.

- local licenses (on PC)
- on a dongle (USB key)
- network licenses

So if a local license is only basic and a dongle with premium license is inserted the functionality is premium (the highest available) but the information in the title bar is basic.

To check your installed licenses, the CodeMeter Control Center tool can be used.
${ }^{\star}{ }^{\star}$ Chapter 6.2.4.3 "Checking licenses with "CodeMeter Control Center"" on page 1352

### 6.2.4.2 Selecting the license used on Automation Builder startup

You can select, which license the Automation Builder should use on startup.

To select which license should be used:

1. In the Automation Builder menu select "Tools $\rightarrow$ Options".
$\Rightarrow$ The "Options" window is opened.
2. In "Startup settings" under "License" select which license should be used.

- Default: The most comprehensive available license will be selected
- Use only local license: Network licenses will never be selected
- Display licenses selection dialog if shared licenses are available: On every Automation Builder startup, you will be asked to select a license


3. To apply the setting select [OK].

### 6.2.4.3 Checking licenses with "CodeMeter Control Center"

1. Open the "CodeMeter Control Center" via the "Windows start menu $\rightarrow$ CodeMeter $\rightarrow$ CodeMeter Control Center".

CodeMeter Control Center
File Process View Help
License
Events

ABB Automation Builder Trial 128-8843150

ABB R\&D non-bind License
128-17149280

Name: ABB Automation Builder Trial

Serial: 128-8843150

Version: CmActLicense 1.19

Status: License activated

## Remove License

CodeMeter service is running.
2. In the "CodeMeter Control Center" window you can see the different license "tickets" / "CmContainers" that are installed on your PC.

To see more details, open the "CodeMeter WebAdmin" by selecting [WebAdmin]


### 6.2.4.4 Setting dedicated network servers in search list

In case of a new installation CodeMeter will check for licenses also in the network. If there is a run-out or wrongly installed license found, the service is closed without any further hint. This looks like Automation Builder starts and closes after a few moments.

To set the search for licenses to your local machine only follow these steps:

1. Open the CodeMeter Control Center $\stackrel{y}{4}$ Chapter 6.2.4.3 "Checking licenses with "CodeMeter Control Center"" on page 1352.
2. Open the CodeMeter WebAdmin by selecting [WebAdmin]

3. Select "Configuration $\rightarrow$ Basic $\rightarrow$ Server Search List"

4. Select "add new Server"

5. Restart the license check
6. Add more servers to the search list by entering the IP address or name of the license servers you know.

### 6.2.4.5 Restarting license check with a dongle bound license

In case of using a dongle bound license it might be necessary to restart the check for license on the PC, e.g. if the dongle was removed and reinserted.

1. Open the CodeMeter Control Center ${ }^{\star} \stackrel{y}{c}$ Chapter 6.2.4.3 "Checking licenses with "CodeMeter Control Center"" on page 1352.
2. Select "Process $\rightarrow$ Restart CodeMeter Service"

CodeMeter Control Center
File Process View Help


Name: ABB R\&D non-bind License

Serial: 128-17149280

Version: CmActLicense 1.19

Status: C License activated

License Update Remove License

CodeMeter service is running.

### 6.2.4.6 Removing trial license to remove expiring message

If an unlimited license is installed after having a trial license activated, the warning message for expiring date of the trial license still pops up at the Startup of the Automation Builder.


To avoid this message the trial license can be removed.

## CAUTION!

- If you remove a license from your PC it will be permanently lost.
- Be aware that the trial license includes all premium functionalities, which will not be available any more if your unlimited license is not a premium license, e.g. standard.

1. Check for the Trial CmContainer number in CodeMeter WebAdmin InterfaceCheck for the Trial CmContainer number in CodeMeter WebAdmin Interface
2. Search CmContainer number in CodeMeter Control Center

3. Remove this selected license in CodeMeter Control Center

### 6.2.4.7 Network licenses

### 6.2.4.7.1 General

It allows sharing of licenses between team members, easy switchover between several workstations with a single license and allows centralized administration (ordering, registration, activation).

The Automation Builder License Manager and CodeMeter need to be used to configure the network server.

- In a typical office LAN (Local Area Network) setup on Client side the default settings of the Automation Builder (and CodeMeter) are sufficient to get the Network Licenses working.
- If an opened Automation Builder is loosing contact to the network server (e.g. due to network problems) Automation Builder will prompt the user to restore the network. After 30 minutes without connection to the network server Automation Builder will fall back to basic edition. Opened editors for non-basic features stay open and usable. So your work will not be lost in case of troubles with the network.
«2 Information on how to borrow and return a network license.
The network lincenses can be used as of Automation Builder 1.2.0.


### 6.2.4.7.2 Setting up a network license

## General

The following setup works in typical environments.

## Configuring a network license server

$\checkmark$ Network license must be registered.
$\square$ Automation Builder license must be activated.

1. Launch CodeMeter WebAdmin as described in chapter 'Managing your licenses' \& Chapter 6.2.4 "Managing your licenses" on page 1350
2. Select "Configuration $\rightarrow$ Server $\rightarrow$ Server Access"

3. Enable Network Server

Keep the default port settings. These should work in most cases.
4. Select [Apply]

5. For the changes to take effect, restart CodeMeter Control Center ${ }^{〔}$ Chapter 6.2.4.5 "Restarting license check with a dongle bound license" on page 1355

## NOTICE!

- In case you want to control usage of network licenses please refer to chapter 'View network server license usage' «y Chapter 6.2.4.7.4 "View network server license usage" on page 1360
- Activation keys for network licenses are valid for one network license each. This one license can be shared among many people but only one Automation Builder instance at the same time. If you want to run more than one Automation Builder instance at the same time you have to activate more than one network license. This means you have to purchase and enter more than one activation key (one per license).


## Configuring the client side

The default settings of Automation Builder and the CodeMeter (on client side) are sufficient in most cases to get the network licenses working. In case of problems accessing the network license, please set the server search list on the client side.


## -Server Search List

1. Automatic server search ( 255.255 .255 .255 )
```
add new Server
```


### 6.2.4.7.3 View network server licenses

On the Network Server side you can find information on existing network licenses and their current allocation.

1. Launch CodeMeter WebAdmin ${ }^{4}$, Chapter 6.2.4.3 "Checking licenses with "CodeMeter Control Center'"' on page 1352.
2. Select "License Monitoring $\rightarrow$ All Licenses"

6.2.4.7.4 View network server license usage
3. Launch CodeMeter WebAdmin ${ }^{\mu}$ Chapter 6.2.4.3 "Checking licenses with "CodeMeter Control Center'"' on page 1352.
4. Select "License Monitoring $\rightarrow$ Sessions"


Sessions

| Client | CmContainer | Firm Item | Product Item | Access Mode |
| :---: | :---: | :---: | :---: | :---: |
| localhost | ABB R\&D non-bind License (128-17149280) | 5000460 : ABB Automation Products GmbH | 12000 : Virtual Commissioning | Station Share |
| localhost | ABB R\&D non-bind License (128-17149280) | 5000460 : ABB Automation Products GmbH | 12000 : Virtual Commissioning | Station Share |
| localhost | ABB R\&D non-bind License (128-17149280) | 5000460 : ABB Automation Products GmbH | 2013 : Open Device Type Editor | Station Share |
| localhost | ABB R\&D non-bind License (128-17149280) | 5000460 : ABB Automation Products GmbH | 2010 : Advanced IO device handling | Station Share |
| localhost | ABB R\&D non-bind License (128-17149280) | 5000460 : ABB Automation Products GmbH | 1008 : Automation Builder Premium 2.x | Station Share |
| localhost | ABB R\&D non-bind License (128-17149280) | 5000460 : ABB Automation Products GmbH | 1008 : Automation Builder Premium 2.x | Station Share |
| localhost | ABB R\&D non-bind License (128-17149280) | 5000460 : ABB Automation Products GmbH | 1005 : Project Scripting | Station Share |
| localhost | ABB R\&D non-bind License (128-17149280) | 5000460 : ABB Automation Products GmbH | 1004 : Project Compare | Station Share |
| localhost | ABB R\&D non-bind License (128-17149280) | 5000460 : ABB Automation Products GmbH | 1004 : Project Compare | Station Share |
| localhost | ABB R\&D non-bind License (128-17149280) | 5000460 : ABB Automation Products GmbH | 1002 : ECAD Import Export | Station Share |

### 6.2.4.7.5 Controlling network server license usage

On the Network Server side you can define settings managing the client access to CodeMeter License Server on a network.

1. Launch CodeMeter WebAdmin $\triangleq$ Chapter 6.2.4.3 "Checking licenses with "CodeMeter Control Center"'" on page 1352.
2. Select "Configuration $\rightarrow$ Server $\rightarrow$ Server Access"

3. Add entries of PCs you want to share licenses with by adding client computers and IP addresses for accessing CodeMeter License Server on a network.


### 6.2.4.8 License borrowing manager

### 6.2.4.8.1 Introduction

The license borrowing manager allows you to borrow a network license for offline use and return it.


The license borrowing manager is supported as of Automation Builder 2.2.3.

### 6.2.4.8.2 Borrowing a network license

$\checkmark$ Network access to the license server required.
$\checkmark$ Opened the license borrowing manager.

1. Select the license you want to borrow.

2. Select [Borrow License].
3. Select the target CmContainer.

Alternatively a new CmContainer can be created.
4. Select the end of the borrowing period.

| Choose Transfer Options |  |
| :--- | :--- |
| Choose Target CmContainer |  |
| CmContainer Server <br> Create new CmContainer (2000) localhost <br>   <br>  Cnd of borrowing period <br> $5 / 12 / 2020$ Longest borrowing period until <br> Ok Cancel |  |

5. Select [OK].
$\Rightarrow$ The license has sucessfully been borrowed.
The list of available licenses has been updated.


### 6.2.4.8.3 Returning a network license

Automatical Network licenses will be returned automatically after the expiration of the maximum borrwoing return of a license period. No licenses server access is required.


Manual return of Network licenses can be returned anytime manually.

## a network

license
$\square \quad$ Network access to the license server required.
$\checkmark$ Opened the license borrowing manager.

1. Select a borrowed license.

2. Select [Return License]
$\Rightarrow$ The license has sucessfully been returned.

### 6.2.4.9 Transfering an Automation Builder license

### 6.2.4.9.1 General

It is possible to transfer normal licenses from a PC to another PC or dongle (DM-Key).
This is not possible for ABB internal or temporary licenses, e.g. the 30 day Trial license.
The process consists of two main steps:

1. Return the actual license from the actual PC
2. Reactivate the license on the new PC

### 6.2.4.9.2 Getting activation code

For all license transfer processes the activation code is required. It is available from the license paper from purchasing the license.
For Automation Builder licenses purchased April 2020 or later, the activation code is available from the activated license:

1. Open CodeMeter Control Center and navigate to the [WebAdmin].

2. Identify the right product code.

Automation Builder editions consist of multiple product codes. The activation ID is available from the product code containing the edition name, e.g. "Automation Builder Standard".

3. Select product code to access the product code details. Under "License Information" you can find the activation code.


### 6.2.4.9.3 Returning an Automation Builder license

$\square \quad$ You need the License Activation code of the license you want to return.

1. Go to the following website: http:///c.codemeter.com/32838/depot-return/index.php

The website is also availaible through the Automation Builder menu under "Help $\rightarrow$ Return of Automation Builder license".
2. Insert your Activation code in the field "Ticket"
3. Select [Next]

## Welcome to CodeMeter License Central WebDepot

Wekome to Codemeter License Central WebDepot You can transfer your licenses to your CmContainer using this webDepot. Please enter your ticket and click "Next

## ncket



Next

Legai Notice / O WIBU-SYSTEMS AG | Codemeter Lucense Central WebDepot v19.05 205.500. ws | 2021-05-28 12.43 .46 (UTC)
4. Select [Re-Host License]

| Home | My Licenses |  |  | English | $\checkmark$ | 415 |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| My Licenses |  |  |  |  |  |  |
|  | Name | Activated On | CmContainer | Status |  |  |
|  | DM200-TOOL - Standard license | 2021-05-28 14:32:30 | - 128-17695656 | Available: 0 (1) |  |  |
|  | Re-Host Licenses |  |  |  |  |  |
| Legal Notice \| © WIBU-SYSTEMS AG | CodeMeter License Central WebDepot v19.05.205.500.ws | 2021-05-28 12:45:32 (UTC) |  |  |  |  |  |  |

$\Rightarrow$ If the CmContainer is found, continue with Online licenses transfer ${ }^{\star} \Rightarrow$ Chapter 6.2.4.9.4 "Online license transfer" on page 1367
$\Rightarrow$ If the CmContainer is not found, continue with Offline license transfer $\Leftrightarrow$ Chapter 6.2.4.9.5 "Offline license transfer" on page 1369

### 6.2.4.9.4 Online license transfer

$\triangleright \quad$ Wait till the CmContainer is found, then select [Deactivate Selected License Now]


[^27]
## Online License Transfer

Starting license transfer.
Creating license request.
Downloading license update.
Importing license update to CmContainer.
Creating receipt.
Uploading receipt.

## License transfer completed successfully!

OK

### 6.2.4.9.5 Offline license transfer

$\triangleright \quad$ If the CmContainer is not found on this PC, select "file-based license transfer" workflow.


Legal Notice | © WIBU-SYSTEMS AG | CodeMeter License Central WebDepot v19.05.205.500.ws | 2021-05-28 12:59:23 (UTC)
$\Rightarrow$ The following dialog opens
Home My Licenses English $\sim$ 保

Re-Hostable Licenses

```
Upload Request
Download Update

To re-host licenses from one CmContainer to another CmContainer via file transfer: - First step "Upload Request":
1. Create a license request file from the CmContainer with Serial 128-17695656 and Firm Code 5000460. This file can for example be created with CodeMeter Control Center. How it works +
2. Select the licenses you want to re-host.
3. Select the created license request file.
4. Click "Upload Request And Continue Now".
\begin{tabular}{ll|l|l|l}
\hline Name & Activated On & CmContainer & Status \\
\hline\(\square\) & DM200-TOOL - Standard license & \(2021-05-28\) & \(14: 32: 30\) & \(128-17695656\)
\end{tabular}

Sick license request file ( \({ }^{*}\). WibuCmRaC)
Choose File No file chosen

Upload Request And Continue Now
```

My License

```

Legal Notice |© WIBU-SYSTEMS AG | CodeMeter License Central WebDepot v19.05.205.500.ws | 2021-05-28 13:02:09 (UTC)

The instructions will lead you through the main steps of the offline license transfer:
1. On the offline PC open the CodeMeter Control Center.
2. Select [License Update].

\(\Rightarrow\) The CmFAS Assistant opens.
? \(\times\)

\section*{\(\leftarrow\) C. CmFAS Assistant}

\section*{Welcome to the CmFAS Assistant!}

The CodeMeter Field Activation Service (CmFAS) assistant helps you adding, changing and deleting licenses from the license management system CodeMeter.

With the CmFAS assistant you can create license request files, which you can send to the vendor of the software by email. You can also import the received license update files with the CmFAS assistant into the license management and create a receipt of the import for the vendor.
3. Select "Create a license request file".
\(\leftarrow\) C) CmFAS Assistant

\section*{Please select the desired action}
( Create license request
Choose this option if you want to create a license request file in order to send it to the vendor of the software.Import license update
Choose this option, if you received a license update file from the software vendor and want to import this file.Create receipt
Choose this option if you want to confirm the successful import of a license update file for the software vendor.
4. Select a location to store the license request file.
5. Transfer the license request file from the offline PC to an online PC.
6. On the online PC choose the license request file and select [Upload Request And Continue Now].

\(\Rightarrow\) The next dialog is opened
\(\qquad\)

Download License Update File


\footnotetext{
4. My Licenses
}

Legal Notice | © WIBU-SYSTEMS AG | CodeMeter License Central WebDepot v19.05.205.500.ws | 2021-05-28 13:32:49 (UTC)

\section*{7. Select [Download License Update File Now].}

8. Save the license update file to a location on your computer.
9. Transfer the license upate file from the online PC to the offline PC.
10. On the offline PC open the CmFAS Assistant.
11. Select "Import license update".
? \(\times\)
\(\leftarrow\) C CmFAS Assistant

Please select the desired actionCreate license request
Choose this option if you want to create a license request file in order to send it to the vendor of the software.Import license update
Choose this option, if you received a license update file from the software vendor and want to import this file.Create receipt
Choose this option if you want to confirm the successful import of a license update file for the software vendor.
12. Select the license update file, to import the new license to the offline PC
13. To confirm a succesful license transfer return to the online PC and select [Next].


Legal Notice |@ WIBU-SYSTEMS AG | CodeMeter License Central WebDepot v19.05.205.500.ws | 2021-05-28 13:32:49 (UTC)
\(\Rightarrow\) The last dialog is opened


Legal Notice | © WIBU-SYSTEMS AG | CodeMeter License Central WebDepot v19.05.205.500.ws | 2021-05-28 13:45:37 (UTC)
14. On the offline PC open the CmFAS Assistant.
15. Select "Create receipt".

\section*{\(\leftarrow\) © CmFAS Assistant}

\section*{Please select the desired action}Create license request
Choose this option if you want to create a license request file in order to send it to the vendor of the software.Import license update
Choose this option, if you received a license update file from the software vendor and want to import this file.
( Create receipt
Choose this option if you want to confirm the successful import of a license update file for the software vendor.
16. Choose a location to save the license receipt file,
17. Transfer the license receipt file from the offline PC to the online PC.
18. On the online PC choose the license receipt file and select [Upload Receipt Now].


Legal Notice | © WIBU-SYSTEMS AG | CodeMeter License Central WebDepot v19.05.205.500.ws | 2021-05-28 13:45:37 (UTC)
\(\Rightarrow\) After a succesful license transfer you will receive the following message
```

Home My Licenses

License Transfer Successfully Completed


Legal Notice | © WIBU-SYSTEMS AG | CodeMeter License Central WebDepot v19.05.205.500.ws | 2021-05-28 13:49:17 (UTC)

### 6.2.4.9.6 Return of unconfirmed license

If customers do not complete the offline activation process by uploading the receipt file, they won't be able to return the license. However, working with Automation Builder is possible.
When the customer tries to return the license, they will see that the license is available, but they won't be able to rehost the license because the button is not visible.
The only way to proceed here is to either create a receipt file from "CodeMeter" and upload it, or contact ABB Support who will manually confirm the activation after reviewing the "CmDust".

Below are the steps to enable the "Re-Host" button again.

Initial situation: No [Re-Host Licenses] button is visible in the "CodeMeter License Central WebDepot".


1. Open the "CodeMeter Control Center" on the PC.
2. Select the correct container under the "License" tab so that it is grayed out.
3. Click the [License Update] button.

$\Rightarrow$ The "CmFAS Assistant" opens.
$\leftarrow$ © CmFAS Assistant

## Welcome to the CmFAS Assistant!

The CodeMeter Field Activation Service (CmFAS) assistant helps you adding, changing and deleting licenses from the license management system CodeMeter.

With the CmFAS assistant you can create license request files, which you can send to the vendor of the software by email. You can also import the received license update files with the CmFAS assistant into the license management and create a receipt of the import for the vendor.
4. Click the [Next >] button.
5. In the window which appears, select the option [Create receipt].

## ? $\times$

$\leftarrow$ C CmFAS Assistant

## Please select the desired action

## Create license request

Choose this option if you want to create a license request file in order to send it to the vendor of the software.Import license update
Choose this option, if you received a license update file from the software vendor and want to import this file.

- Create receipt

Choose this option if you want to confirm the successful import of a license update file for the software vendor.
6. Click the [Next >] button.
7. Select a file name for storing the receipt file.
$\leftarrow$ C CmFAS Assistant

## Please select the file name

## C: \Users\DEKAKUR\128-and. WibuCmRaC

Select a file name for storing the receipt file. Then click on 'Commit' to create the file. You can then send this file to the vendor by email.
8. Click the [Commit] button.
$\leftarrow$ © CmFAS Assistant

The license receipt file has been successfully created.
The license receipt file has been successfully created.
You can send it now to the vendor of the software by email.
9. Click the [Finish] button.
10. Open the CodeMeter License Central WebDepot
11. Click on the [Upload activation receipt] button.
HOME
Automation Builder - Offline Activation
Upload activation request
Activating your licenses offline - First step "Upload activation request":

1. Create an activation request file with the Automation Builder Activation Wizard.
2. Pick the created activation request file.
3. Click "Upload request and continue".
Pick activation request file (*.xml)
Choose File No file chosen activation receipt
Upload request and continue
4. Click the [Choose File] button and select the receipt file that has just been created by "CodeMeter". A new window will open.

5. Click the [Upload activation receipt] button and wait for the success message.

6. Go again to the return WebDepot and try to return license. Now the [Re-Host Licenses] button is visible.

$$
\text { Home My Licenses } \quad \text { English } \sim \text { 是 }
$$

My Licenses

| Name | Activated On | CmContainer | Status |
| :--- | :--- | :--- | :--- |
| DM200-TOOL-Standard license | $2023-07-1717: 35: 46$ | $\bullet 128-17 n$ | Available: $0(1)$ |
|  |  |  |  |

## Re-Host Licenses

6.2.4.10 Generating license information file for support

To create a license information file which includes all license information for the support:

1. Select "Windows start menu $\rightarrow$ CodeMeter $\rightarrow$ Tools $\rightarrow$ CmDust".
$\Rightarrow$ The explorer window opens and shows the folder where the created log file CmDustResult.log is stored.
2. Please attach this file to any support request regarding your licenses.

### 6.2.4.11 Generating log files for support

Sometimes more detailed log files are needed to analyse a situation.
Then please also zip the following folder and attach it to your support request.
C:IProgramDatalCodeMeterlLogs
This folder includes

- CmActDiagLogyyyy-mm-dd-hhmmss.log
- CodeMeteryyyy-mm-dd-hhmmss.log

To make it easier to distinguish when the files were created, they are named as follows:

- yyyy - year, mm - month, hh - hour; mm - minutes, ss - seconds.


### 6.2.4.12 Further information

For more information about the installation and activation of the Automation Builder see: Automation Builder Installation and Activation

### 6.2.5 Setting up of communication parameters in Windows

Setting up of communication parameters

To set up the communication between the PC and the PLC, e.g., for downloading the compiled program, you have to set up the communication parameters.

The IP address of your PC must be in the same class as the IP address of the CPU.
The factory setting of the IP address of the CPU is 192.168.0.10.
The IP address of your PC should be 192.168.0.X. Avoid X = 10 in order to prevent an IP conflict with the CPU.

Subnet mask should be 255.255.255.0.

## Changing of the IP address

1. Open Windows control panel. Click "Network and Internet $\rightarrow$ Network and Sharing Center".
2. Click "Change adapter settings".

3. Right-click "Local Area Connection (Ethernet)" and select [Properties].

4. Double-click "Internet Protocol Version 4 (TCP/IPv4)".

5. Enter your desired IP address and subnet mask.

### 6.2.6 Version information

Information on the installed Automation Builder version such as installed packages or license terms can be found in menu "Help $\rightarrow$ About".


Package version information
Plug-in version information Safety version information

User registration data License Terms
Create package for support

Information about all installed software packages is shown.
Information about all installed plug-ins is shown.
Information about all safety components is shown. This button and the safety version are only visible if safety option is installed.
Enter or change your registration data.
Information about the license terms.
Creates a package which can be saved or sent to support ${ }^{4}$ Chapter 6.2.8 "Create log files for support" on page 1385.

It is possible to either continue working with a project on an older Automation Builder version or to update a project to the latest Automation Builder version.

### 6.2.7 PLC runtime and demo licensing

The use of some libraries and devices require the PLC to have a runtime license. Further it is possible to try out device features or library features by using a demo license ${ }^{*} \Rightarrow$ Chapter 6.3.2.2.2 "PLC runtime licensing" on page 1446.

### 6.2.8 Create log files for support

If a problem persists, contact ABB technical support. In order to be able to solve your problem as soon as possible, please provide the affected Automation Builder project or Panel Builder project, the log files and the firmware versions. How to read out log files is described in the application note How to get log information.

### 6.2.9 Menues, views, windows

### 6.2.9.1 Display settings

> Ensure the full display of Automation Builder editors by choosing the option "Smaller - 100 \% (default)" in "Start $\rightarrow$ Control Panel $\rightarrow$ Appearance and Personalization $\rightarrow$ Display".


### 6.2.9.2 Start page and menus

After start-up of Automation Builder software the start page is displayed.


All items of the Automation Builder user interface are described in the CODESYS documentation:

- ${ }^{\#}$ Chapter 6.4.1.21.3 "Menu Commands" on page 2551
- \& Chapter 6.4.1.21.2 "Objects" on page 2409
- \& Chapter 6.4.1.21.4 "Dialogs" on page 2745


### 6.2.9.3 "All Messages" window

Errors, warning and success messages are written to the "All messages" window:

| All messages | - 00 error(s) | (1) 0 warning(s) | (3) 2 message(s) | $x$ |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Description |  |  |  | Project | Object |
| (6) Parameters read successfully |  |  |  | Project1 | CI522_MODTCP [Modbus_devices] |
| (3) Parameters stored successfully |  |  |  | Project1 | C1522_MODTCP [Modbus_devices] |

*) Chapter 6.4.1.21.3.4.5 "Command 'Messages"' on page 2583

### 6.2.10 Device repository

The Device Repository of Automation Builder manages the pool of devices that can be used in the PLC configuration.
You install or uninstall devices in the "Device Repository" dialog box. The system installs a device by reading the device description files, which define the device properties for configurability, programmability, and possible connections to other devices.
You can use the devices provided in the device repository by adding them to the device tree of your project.

## Dialog device

 repository1. Click "Tools $\rightarrow$ Device Repository".
$\Rightarrow$ The "Device Repository" dialog box opens.

[Edit Locations]: Changes the default repository location. The devices can be managed at different locations.
[Install] / [Uninstall]: Installs or uninstalls devices.
[Renew device repository]: Updates the device list, e.g. after uninstallation of a device.
[Details]: Provides technical details on the selected device.
2. Select the install location. "System Repository" is set by default.

Installing devices

The device repository cannot be changed manually, e.g. by copying or deleting files. Use always the "Device Repository" dialog to add or remove devices.

1. Click [Install] and select the appropriate file format.
$\Rightarrow$ The "Install Device Description" dialog box opens.

2. Select the file path of the device description.
3. Select the file type filter of the required device description.
$\Rightarrow$ All device descriptions of the selected file type are listed.
4. Select the required device description and click [Open].
$\Rightarrow$ Automation Builder adds the device description to the matching category of your device repository.
If errors occur during installation (for example, missing files that are referenced by the device description), then Automation Builder displays them in the lower part of the "Device Repository" dialog box.

During the installation the device description files and all additional files referenced by that description will be copied to an internal location. Altering the original files will have no further effects to an internal location.

The changes take only effect after reinstalling the corresponding device(s).
The version number shown in the information section of the device should be verified.

## Uninstalling Select the device you want to remove and click [Uninstall]. devices The device is removed from the list.

Uninstalled devices which are used in existing projects are indicated by the symbol ?. The device will not be configured properly.

### 6.2.11 Creating and configuring projects

What is a - A project contains the objects which are necessary to create a controller program ("applicaproject? tion"):

- Pure POUs, for example programs, function blocks, functions, and GVLs.
- Objects that are also required to be able to run the application on a PLC. For example, task configuration, Library Manager, symbol configuration, device configuration, visualizations, and external files.
- In a project, you can program multiple applications and connect multiple controller devices.
- CODESYS manages device-specific and application-specific POUs in the "Devices" view ("device tree") and project-wide POUs in the "POUs" view.
- For the creation of projects, there are templates that already contain certain objects.
- Basic configurations and information for the project are defined in the "Project Settings" and "Project Information". For example:
- Compiler settings
- User management
- Author
- Data about the project file

There are settings for the version compatibility of the project in the configuration dialogs in the "Project Environment".

- You save a project as a file in the file system. As an option, you can pack it together with project-relevant files and information into a project archive. It is also possible to save files in a source code management system such as SVN.
- Each project contains the information about the CODESYS version with which it was created. When you open it in another version, CODESYS will notify you about possible or necessary updates regarding file format, library versions, etc.
- You can compare, import/export projects, and create documentation for them.
- You can protect a project from being changed, or even completely protect it from being read. By using user management, you can selectively control the access to the project and even to individual objects in the project.
- $\quad \otimes$ Chapter 6.4.1.21.2.2 "Object 'Application'" on page 2410
- $\underbrace{*}$ Chapter 6.4.1.21.2 "Objects" on page 2409
- $\underbrace{*}$ Chapter 6.4.1.21.4 "Dialogs" on page 2745
- ${ }^{\bullet}>$ Chapter 6.4.1.21.3.5.13 "Command 'Project information'" on page 2604
- $\Leftrightarrow$ Chapter 6.4.1.6 "Protecting and Saving Projects" on page 1819

Auto generated When creating, saving or exporting, files may be created and their extensions are briefly listed files here for better understanding. Some of the files are generated when you login for the first time to the PLC and the boot application is created. The list does not claim to be complete.

| File extension | Description |
| :--- | :--- |
| *.opt | Project specific options. |
| *.bootinfo | Will be created during boot project creation. <br> When creating a boot project the bootinfo is created. So <br> that this must not be read in completely to find out whether <br> the information is relevant, the checksums are created in a <br> separate file. |
| *.bootinfo_guids | Contains the CRC of the bootinfo. |
| *.crc | Checksum of the boot application. |


| File extension | Description |
| :--- | :--- |
| *.compileinfo | Will be created when downloading the project. <br> The compileinfo is loaded when opening the project, when <br> logging in it is checked if it matches the project on the <br> controller, if not the bootinfo-Guids are checked and if they <br> match then the bootinfo is loaded. |
| *.xml | Contains the symbols exported via symbolconfig symbolfile. |
| *.precompilecache | The precompile cache is only relevant when loading the <br> project. This is faster if the cache is available. |
| *.app | Application file. A boot application is the application that is <br> started automatically when the controller is switched on or <br> started. For this to happen, the application on the controller <br> must exist as a file. |

Handling of AC500 projects such as project creation, export/import, comparison of projects etc. is described in the sections for AC500 V3 products.
Chapter 6.3.1.1 "Project handling" on page 1398

### 6.2.12 Handling of AC500 projects

Handling of AC500 projects such as project creation, export/import, comparison of projects etc. is described in the sections for AC500 V3 products.
${ }^{\star}$ Chapter 6.3.1.1 "Project handling" on page 1398

Copy-and-paste from one project to another project in two different Automation Builder instances is possible. After copying parts of a project to a higher Automation Builder version the copied components have to be updated.


Project archive Automation Builder supports the creation and the import of project archive files. Archive files contain all relevant project data including the PLC configuration, the CODESYS project files and all device descriptions. This allows exchanging Automation Builder projects without taking care of the target environment General Settings. « Chapter 6.3.1 "General settings" on page 1398

User and access The 'User Management' provides functions for defining user accounts and configure the access rights of a project rights within a project. The rights to access project objects via specified actions are assigned only to user groups, not to a single user account. So each user must be member of a group General Settings. ${ }^{\mu}$ Chapter 6.3.1 "General settings" on page 1398

### 6.2.13 Connection of devices

### 6.2.13.1 Configuring devices

All installed devices that are available in Automation Builder are listed in the device repository.
« Chapter 6.2.10 "Device repository" on page 1387

Modify your Automation Builder project by adding device objects. Preset items can be replaced in the same way.

1. In the device tree, right-click an item node. Select [Add object].

2. Select the desired object and click [Add object].
3. Double-click the new object in the device tree to configure the device settings. Depending on the selected item different configuration tabs are available.

| A1523 Parameters | Parameter | Type | Value | Defaut Value Unit | Description |
| :---: | :---: | :---: | :---: | :---: | :---: |
| A1523 fo Mapping | - Ignore modic | Enumerstion of BrTE | No | No | This parameter allows to set whether |
|  | - Cheds rupoly | Enumerstion of ETTE | On | On | Obeck supply |
| V/0 mapping lat | - Inout 0, chamel conflgraton | Enumerstion of BYTE | Not used | Not used | Analog ingot 0-Configuration of man |
|  | - Input 0 , ched chamel | Enumeration of BYTE | Pheusib, Cut wire, short cireat | Phausib, Cat wire, Stort diruit | Anslog incut 0 - Check dannel |
| Information | - Irout 1, chamel configration | Enumeration of SYTE | Not used | Not used | Analog ingot 1- Configuration of ana |
|  | - trout 1, check chamel | Enumeration of EYTE | Plisusib, Cut wire, Short circut | Plausib, Cut wre, Short dircuit | Anslog ingut 1-Check diannel |
|  | - Inout 2, cramel confgaraton | Enumeration of BrTE | Not used | Not used | Anslog ingut 2- Configuration of ana |
|  | - Irout 2, deck chamel | Enumerstion of BrTE | Plavsib, Cut wire, Short dircuit | Flausib, Cut wire, Short circuit | Anslog input 2 - Check channel |
|  | - Inout 3, charnel conflgraton | Enumeration of BrTE | Not used | Not used | Ansalog ingut 3-Configuration of ana |
|  | - Inout 3, chedichamel | Enumerstion of BYTE | Plousib, Cut wire, Short ircuit | Plausib, Cut wire, Sort circuit | Anslog ingut 3-Check dasnel |
|  | - Input 4, chamel conflgraton | Enumeration of BYTE | Not used | Not used | Analog ingut 4-Coefiguration of anam |
|  | - Inout 4, deedchasmel | Enumeration of EYTE | Pheusib, Cut wire, Short dircuit | Plausib, Cut mire, Stort dircuit | Anslog ingut 4 - Check dannel |
|  | - Inout S, chamel configraton | Enumeration of BYTE | Not used | Not used | Analog ingut 5 - Configuration of ana |

### 6.2.13.2 Symbolic names for variables, inputs and outputs

The IEC naming rules are not checked during input in Automation Builder.

Input and output Devices with I/Os provide an "//O Mapping" tab in their configuration editor where the available mapping $\quad \mathrm{I} / \mathrm{O}$ channels can directly be mapped to a global variable.

The corresponding variable declarations are automatically available in the project.
All available I/O channels can easily be assigned to a variable.

| Variable | Mapping | Channel | Address | Type | Default Value | Unit | Description |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| - b_Input_IBO | ${ }^{*}$ | Digital inputs $10-\mathrm{I7}$ | \%IB0 | BYTE |  |  |  |
| x_Input_IB0_IX0 | * | Digital input IO | \%IX0.0 | BOOL |  |  |  |
| x_Input_IB0_IX1 | * | Digital input I1 | \%IX0.1 | BOOL |  |  |  |
| * x_Input_IB0_IX2 | * | Digital input I2 | \%IX0.2 | BOOL |  |  |  |
| x_Input_IB0_IX3 | * | Digital input I3 | \%IX0.3 | BOOL |  |  |  |
| x_Input_IB0_IX4 | * | Digital input I4 | \%IX0.4 | BOOL |  |  |  |
| * x_Input_IBO_IX5 | * | Digital input 15 | \%IX0. 5 | BOOL |  |  |  |
| * x_I $^{\text {Input_IB0_IX6 }}$ | * | Digital input 16 | \%IX0. 6 | BOOL |  |  |  |
| x_Input_IB0_IX7 | * | Digital input I7 | \%IX0. 7 | BOOL |  |  |  |
| + ${ }^{4}$ |  | Digital inputs I8-I15 | \%IB1 | BYTE |  |  |  |
| $\pm 4$ |  | Digital inputs I16-I23 | \%IB2 | BYTE |  |  |  |
| $\pm{ }^{+}$ |  | Digital inputs I24-I31 | \%IB3 | BYTE |  |  |  |
| $\pm$ Fast counter |  |  |  |  |  |  |  |

## AC500 uses Intel Byte Order (Little Endian).

### 6.2.13.3 Update of AC500 devices

Perform a firmware update to update AC500 V3 devices. «̌ Chapter 6.3.1.4.3 "Installation and update of the AC500 V3 firmware" on page 1422

### 6.2.13.4 Comparing objects

To compare similar objects within a project (such as the project configuration) select both objects. Right-click and select "Compare Objects" to see the differences.


### 6.2.14 Connection of serial interfaces

Depending on the device type, the configuration of serial interfaces is different.
(4) Chapter 6.3.2.14 "Serial interface" on page 1586

### 6.2.15 Converting an AC500 V2 project to an AC500 V3 project

Application A project that has been configured for an AC500 V2 PLC can be converted to a example project for an AC500 V3 PLC. Essentially, the conversion is done in Automation Builder, however, some additional actions have to be executed manually. The complete conversion procedure of a V 2 project to a V 3 project is described in the application note Converting an AC500 V2 project to an AC500 V3 project.

### 6.2.16 Automation Builder installation manager

### 6.2.16.1 Introduction

Automation Builder installation manager allows you to install customer specific software packages, modify the existing installation, update installation information and to uninstall Automation Builder software packages in a comfortable and flexible way.
$\triangleright \quad$ You can launch installation manager from the main menu of Automation Builder, "Tools
$\rightarrow$ Installation Manager", or from Windows start menu.
$\Rightarrow$ Installation manager starts.

## AB ABB Automation Builder <br> Automation Builder

Engineering Productivity


Help
Installed packages Licensing information Installed updates

| Name | Type | Version | Installation Date | $\wedge$ | Uninstall all |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Libraries | CodesysPackageComponent | 2.11.0.2217 | 04/02/ |  |  |
| StandardFieldbus Devices | CodesysPackageComponent | 2.11.0.2215 | 04/02/ |  |  |
| Visualization Files | CodesysPackageComponent | 2.11.0.2215 | 04/02/ |  | Install Package |
| Automation Builder Platform Extensions | CodesysPackageComponent | 2.5.0.1881 | 04/02/ |  |  |
| AC31 Replacement | CodesysPackageComponent | 2.11.0.2218 | 04/02/ |  | Modify |
| AC500-Co-V3 | CodesysPackageComponent | 2.11.0.2218 | 04/02/ |  |  |
| AC500-Co | CodesysPackageComponent | 2.11.0.2218 | 04/02/ |  |  |
| AC500-S V3 | CodesysPackageComponent | 2.11.0.2218 | 04/02/ |  |  |
| AC500-S | CodesysPackageComponent | 2.11.0.2218 | 04/02/ |  |  |
| AC500-S Configuration | CodesysPackageComponent | 2.11.0.2218 | 04/02/ |  |  |
| AC500 V3 | CodesysPackageComponent | 2.11.0.2218 | 04/02/ | $\checkmark$ |  |


| Info Export | Check for new service release |
| :---: | :---: |

Uninstall all
Install Package
Modify
Info Export
Check for new
service release

Uninstalls the currently installed Automation Builder software. 少 Chapter 6.2.16.5 "Uninstalling Automation Builder" on page 1397
Installs customer specific software packages. $\Rightarrow$ Chapter 6.2.16.2 "Installing customer specific package" on page 1395
Adds or removes installed software packages. «y Chapter 6.2.16.3 "Adding or removing installed software packages" on page 1396
Exports detailed information of installed packages in a notepad.
Checks if your installed version of Automation Builder is up to date and checks for updates.
\# Chapter 6.2.16.4 "Checking for updates" on page 1396

### 6.2.16.2 Installing customer specific package

Installation manager allows you to install customer specific software packages (CABPKG files). These packages are separately distributed to the customer based on the customer requirement.

1. In the installation manager, click [Install Package].
2. Select the package to be installed (.cabpkg file) from the file system.
3. Select the components to be installed.
4. Click [Install].
$\Rightarrow$ Data installation starts.
5. Successfully installed components are indicated with

Errors during data download are indicated with $\otimes$. Errors during download of any package component aborts the installation. In this case click "Show Log" and save the log data. Send the log file to ABB support team.

Click [Finish] to end the wizard.

### 6.2.16.3 Adding or removing installed software packages

1. In the installation manager, click "Modify".
$\Rightarrow$ The selection page opens.
The selected software packages are installed already.
The not selected software packages are not installed.
2. Select the software packages you want to install.

Unselect the software packages you want to uninstall.

You cannot unselect the main Automation Builder software package.
3. Click "Continue".
$\Rightarrow$ First, the unselected software packages will uninstall.
Afterwards, the new selected software packages will be downloaded and installed.
4. Successfully downloaded components are indicated with 0 .

Errors during data download are indicated with *. Errors during download of any package component aborts the installation. In this case click "Show Log" and save the log data.
Send the log file to ABB support team.
Click [Finish] to end the wizard.

### 6.2.16.4 Checking for updates

$\triangleright$ In the installation manager, click "Check for new service release".
$\Rightarrow$ If a newer Automation Builder version is available, you will get an option to download and install the new version.


Create a project archive before upgrading Automation Builder. Project archives contain all project data, including data that is not stored with a *.project file, e.g. device description files for third party devices.
\& Chapter 6.3.1.1.8.2 "Creation of an archive " on page 1408

### 6.2.16.5

## Uninstalling Automation Builder

Installation manager offers a comfortable way to uninstall the Automation Builder software. This will uninstall all related packages of Automation Builder platform as well, such as Mint Plug-in, Automation Builder Extensions, Drive Manager etc.

1. In the installation manager, click [Uninstall all].
$\Rightarrow$ A warning message is displayed to uninstall Automation Builder software.
Click [Yes] to continue.

2. If Automation Builder instances are running, a warning message is displayed.

Close running instances of Automation Builder and click [Retry] to continue uninstallation.
With [Abort] uninstallation of the current package is stopped. Uninstallation is continued with the next package. With [lgnore], uninstallation is forced. As this can lead to an erroneous uninstallation, we recommend you, not to use this option.

3. If installation manager was launched with "Tools $\rightarrow$ Installation Manager", the following message is displayed as Automation Builder is still running:

With [Yes] Automation Builder software is closed to continue uninstallation procedure.
With [No] uninstallation of the current package is stopped. Uninstallation is continued with the next package.

| ABB Automation Builder |
| :--- |
| The following options will be uninstalled: <br> PLC - AC500 V2 <br> Safety PLC - AC500-S <br> Do you want to proceed? |
|  |
|  |

4. For each of the packages being uninstalled, system may prompt to continue uninstallation.
5. Successfully uninstalled components are indicated with $Q$.

Errors during uninstallation are indicated with $\otimes$. Errors during uninstallation of any package component aborts the uninstallation. In this case click "Show Log" and save the log data. Send the log file to ABB support team.
Click [Finish] to end the wizard.

### 6.3 Configuration in Automation Builder for AC500 V3 products

### 6.3.1 General settings

### 6.3.1.1 Project handling

### 6.3.1.1.1 General

What is a project?

- A project contains the objects which are necessary to create a controller program ("application"):
- Pure POUs, for example programs, function blocks, functions, and GVLs.
- Objects that are also required to be able to run the application on a PLC. For example, task configuration, Library Manager, symbol configuration, device configuration, visualizations, and external files.
- In a project, you can program multiple applications and connect multiple controller devices.
- CODESYS manages device-specific and application-specific POUs in the "Devices" view ("device tree") and project-wide POUs in the "POUs" view.
- For the creation of projects, there are templates that already contain certain objects.
- Basic configurations and information for the project are defined in the "Project Settings" and "Project Information". For example:
- Compiler settings
- User management
- Author
- Data about the project file

There are settings for the version compatibility of the project in the configuration dialogs in the "Project Environment".

- You save a project as a file in the file system. As an option, you can pack it together with project-relevant files and information into a project archive. It is also possible to save files in a source code management system such as SVN.
- Each project contains the information about the CODESYS version with which it was created. When you open it in another version, CODESYS will notify you about possible or necessary updates regarding file format, library versions, etc.
- You can compare, import/export projects, and create documentation for them.
- You can protect a project from being changed, or even completely protect it from being read. By using user management, you can selectively control the access to the project and even to individual objects in the project.
- $\Leftrightarrow$ Chapter 6.4.1.21.2.2 "Object 'Application"' on page 2410
- $\underbrace{*}$ Chapter 6.4.1.21.2 "Objects" on page 2409
- \& Chapter 6.4.1.21.4 "Dialogs" on page 2745
- ${ }^{\bullet}$ Chapter 6.4.1.21.3.5.13 "Command 'Project information'" on page 2604
- $y^{4}$ Chapter 6.4.1.6 "Protecting and Saving Projects" on page 1819


### 6.3.1.1.2 Creating a new project

1. Select "File $\rightarrow$ New Project".
2. Select "AC500 project", enter a project name and specify the storage location for the new project.

With "Empty project" a project without a PLC is created.
3. Select the device type for the new project and click [Add object].

$\Rightarrow$ A new project is created and can be configured.

### 6.3.1.1.3 Opening an existing project

## NOTICE!

Risk of damaging Automation Builder projects!
Projects created with Automation Builder are incompatible with CODESYS V2.3.9.x. Do not open projects with CODESYS V2.3.9.x as this can cause corrupted Automation Builder projects.


Automation Builder performs an integrity check for the PLC configuration before generating the configuration.

## Opening a

 project1. Select "File $\rightarrow$ Open Project".
$\Rightarrow$ The "Open Project" dialog appears.
2. Select a previously saved project from the file system.
$\Rightarrow$ Automation Builder switches to the version of the project and opens the project.

### 6.3.1.1.4 Exporting and importing a project

Configuration of a complete PLC or of single devices can be reused within the same project by copy-and-paste the desired nodes in the device tree.
In order to reuse a PLC configuration cross-over projects, the project configuration can be exported and imported afterwards into another project.

An exported project configuration can only be imported to a project with the same Automation Builder version. If the versions are not the same, the import fails with an error message.

Automation Builder performs an integrity check for the PLC configuration before generating the configuration.

Project export From the menu, select "Project $\rightarrow$ Export $\rightarrow$ Project". Select the objects to be exported. The configuration of the selected items will be added to an export file (*.export).

"One file per subtree": If this option is activated, all objects belonging to the same subtree will be exported into the same export file, otherwise a separate file will be created for each particular object.

Project import For importing a project a basic and an advanced function is available.
"Basic project import": Users with a basic or a standard Automation Builder license can perform a basic project import. Command: "Project $\rightarrow$ Import $\rightarrow$ Project".

A previously exported project configuration is imported into the current project. With this, the current project configuration is overwritten.
In order to supplement the current project with the project configuration of a previously exported project, use the compare function. Command: "Project $\rightarrow$ Compare".
② Chapter 6.3.1.1.7 "Comparing projects" on page 1405
"Advanced project import": Users with a premium Automation Builder license can perform an advanced project import. Command: "Project $\rightarrow$ Import $\rightarrow$ Project with compare". This command allows to compare two projects, to check on differences and to adapt single parts of the project configuration easily.

## Basic project import

1. From the menu, select "Project $\rightarrow$ Import $\rightarrow$ Project".

> A previously exported project configuration is imported into the current project. With this, the current project configuration is overwritten.
> In order to supplement the current project with the project configuration of a previously exported project, use the compare function. Command: "Project $\rightarrow$ Compare".
> घ Chapter 6.3.1.1.7 "Comparing projects" on page 1405
2. Select the export file from the file system and click [Open] to import the project configuration.

An exported project configuration can only be imported to a project with the same Automation Builder version. If the versions are not the same, the import fails with an error message.

## Advanced project import

Perform an advanced project import in order to compare two projects, to check on differences and to adapt single parts of a previously exported project configuration easily.

1. From the menu, select "Project $\rightarrow$ Import $\rightarrow$ Project with compare".
2. Select the export file from the file system and click [Open] to import the project configuration.
$\Rightarrow$ The project import is started.
3. Once the project file is imported, a compare view is displayed. The left pane represents the current project, the right pane represents the imported project.


## Comparison options Ignore Spaces: OFF Ignore comments: OFF Ignore properties: on

| Comparison statistics 1 | 1 addition(s) | 4 deletion(s) 3 change(s) |
| :--- | :--- | :--- | :--- |

$\Rightarrow$ Differences between the current project and the imported project are highlighted in red color.
$\Rightarrow$ Additional modules in the current project that are not available in the imported project are highlighted in green color.
$\Rightarrow$ Additional modules in the the imported project or deleted modules in the current project are highlighted in blue color.
$\Rightarrow$ A summary of all differences within the projects is given in the "Comparison statistics" under the device tree.
4. Every highlighted item of both projects can be handled individually and can either be transferred to the current project or skipped.

- [Accept Block]: All items of the selected node are transferred to the current project with one click. Use this function for example to copy all nodes of a PLC configuration from the imported project to the current project (select "//O_Bus" node).
- [Accept Single]: Only a single item from a node is transferred to the current project.

Use this function for example to copy certain I/O modules from the imported project to the current project.

$\Rightarrow$ All accepted items are highlighted in the current project in yellow color.
$\Rightarrow$ To undo a selection, again, click [Accept Block] or [Accept Single].
$\Rightarrow$ To accept all changes on the current project, close the "Project Comparison Differences" tab and confirm the prompted dialog.
5. If in the import project the PLC contains an AC500-S safety module, a security check is performed which requires user authentication:

6. After a successful user authentication the AC500-S safety modules are added to the compare view and can be imported to the current project.

6.3.1.1.5 Upgrade a project to a new Automation Builder version or profile

With the Automation Builder version 2.6 .0 a side-by-side installation is possible, i.e. any Automation Builder versions can be installed in parallel.
Automation Builder 2.5 or a lower version can be installed for access to all required previous versions.
When updating the Automation Builder version, a previously configured project must be converted in order to be used in a new Automation Builder version.
1.

"File $\rightarrow$ Project Archive $\rightarrow$ Save/Send Archive..."
2. Use the Automation Builder update notification to inititate the software update.

๕ Chapter 6.2.2 "Automation Builder updates" on page 1350
or:
3. Download the file from the PLC website - all versions of the Automation Builder are available.
4. Install the new Automation Builder and follow the instructions.
5. Open a previously configured project.
$\Rightarrow$ Automation Builder performs an integrity check for the PLC configuration before generating the configuration.
Automatically a window appears with the request to upgrade the project.

Automation Builder - Version mismatch

After upgrading the file to Automation Builder 2.7, it cannot be opened with older versions. Cancel: the project file will be closed unchanged.

Please select
(-) Upgrade the project file to version: Automation Builder 2.7 (link to release notes)
It could take several minutes.
$\checkmark$ Create backup (Z:VAB 2.6.1\Project1_AB_261_2023_9_25_14_36_55_Backup.project)
6. Click the [OK] button.
$\Rightarrow$ During the project upgrade, an option for migration of third party devices can be selected.

```
Automation Builder - Version mismatch
```

After upgrading the file to Automation Builder 2.7, it cannot be opened with older versions. Cancel: the project file will be closed unchanged.

7. Click the $[O K]$ button to upgrade third party devices.

If this option was not selected during the upgrade procedure, migration can be initiated manually after an Automation Builder upgrade in order to migrate all third party devices to the project.
\& Chapter 6.3.1.5 "Migration of third party devices" on page 1439
8. The firmware of the communication interface modules will not automatically be upgraded. You can do this with the IP configuration tool. For this, the IP configuration tool must be used as stand-alone variant.
Follow the instructions for the firmware upgrade for communication interface modules.
« Chapter 6.3.2.2.4.3.2 "Stand-alone installation" on page 1457

| Application | A project that has been configured for an AC500 V2 PLC can be converted to a <br> example |
| :--- | :--- |
|  | project for an AC500 V3 PLC. Essentially, the conversion is done in Automation Builder, <br> however, some additional actions have to be executed manually. The complete conver- <br> sion procedure of a V2 project to a V3 project is described in the application note <br> Converting an AC500 V2 project to an AC500 V3 project. |



It is not possible to downgrade a project to an earlier Automation Builder version.

### 6.3.1.1.6 I/O mapping export and import

Export I/O map- To exchange information on I/O mapping only, data can be exported as .csv file. This allows ping maintenance of I/O data outside Automation Builder, e.g. in MS Excel.

Right-click the "Processor Module" node or "//O_Bus" node in the device tree and select "Export -> IO mapping". To export the I/O Mapping for the complete project, e.g. with more than one configured processor modules, I/O data of the complete project can be exported "Project -> Export -> I/O mapping".

Import I/O map- A previously exported .csv file can be imported to the project:
ping "Project -> Import -> I/O mapping".

### 6.3.1.1.7 Comparing projects

## Introduction

You can compare the currently open project with another project - a reference project. The differences in contents, properties, or access rights are detected and shown in a comparison view.


## Application example

How to compare Automation Builder projects for an
AC500 V3 PLC is decribed in the application note
Project Comparison - Compare AC500 Projects and Integrate the Differences.

Clicking "Project $\rightarrow$ Compare" opens the "Project Compare" dialog for you to configure and run the comparison. Then the result is shown in the comparison view "Project Compare Differences" where the objects are aligned in a tree structure. Objects that indicate differences from the respective reference object are identified by colors and symbols. This is how you detect whether or not the contents, properties, or access rights are different.
For differences in the contents, you can also open the detailed compare view "Project Compare - <object name> Differences" in order to zoom into the object. In the detailed compare view, the contents of the object and reference object are displayed or their source code aligned. The detected differences are marked. Previously opened views are not closed. In this way, you can have any number of comparison views open and read them, in addition to the project compare view.

You can accept the detected differences from the reference project into the current project. This is possible only from the reference project into the open project. To do this, you activate differences (for example in the code) that should be accepted in the current project with the commands $\#, \downarrow$, or ${ }^{3}$ in the active comparison view for accepting. These positions are highlighted in yellow. Make sure that any other open compare views are inactive (write-protected, read-only). therefore, you can activate differences to be accepted in exactly one comparison view only. When exiting the active compare view, if you confirm that the differences that are activated for acceptance are actually accepted into the current project, then the current project is modified.

In order to exit the project comparison completely, close the project compare view.

## Creating a comparison view

Requirement: You have made changes in your current project and wish, for example, to compare it with the last-saved version. In the meantime, for example, you have added further POUs, removed a POU, changed single lines of code or the object properties in function blocks.

1. Select the command "Project $\rightarrow$ Compare".
$\Rightarrow$ The "Project Comparison" dialog box opens.
2. Enter the path to the reference project, for example the path to the last-saved version of your current project.
3. Leave the activation of the comparison option "Ignore Spaces" as it is.
4. Click on "OK".
$\Rightarrow$ The comparison view opens. Title: " Project Comparison - Differences". The Device trees of the current project and the reference project are displayed alongside each other and the changed objects are marked in color
5. Select an object marked in blue in the tree of the reference project (right). The current project no longer contains this object.

Click on $\downarrow$ "Accept Single"
$\Rightarrow$ The object is added to the tree of the current project (left). The line has a yellow background. * ${ }^{\text {a }}$ appears in the middle column.
6. Select an object marked in green in the tree of the current project (left). The reference project does not contain this object.
Click on $\downarrow$ "Accept Single"
$\Rightarrow$ The object is removed again from the tree of the current project (left). The line has a yellow background. ${ }^{\text {m }}$ appears in the middle column.
7. If changes are detected in the content of an object that is contained in both the current project and the reference project, this is indicated by red lettering. You can then switch to the detailed comparison view for the object by double-clicking on the object.
8. Close the comparison view and answer the query whether the changes made are to be saved with "Yes".
$\Rightarrow$ The changes become effective in the project.

## Opening the detailed compare view

Requirement: For example, a user modified the code in a POU of the current project. You have performed the project comparison by clicking "Project $\rightarrow$ Compare". The project compare view shows this POU highlighted in red in the aligned in the project tree.

1. Double-click the line of the aligned POU versions.
$\Rightarrow$ The compare view switches to the detailed compare view of the POU. The modified code lines are highlighted in gray and written in red.
2. Click $=$
$\Rightarrow$ Code lines with changes (red) are extended by two lines: an line with insert (left, green) and a line with delete (right, blue).
3. Click $\rightleftharpoons$ again.
$\Rightarrow$ The code line is marked again as modified.
4. Move the mouse pointer to the code line marked as modified and click $\checkmark$ "Accept Single".
$\Rightarrow$ The code line from the reference project is activated for acceptance into the current project.
5. Click
$\Rightarrow$ The project compare view opens for the entire project. It is write-protected (read-only) to prevent you from activating differences for acceptance. The link highlighted in yellow above the tree view also indicates this.
6. Click the link: "Project compare view is read only because there are uncommitted changes in another view. Click here to switch to the modified view."
$\Rightarrow$ The detailed compare view opens again. The unconfirmed changes are highlighted in yellow.
7. Click $\times$ in the tab of the view and confirm that the changes should be saved.
$\Rightarrow$ The detail project view is closed and the POU is overwritten. Now it corresponds to the POU of the reference project. The project view is active again so that you can continue working with project compare.

If you do not click the link, but click $\times$ instead to close the editor of the project compare view, then you will also confirm the acceptance of changes into the current project. The detail changes are accepted and then the project compare is closed completely.

See also

- ※ Chapter 6.4.1.5 "Comparing projects" on page 1817
- $y^{2}$ Chapter 6.4.1.21.3.5.21 "Command 'Compare"' on page 2607
- ② Chapter 6.3.1.1.7.2 "Creating a comparison view" on page 1406


### 6.3.1.1.8 Project archive

## General

Automation Builder supports the creation and the import of project archive files. Archive files contain all relevant project data including the PLC configuration, the project files of the CODESYS and all device descriptions. This allows exchanging Automation Builder projects without taking care of the target environment.

## Creation of an archive

$\checkmark$ The following steps describe the creation archive file from an Automation Builder project:

1. Select "File $\rightarrow$ Project Archive $\rightarrow$ Save/Send Archive".
2. Select the information which should be included in the archive file from the list box.

| Section/Control | Parameter | Description |
| :--- | :--- | :--- |
| Information <br> selection list box | Options | Not supported |
|  | Referenced <br> devices | The referenced devices can be selected by expanding <br> the "Referenced devices" item of the list box. It is <br> strongly recommended to include all devices in the <br> project archive to maintain consistency. |
| Additional files | - | Not supported |
| Comment | - | Opening a control window which allows the input of a <br> comment to the project archive. |
| Save | - | Opening a dialog window to determine the path and the <br> file name of the project archive and storing it to the file <br> system. |
| Send | - | Not supported |
| Cancel | - | Canceling the operation and closing the dialog window. |

With [Comment] additional information can be added to the project archive, for example to add a brief description or some information concerning the project.
3. Proceed with [Save...].

It is strongly recommended to keep the default settings.

## Extraction of an archive



The currently loaded project will be closed automatically when extracting the selected project archive. It is recommended to open a new instance of Automation Builder before starting the extraction process.

The following steps describe the extraction of an archive file and the import to Automation Builder.

1. Select "File $\rightarrow$ Project Archive $\rightarrow$ Extract Archive".
2. Select the desired project file and click [Open].

| Section/ <br> Control | Parameter | Description |
| :--- | :--- | :--- |
| Locations | Extract into the <br> same folder where <br> the archive is <br> located | The project archive will be extracted to the same path <br> where the archive is located. |
|  | Extract into the fol- <br> lowing folder | Path to which the project archive should be extracted. |
|  | Button ... | Opening a folder selection dialog which allows <br> selecting the desired path. |
| Contents | Items | Select the items which should be extracted. |
|  | Comment | Displaying comments included inside the Project <br> archive file. |
|  | Triggering the extraction process. Automation Builder <br> extracts the archive and creates a project from out <br> the archive. After creating the project Automation <br> Builder checks the version of the project. If the <br> project version and the activated Automation Builder <br> version is not identical the workflow is the same as <br> described in "Opening an Existing Project". |  |
| Cancel | Closing the Extract Project Archive dialog and can- <br> celing the extraction process. |  |

### 6.3.1.2 User and access rights management

### 6.3.1.2.1 User and access rights

## General

The "User Management" provide functions for defining user accounts and configure the access rights within a project. The rights to access project objects via specified actions are assigned only to user groups, not to a single user account. So each user must be member of a group.

## User management

With the help of the integrated user management, user groups with different access rights and authorizations can be defined. Configuration and handling of the user management in Automation Builder and a AC500 V3 is decribed in the application note User Management with V3 - Configuration and Handling.

## NOTICE!

For security reasons we recommend activating the user administration for the PLC (and the Visualization).

Please note that when user management for the PLC is enabled, the username and password are always required to connect to the PLC.

If you have lost the administrator password, there is no way to recover it and the PLC must be sent in for reset!

² Chapter 6.1 "Cyber security" on page 1340

Before setting up users and user groups, notice the following: The configuration of users and groups is done in the Project Settings dialog.
„ Chapter 6.3.1.2.3 "Project Settings - Users and groups" on page 1413

- Automatically there is always a group "Everyone" and by default primarily each defined user or other groups are members of this group. Thus each user account at least automatically is provided with defined default settings. Group "Everyone" cannot be deleted, just renamed, and no members can be removed from this group.
- Also automatically there is always a group "Owner" containing one user "Owner". Users can be added to or removed from this group, but at least one user must remain. This group also cannot be deleted and always has all access rights. Thus it is not possible to make a project unusable by denying the respective rights to all groups. Both group and user "owner" might be renamed.
- When starting the programming system resp. a project, primarily no user is logged on the project. But then the user optionally might log on via a defined user account with user name and password in order to have a special set of access rights.


## Notice that each project has its own user management!

So, for example to get a special set of access rights for a library included in a project, the user must separately log on to this library. Also users and groups, set up in different projects, are not identical even if they have identical names.

## CAUTION!

## The user passwords are stored irreversibly!

If a password gets lost, the respective user account gets unusable. If the "Owner"-password gets lost, the entire project might get unusable!

## Access right management

User management in a project is only useful in combination with the access right management. Notice the following:

- In a new project basically all rights are not yet defined explicitly but set to a default value. This default value usually is: "granted".
- In the further run of working on the project each right can be explicitly granted or denied resp. set back to default. The access right management of a project is done in the Permissions dialog ${ }^{«}$ "Permissions" on page 1411.
- Access rights on objects get "inherited". If an object has a "father" object (example: if an action is assigned to a program object, that is inserted in the structure tree below the program, then the program is the "father" of the action object), the current rights of the father automatically will become the default settings of the child. Father-child relations of objects concerning the access rights usually correspond with the relations shown in the POUs or Devices tree and are indicated in the Permissions dialog by the syntax "<father object>.<child object>".


#### Abstract

Example Action ACT is assigned to POU object PLC_PRG. So in the POUs window ACT is shown in the objects tree indented below PLC_PRG. In the Permissions dialog ACT is represented by "PLC_PRG.ACT" indicating that PLC_PRG is the "father" of ACT. If the "modify" right would be denied explicitly for PLC_PRG and a certain user group, the default value of the "modify" right for ACT automatically also would be "denied".


### 6.3.1.2.2 User management commands

The 'User Management UI' plug-in provides commands for command category 'User Management'.

These are used for:

- Configuration of access rights on the project objects
- Logging on or off to/from the project via a defined user account in order to get the access rights which are associated to this account
The configuration of user accounts and groups is done in the "Project Settings" subdialog "User Management" $\Longleftrightarrow$ Chapter 6.3.1.2.3 "Project Settings - Users and groups" on page 1413.
By default the following commands are part of submenu "User Management" in the "Project" menu: "Logon", "Logoff", "Permissions".


## Logon

Symbol:
This command opens the "Logon" dialog for logging on to a project or library via a defined user account.
Logging on with a certain user account means to log on with those object access rights which are granted to the group which the user belongs to. The configuration of user accounts and groups is done in the "Project Settings" subdialog "User Management".
To log on select the project or an included library from the selection list in the Project/Library field. Enter User name and Password of a valid user account, noticing that each project or library has an own user and access rights management. Log on with OK.
If already another user is logged on the project, this one will be logged out automatically by the new log-on action.
When you are logged on to a project or library and try to perform an action for which you have no right, automatically a Logon dialog will be opened, giving the possibility to log on with another user account provided with the appropriate rights.
The status bar always displays which user currently is logged on the project.

```
Gurrent user: User1
```


## Logoff Symbol: $\otimes$

This command logs off the currently logged on user. If no user had been logged on to the currently opened project or to a referenced library an appropriate message will appear when trying to log off.
If the user currently is logged on to more than one project or referenced library (not necessarily with the same user account) a "Logoff" dialog will appear when trying to log off.
From the Project/Library selection list choose those project/library for which you want to log off. The name of the current user is displayed just for information.

The status bar always displays which user currently is logged on the project.

Permissions This command opens the "Permissions" dialog, where the rights to work on objects or to perform commands in the current project can be configured.

## Any changes made in this dialog will be applied immediately.



The Actions window displays all possible rights, that is all actions which might be performed on any object of the current project.
The tree is structured in the following way:

- Top-level see the names of some categories, which have been set up just for the purpose of optical structuring the rights management.
They are grouping concerning the execution of commands, the configuration of User accounts and Groups, the creation of Object Types, the viewing, editing, removing and handling of child objects of Project Objects.
- $\xi$ Below each category node there are nodes for the particular actions which might be performed on the command, user account, group, object type or project object. These nodes also only have optical function. Possible Actions:
- execute (execution of a menu command)
- create (creating a new object in the current project)
- add or remove children (adding or removing of "child" objects to an existing object)
- modify (editing an object in an editor)
- remove (deleting or cutting an object)
- view (viewing an object in an editor)

Below each action node find the possible targets, that is project objects, of the respective action.

The Permissions window provides a list of all currently available user groups (except the "Owner" group) and a toolbar for configuring rights to a group.
Select the group and configure it's permissions.
Left to each group name one of the following icons indicates the currently assigned permission concerning the target which is currently selected in the Actions window:

- $\wp$ : The action(s) $z$ for the target(s) $\Rightarrow$ currently selected in the Actions window are granted for the selected group.
- $\quad$ : The action(s) $\Rightarrow$ for the target(s) $\Rightarrow$ currently selected in the Actions window are denied for the selected group.
- $\quad$ : The right to perform the action(s) $/ \bar{z}$ which are currently selected for the selected target(s) $\Rightarrow$ in the Actions window, has not been granted explicitly, but is granted by default, for example because the corresponding right has been granted to the "father" object. (Example: The group has got the right for object "myplc", thus it by default it also has got it for object "myplc.pb_1". ) Basically this is the default setting for all rights which not explicitly have been configured.
- $\quad$ : The right to perform the action(s) $\bar{y}$ which are currently selected for the selected target(s) $\Rightarrow$ in the Actions window, has not been denied explicitly, but is denied by default, for example in case because the corresponding right has been assigned to the "father" object.
If currently multiple actions are selected in the Actions window, which do not have unique settings referring to the currently selected group, no icon will be displayed.

To configure the rights for a group select the desired action(s) $\bar{y}$ and target $\Rightarrow$ in the Actions window and the desired group in the Permissions window. Then use the appropriate button in the toolbar of the Permissions window:
§ Grant: Explicit granting.
= Deny: Explicit denying.
© Clear: The currently granted right for the action(s) currently selected in the Actions window will be deleted, that is set back to the default.

### 6.3.1.2.3 Project Settings - Users and groups

## General

The "Project Settings" dialog in category "Users and Groups" provides three subdialogs for the user management for the current project: Users, Groups, Settings.
$\Leftrightarrow$ General description on users and access rights management

## Users dialog

The currently registered users are listed in a tree structure. The ownerships of each user is displayed and each user is a member of a group by default.
※ Chapter 6.3.1.2.1 "User and access rights" on page 1409

## Define a new user account

1. Click "Add" to open the "Add User" dialog.
2. Define the user credentials and click [OK] to set up the new user. If there are incorrect entries (no login name, password mismatch, user already existing) you will get an appropriate error message.

Modify a user Click "Edit" to open the "Edit User" dialog. The entry fields are the same as in the "Add User" account dialog. The password fields however - for security reasons - will show 32 * characters. After having modified the desired entries close the dialog with [OK] to get applied the new settings.

[^28]
## Groups dialog

Add a group The currently available groups are displayed in a tree structure. A member also might be a group.

1. Click "Add" to open the "Add Group" dialog.
2. Define a name for the new group and enable all entries (single users or groups) which should be members of the new group.
3. Click [OK] to set up the new group. If there are incorrect entries (no name defined, group already existing, in Members having selected a group which would cause a "group cycle", you will get an appropriate error message.

Modify a group Click "Edit" to open the "Edit User" dialog. The entry fields are the same as in the "Add Group" dialog (see above). After having modified the desired entries close the dialog with [OK] to get applied the new settings.

Remove groups Enable the entries to be removed in the groups tree and click "Remove". Note that you will get no further inquiry! The members of the deleted groups will remain unmodified. An error message appears if you try to delete the groups "Everyone" and/or "Owner".

## Settings dialog

The following basic options and settings concerning the user accounts can be made:

- "Maximum number of authentication trials": If activated, the user account will be set invalid after the specified number of trials to log in with a wrong password. If not activated, the number of erroneous trials is unlimited. Default: option activated, number of trials: 3; permissible values: 1-10.
- "Automatically log out after time of inactivity": If activated, the user account will be logged out automatically after the specified number of minutes of inactivity (no user actions via mouse or keyboard registered in the programming system). Default: option activated, time: 10 minutes; permissible time values: 1-180 minutes.


### 6.3.1.3 Later change-over of a target system

### 6.3.1.3.1 Changing the processor module type

## Genera

In a project, you can change the target system by changing the type of processor module or terminal base type. If possible, the device configuration of fieldbuses and interfaces is kept and switched over to the device configuration of the new module.

Target change options:

- between platforms: from V2 platform to V3 platform (and vice versa)
- between module types: from AC500 (standard) to AC500-eCo (and vice versa)
- a combination of changed platform and changed module type


## Target change from a V2 processor module to a V3 processor module

Target change options:

- AC500 V2 processor module to AC500 V3 processor module
- AC500 V2 processor module to AC500-eCo V3 processor module
- AC500-eCo V2 processor module to AC500-eCo V3 processor module
- AC500-eCo V2 processor module to AC500 V3 processor module


## Procedure: 1. Close CODESYS.

2. Double-click the "PLC_AC500_V2 <...>" node and open the "PM5<...> Hardware" tab.
3. Enable "Change to AC500 V3 PLC" and select the desired V3 processor module from the "PM5xx Type" drop-down list.

4. Click [Create V3 PLC].
$\Rightarrow$ The new V 3 processor module is displayed in the navigation tree.
$\Rightarrow$ Change the node name of the processor module, if desired.


In case of a target change from AC500-eCo V2 to AC500-eCo V3, the I/O bus and Ethernet configuration is kept.

Target change from a V3 processor module to another V3 processor module
Target change options:

- AC500 V3 processor module to AC500 V3 processor module
- AC500 V3 processor module to AC500-eCo V3 processor module
- AC500-eCo V3 processor module to AC500 V3 processor module
- AC500-eCo V3 processor module to AC500-eCo V3 processor module

Procedure:

1. Close CODESYS.
2. Double-click the "PLC_AC500_V3 <...>" node and open the "PM5<...> Hardware" tab.
3. Select the desired V3 processor module from the "PM5xx Type" drop-down list.


Fig. 269: Change_Hardware_V3

4．Ensure the correct＂Terminal Base Type＂is selected and click［Change PM／TB type］．
$\Rightarrow$ If possible，the device configurations from the previous processor module will be kept and switched over to the new processor module．

The device configurations that cannot be kept are listed in a prompted information dialog．


By default，all device configurations which cannot be switched over will be copied to a＂device pool＂section in the navigation tree（option＂Copy all objects that cannot be added to the new PLC into a device pool for further access＂）．If required，this backuped configuration can be used in another project or in another processor module configuration．

If the checkbox is deactivated all device configurations that cannot be switched will be lost after the execution of the target change．


1 PLC＿AC500＿V3＿eco＿unplaced＿devices（Device Pool）
Ois OnBoard＿IO（12DI／6DO－R／2DC）
－ －Interfaces
蚟自 TA5120＿2AI＿UI（TA5120－2AI－UI）
目自 TA5126＿2AO＿UI（TA5126－2AO－UI）
－10）OnBoard＿RTC

Target change
The configuration of the onboard I／Os，the option board slots and the onboard RTC cannot be changed－over to the new module．

Target change from AC500 V3 to AC500－eCo V3


ETH1 configuration is kept even if the configured protocols are not allowed for the selected AC500－eCo V3 PLC．In this case error messages are displayed in the messages window．


#### Abstract

Libraries which are not used anymore are not deleted with the target change. Libraries of option boards are kept in the Library Manager even if no longer available at the target module.


### 6.3.1.3.2 Customer libraries

CODESYS for AC500 V2 products contains different types of libraries:

- Standard CODESYS libraries
- Specific AC500 libraries
- Customer libraries

In general, the Standard CODESYS libraries and the AC500 libraries are automatically converted during a target change from AC500 V2 to AC500 V3. Those libraries that cannot be converted (e. g. because there is no matching in V3) are created automatically in the V3 Library Manager and must be manually deleted by the user after the target change.
The customer libraries have to be converted manually using the Library Converter integrated into the Automation Builder installation:

1. In Automation Builder click "File $\rightarrow$ Open project".
2. Select the CODESYS library for AC500 V2 products which has to be converted.
3. After conversion of the library, open the view POUs in the device navigator and doubleclick "Project Information".
4. To have the library automatically available in the V3 project, enter "Company", "Title" and "Version" in the specific fields of the dialog.
Then, open the "Properties" tab. For the target change the new "Key" "CoDeSysV2Library" has to be added. Under "value", enter the name of the CODESYS library and click the "Add" button.


Click "File $\boldsymbol{\rightarrow}$ Save project" and install into the library repository.

### 6.3.1.4 Firmware identification and update

### 6.3.1.4.1 General information

Without direct access to the internet, a firmware update with the memory card is also possible.

* Chapter 6.3.1.4.5 "Firmware installation and update with memory card" on page 1426


### 6.3.1.4.2 Device state firmware version

## Introduction

As a process participant, whether as a user, machine builder, plant operator or maintenance engineer, it is helpful to be able to see at a glance which component has which firmware version.
If necessary, an error can then be quickly corrected or versions adjusted. The possible ways of displaying the various components are shown below.

- CPU
- Communication modules
- AC500-eCo option boards
- Field devices


## CPU firmware

## Version query via Automation Builder tab "Version information"

## Version information

Information on the firmware versions of the processor modules or communication modules, is provided on the "Version information" tab.
Remarks:

- The "Version information" tab displays the version identified on the device and the version provided with Automation Builder.
- The firmware on the devices must match to the Automation Builder supported versions. It might be required to upgrade or downgrade the current firmware for using the PLC with the current Automation Builder.
- The "Required Version" might contain more than one user selectable version (drop-down list), this is indicated by an asterisk after the file name.


State icons

| (8) | Firmware version on device matches version supplied with <br> Automation Builder. |
| :--- | :--- |
|  | Firmware version (or type) on device is different from version supplied <br> with Automation Builder. Upgrade/downgrade to version supplied with <br> Automation Builder is recommended. |
| No icon | Only for communication modules if CPU firmware must be updated <br> first. This happens when CPU firmware has version below 2.5.0.0. <br> Firmware version (or type) on device is different from version supplied <br> with Automation Builder. Upgrade/downgrade to version supplied with <br> Automation Builder is recommended. |
|  | Identified device is different from configured device, thus no firmware <br> update is possible. Happens only for Communication Modules. |
|  | Firmware of device is not updateable or no newer firmware than the <br> initial version is available. |

[Update Firmware] to download the new firmware is only enabled if there is updateable firmware.

## Version query via Automation Builder tab "PLC Shell"

The PLC shell is used for requesting specific information from the controller. By entering a device specific command the response is returned in a result window. The PLC shell can be issued without login.
Proceed as fol- 1. Ensure the gateway is configured properly and a connection to the controller can be lows: established.
2. In Automation Builder double-click the PLC node and open the tab "PLC Shell".
3. Enter versioninfo or rtsinfo in the command line of the tab window. The current firmware versions are listed.

## Version query via application

The version information are accessible via the function block PmVersion, production data accessible via function block PmProdReadAsync.
② Chapter 6.5.14 "Reference, function blocks" on page 4086

## Version query via IP configuration tool

IP configuration The IP configuration tool is part of Automation Builder and can be called via "Tools
tool
$\rightarrow$ IP-Configuration".

* Chapter 6.3.2.2.4.3.2 "Stand-alone installation" on page 1457

Further information on page 1459

## Communication modules

Version query via Automation Builder tab "Version information"
Chapter 6.3.1.4.2.2.1 "Version query via Automation Builder tab "Version information"" on page 1419

## Version query via Automation Builder tab "PLC Shell"

The PLC shell is used for requesting specific information from the controller. By entering a device specific command the response is returned in a result window. The PLC shell can be issued without login.
Proceed as fol- 1. Ensure the gateway is configured properly and a connection to the controller can be lows: established.
2. In Automation Builder double-click the PLC node and open the tab "PLC Shell".
3. Enter coupler desc in the command line of the tab window. The current firmware versions are listed.

## Version query via application

The version information are accessible in the property "Devicelnfo" of the module

## AC500-eCo V3 option boards

## Version query via Automation Builder tab "Version information"

Chapter 6.3.1.4.2.2.1 "Version query via Automation Builder tab "Version information"" on page 1419

Version query via Automation Builder tab "PLC Shell"
${ }^{*}{ }^{\star}$ Chapter 6.3.1.4.2.2.2 "Version query via Automation Builder tab "PLC Shell"" on page 1420

## Version query via application

The version information are accessible in the property "DeviceInfo" of the module.

## Field devices

Version query via Automation Builder
The firmware versions of the ABB communication interface devices for PROFINET (CI50xPNIO) and EtherCAT (CI51x-ETHCAT) is available from the respective editor in online mode in the Automation Builder.

## Version query via application

PROFINET, PROFIBUS, EtherCAT, CANopen: According to standard of the corresponding fieldbus protocol.

## Version query via IP configuration tool

« Chapter 6.3.1.4.2.2.4 "Version query via IP configuration tool" on page 1421

Only for Ethernet-based fieldbuses!

### 6.3.1.4.3 Installation and update of the AC500 V3 firmware

The multi download tool can be used to update multiple PLCs with firmware or an application at the same time.
ⓨ Chapter 6.6.4 "Multi download tool" on page 4306

Firmware update with activated user management

The PLC user management will remain also after a firmware update or downgrade.
Due to a structural change in the PLC user management in firmware version 3.5 there are some limitations when updating the PLC firmware from a version 3.4 or lower to 3.5 or higher with activated user management.

If you update the PLC firmware from version 3.4 or lower to version 3.6 or higher this must be done in two steps. First an update to 3.5 must be done via Automation Builder. Afterwards the firmware can be updated to any later firmware version - either via Automation Builder or via memory card.

The PLC firmware can be updated via Automation Builder.


A very new CPU has no pre-installed firmware. To guarantee the authenticity of delivered AC500 firmware, V3 CPUs are delivered with a boot loader only. You need to download a valid firmware to the CPU. After download, the functionality of the CPU is given.
$\checkmark$ An Automation Builder project with an AC500 V3 CPU is open.
$\checkmark \quad$ CPU is in "stop" mode or shows uPdAtE (update) on the display.
$\square \quad$ After update the CPU shows either donE or StoP on the display
$\square$ For new modules: IP address is set. (The default IP address is 192.168.0.10)

1. Double-click CPU "PLC_AC500_V3".
2. Select "Version information".


## 3. Select "Update Firmware".

$\Rightarrow$ While the update process is running, the RUN and ERR LEDs are toggling, i.e., they are flashing alternating.
4. Wait for the PLC to finish the update.

## $\stackrel{9}{9}$ <br> Firmware updates that include "UpdateFW" changes must be performed in two steps:

First, the "UpdateFW" is updated, and once this is done click the update button again to execute the "SystemFW" update.

Note: Firmware updates via memory card automatically cover both updates in one step.

A completed update is indicated by a message on the display. Either donE, or StoP.

## NOTICE!

Do not disconnect the power supply during the update process! The PLC could be damaged.
$\Rightarrow$ StoP indicates a restart has been performed by the CPU. When donE is displayed sometimes it is necessary to re-boot the CPU manually, e.g., by powering-off. Manual re-boot might be, e.g., for some older CPU versions or if downgrading to an older firmware version according to application settings.


The CPU display shows StoP after re-boot. The update process is finished.
5. If necessary, refresh the version information by switching to another tab and back. $\Rightarrow$ Successful firmware update:


Behavior of LEDs during firmware update


| LED | LED flashes | Status |
| :--- | :--- | :--- |
| RUN and ERR | Toggling | Update pending |
| RUN | Flashing slow | Done successful |
| ERR | Flashing slow | Done failed |

### 6.3.1.4.4 AC500-eCo V3 firmware installation and update

The multi download tool can be used to update multiple PLCs with firmware or an application at the same time.

② Chapter 6.6.4 "Multi download tool" on page 4306

The PLC firmware can be updated via Automation Builder.

This is also necessary for commissioning AC500-eCo V3 CPUs.

A very new CPU has no pre-installed firmware. To guarantee the authenticity of delivered AC500-eCo firmware, V3 CPUs are delivered with a boot loader only. You need to download a valid firmware to the CPU. After download, the functionality of the CPU is given.

An Automation Builder project with an AC500-eCo V3 CPU is open.
CPU is in "stop" mode without firmware.The power LED is ON.
$\checkmark \quad$ For new modules: IP address is set. (The default IP address is 192.168.0.10)

1. Double-click CPU "PLC_AC500_V3".
2. Select "Version information".
ate Project1.project** Automation Builder 2.4-Premium
File Edit View Project Build Online Debug Tools BACNet Window Help
目

3. Select [Update Firmware].
$\Rightarrow$ While the update process is running, the RUN and ERR LEDs are toggling, i.e., they are flashing alternating.
4. Wait for the PLC to finish the update.

## NOTICE!

Do not disconnect the power supply during the update process! The PLC could be damaged.
5. If necessary, refresh the version information by switching to another tab and back. $\Rightarrow$ Successful firmware update:


Behavior of LEDs during firmware update

- CPU without firmware, only the power LED is on.
- While the firmware update process is running, the RUN and ERR LEDs are toggling, i.e., they are flashing alternating.

| LED | LED flashes | Status |
| :--- | :--- | :--- |
| RUN and ERR | Toggling | Update pending |
| RUN | Flashing slow | Done successful |
| ERR | Flashing slow | Done failed |

- CPU with installed firmware, only the power LED is on.
- If the CPU is running, then the RUN LED is on.
- If the CPU is in STOP mode, the RUN LED is off.


### 6.3.1.4.5 Firmware installation and update with memory card

## General

The memory card is a removable persistent mass storage device and can be used for any application purpose. Both firmware updates and boot project updates can be run from the memory card.

## NOTICE!

## Removal of the memory card

Do not remove the memory card when it is working. For memory card activity the black square ( $\square_{\text {) is shown on PLC display as long as a file is open on }}$ the memory card. Remove the memory card only when no black square ( $\square$ ) is shown next to memory card in the display. Otherwise the memory card and/or files on it might get corrupted and/or normal PLC operation might be disturbed.

Firmware and/or application installation and update Notes

Firmware update with activated user management
$A B B$ recommends that users carry out the firmware update via Automation Builder. \& Chapter 6.3.1.4 "Firmware identification and update" on page 1419
Not every user has an Automation Builder at his disposal with which a firmware or boot project update can be easily realized. In this case, the user must be provided with a prepared memory card from his client.

It is not possible to update the communication interface modules with a memory card. The firmware of the communication interface modules can only be updated with the IP configuration tool.

The PLC user management will remain also after a firmware update or downgrade.
Due to a structural change in the PLC user management in firmware version 3.5 there are some limitations when updating the PLC firmware from a version 3.4 or lower to 3.5 or higher with activated user management.

If you update the PLC firmware from version 3.4 or lower to version 3.5 this must be done via Automation Builder. An update via memory card with activated user management will result in losing all access rights to the PLC.

If you update the PLC firmware from version 3.4 or lower to version 3.6 or higher this must be done in two steps. First an update to 3.5 must be done via Automation Builder. Afterwards the firmware can be updated to any later firmware version - either via Automation Builder or via memory card.

* Chapter 6.3.1.4.3 "Installation and update of the AC500 V3 firmware" on page 1422


## Preparation of memory card

② MC5141-memory card
MC5102-micro memory card with adapter

Firmware packages can be downloaded from $A B B$ website or exported from Automation Builder.

In Automation Builder, you can choose if you want to export the firmware only or, in addition, the boot project as well. If the boot project does not exist yet, it will be created automatically.
The memory card will contain the firmware of the PLC, the firmware of the communication modules, the application, the visualizations and all related objects (like text lists).


Fig. 270: Example

1. Right-click "Application" in the device tree.
2. Select "Export $\rightarrow$ Boot project and firmware" or "Export $\rightarrow$ Firmware".
3. Choose a location for the export folder.

Select [Make New Folder] and type in "MEMORY CARD".
4. Select $[O K]$ to add the folder.
5. Select folder "MEMORY CARD".

$\Rightarrow$ The "MEMORY CARD" structure has been created and the firmware and application have been exported.


The created "MEMORY CARD" folder does not contain user data, remanent data, config data, safety PLC power dip data and safety PLC password.
Add this data if required by the used application.
6. Copy all subfolders and files of the "MEMORY CARD" folder to a memory card.

The copy process can be performed with the Automation Builder software with the memory card inserted in the processor module or with the PC with a suitable memory card slot.

Copy process with Automation Builder software

1. Insert memory card into the processor module.
2. Connect the PC to the processor module with an Ethernet cable.
3. In Automation Builder, log-in to the processor module.

4. Double-click on the processor module in the device tree and select "Files".
5. On the right side, select and open the folder "sdcard".
6. On the left side, select all the subfolders and files of the MEMORY CARD folder, but not the folder itself.
7. Copy all selected subfolders and files from the left side to the memory card by clicking $\square \gg$ to the right side.
8. Insert the memory card into the PC.

Copy process on the PC

2. Mark all subfolders and files of the "MEMORY CARD" folder, but not the "MEMORY CARD" folder itself, and copy them.


5 This PC

- My Passport (D:)


## 蔵SDHC (E:)

3. Select the memory card and paste the copied subfolders and files.

## Execution of update via memory card

Boot project and firmware update

Precondition: Prepared memory card with boot project and firmware ${ }_{4}{ }^{4}$ Chapter 6.3.1.4.5.2.2 "Preparation of memory card" on page 1427.

1. Switch off the device.
2. Insert the memory card.
3. Switch on the device.
$\Rightarrow$ The alternate flashing of the RUN and the ERR LED indicates the running update process.

At the end of the update process a reboot is executed and the boot project and system firmware is started for the finishing of the update process.
If RUN LED blinks (ERR LED is off), the update was successful and the display shows "done".

If ERR LED blinks (RUN LED is off), the update failed and the display shows "FAIL".
The text file "SDCARD.RDY" includes the results of the different updates. If the update fails, the file contains the reasons for the abort. Based on this, further steps can be taken to fix the problem.
4. Switch off the device.
5. Remove the memory card.
6. Switch on the device.
$\Rightarrow$ The system starts with the new boot project and firmware.

| Firmware |  |
| :--- | :--- |
| update | Precondition: Prepared memory card with boot project and firmware ${ }^{4} \Rightarrow$ Chapter |
| 6.3.1.4.5.2.2 "Preparation of memory card" on page 1427. |  |

1. Switch off the device.
2. Insert the memory card.
3. Switch on the device.
$\Rightarrow$ The alternate flashing of the RUN and the ERR LED indicates the running update process.

At the end of the update process a reboot is executed and the system firmware is started for the finishing of the update process.

If RUN LED blinks (ERR LED is off), the update was successful and the display shows done.

If ERR LED blinks (RUN LED is off), the update failed and the display shows FAIL.
The text file "SDCARD.RDY" includes the results of the different updates. If the update fails, the file contains the reasons for the abort. Based on this, further steps can be taken to fix the problem.
4. Switch off the device.
5. Remove the memory card.
6. Switch on the device.
$\Rightarrow$ The system starts with the new firmware.

## Description of LEDs

The LEDs below the display indicate the state of the processor module:

| LED | State | Color | LED $=0 \mathrm{~N}$ | LED = OFF | LED flashes |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Power LED (PWR) | Denotes the power supply state of the processor module | Green | Voltage is present (24 V DC) | Voltage is missing |  |
| Run LED (RUN) | Denotes the activity state of the processor module | Green | Processor module is in RUN mode | Processor module is in STOP mode | If the LED flashes fast (4 Hz ) a firmware update is finished with no errors. <br> If the Run LED flashes fast $(4 \mathrm{~Hz})$, alternating with a flashing Run LED the firmware is updated. <br> To enforce boot mode 1, keep the RUN function key pressed during the boot procedure. In this case, the Run LED flashes slowly ( 1 Hz ). A subsequent project download (from within Automation Builder) cancels the blinking. |
| $\begin{array}{\|l\|} \hline \text { Error LED } \\ \text { (ERR) } \end{array}$ | Denotes an error | Red | An error has occured. | No errors or only warnings have occurred | If the Error LED flashes slowly ( 1 Hz ) a firmware update from the memory card is finished with errors. <br> If the Error LED flashes fast with AC500 on display a fatal system error has occurred. <br> If the Error LED flashes fast ( 4 Hz ) alternating with a flashing Run LED the firmware is updated. |

A running processor module is indicated with the state RUN on the display, a deactivated processor module is indicated with the state STOP. In both cases the display's backlight is off.

## Advanced memory card handling for firmware/application update

Memory card file content: Firmware version V3.x

Only advanced users should apply the instructions in this chapter.

General update process

② Chapter 6.3.1.4.3 "Installation and update of the AC500 V3 firmware" on page 1422
The main components of the V3 CPU firmware are:

- BootFW (boot firmware): responsible for the starting of the UpdateFW or the SystemFW
- UpdateFW (update firmware): responsible for the update of BootFW, UpdateFW, SystemFW, UpdateHook and boot project
- SystemFW (system firmware, CPUFW): Runtime system of the PLC, additionally responsible for the update of the DisplayFW (display firmware) and the firmware of the communication module
- DisplayFW (display firmware): Firmware of the display

Additionally the update process includes the following parts:

- Communication module (communication module firmware): Firmware of the different communication module
- UpdateHook: Specific patches for the PLC
- UserProgram: Boot project of the application
- License features: Import and export of license files. The license file for the "ImportLicense" is a Wbb or a WibuCmRaU file. The license file of the "ExportLicense" is a WibuCmRaC file.

The firmware updates are triggered by the command file SDCARD.INI. In addition a result file of the firmware update is generated (SDCARD.RDY, identical path as SDCARD.INI). For the group [FirmwareUpdate] the parameters $0,11,12$ and 13 are defined. For each firmware update two files are necessary. The firmware file and the corresponding signature file.
For example:
AC500_V3_SystemFirmware_V3.0.1.73.tar.bz2
AC500_V3_SystemFirmware_V3.0.1.73.tar.bz2.sig
AC500_V3_DisplayFirmware_V3.0.0.0.app
AC500_V3_DisplayFirmware_V3.0.0.0.app.sig
For the user program the application file and the application $C R C$ are necessary. For example:

## Application.app

## Application.crc

If the signature file and the firmware file do not match, no update is performed and the correspondent error result is written to the file SDCARD.RDY.
If the update firmware is running the display shows the text update. The blinking of the RUN and the ERR LED's indicates the update process.
The file "SDCARD.RDY" includes the results of the different updates. After an update of a communication module CODESYS Control is started in safe mode (no download or starting of the application is possible) and the PLC needs a reboot (power down/up; the display shows please and reboot alternately).

As of system firmware 3.2 the compatibility file "Version.txt" (with the corresponding signature file "Version.txt.sig", identical path as "SDCARD.INI") is necessary for the update process. The update firmware checked the compatibility of the following parts:

- CPUFW (system firmware)
- BootFW (boot firmware
- UpdateFW (update firmware)
- DisplayFW (display firmware)

A missing "Version.txt" or a missing/corrupt "Version.txt.sig" file is signalled at the component "CPUFW" (file "SDCARD.RDY").
If the update process would result in incompatible parts of firmware no update is performed in the update firmware. After starting of the system firmware the compatibility of the communication module firmware is checked additionally. The check of the compatibility of the firmware is executed always (independent of the parameter for the component). Incompatibility is signalled at the corresponding component (file SDCARD.RDY").

## Automated reboot of the PLC

If a firmware or boot application update is currently being performed via an memory card, the PLC will wait for the user. To remove the memory card and do a power cycle for completing the update process.
In the case of an unattended remote update via memory card, this is an obstacle because there may be no one on site to perform the power cycle. To avoid this situation, additional options have been added to SDCARD.ini to enable an unattended automatic restart when entering RTS safe mode after a successful firmware or boot project update.

The changes are an optional behavior that is triggered by the user setting in SDCARD.ini! The default behavior (RTS stays in safe mode until manual power cycle) is maintained!

The "FunctionOfCard" is reset from (4-6) to 0 after the update. Which means after a successful update operation the memory card cannot be used for updating another PLC without manual modification of the "FunctionOfCard".
The intended use case is for the memory card to always remain in the respective PLC to be used as mass storage and for the unattended update.

In case of changed boot parameters, an additional reboot is required. This additional reboot is never done automatically and herefore it has to be done manually.

## Command file SDCARD.INI for AC500 V3 Products

Only advanced users should apply the instructions in this chapter.

| [FirmwareUpdate] | 0 = No update |
| :---: | :---: |
| $\begin{aligned} & \text { CPUFW=x } \\ & x=0,11,12,13 \end{aligned}$ | 11 = Update system firmware always with the file specified in module's section [CPU] and component's path key "CPUFW". <br> $12=$ Update with different version, the update is only performed if the version of the file specified by the component path key "CPUFW" in module's section [CPU] differs from the current version of the CPU. <br> 13 = Update with newer version, the update is only performed if the version of the file specified by the component path key "CPUFW" in module's section [CPU] is newer than the current version of the CPU. |
| $\begin{aligned} & \text { BootFW=x } \\ & x=0,11,12,13 \end{aligned}$ | See description CPUFW. The component's path key for the boot firmware in module's section [CPU] is "BootFW". |
| UpdateFW=x $x=0,11,12,13$ | See description CPUFW. The component's path key for the update firmware in module's section [CPU] is "UpdateFW". |
| $\begin{aligned} & \hline \text { DisplayFW=x } \\ & x=0,11,12,13 \end{aligned}$ | See description CPUFW. The component's path key for the display firmware in module's section [CPU] is "DisplayFW". |
| UpdateHook=x $x=0,11$ | 11 = Execute UpdateHook always with the file specified in module's section [CPU] and component's path key "UpdateHook". |
| ImportLicense=x $x=0,12$ | $12=$ Import the license always with the file specified in module's section [CPU] and component's path key "ImportLicense". The license file is a Wbb or a WibuCmRaU file. The update process imports this file into the plc. <br> Note: Do not use parameter 11 for license import. |
| $\begin{aligned} & \text { ExportLicense=x } \\ & x=0,12 \end{aligned}$ | 12 = Export the license always to the file specified in module's section [CPU] and component's path key "ExportLicense". The exported license file is a WibuCmRaC file. <br> Note: Do not use parameter 11 for license export. |



## Example: SDCARD.INI as of CPU firmware V3.x

Only advanced users should apply the instructions in this chapter.
[Status]
\# FunctionOfCard $\$$ FocValue (0, 1, 2, 3, 4, 5, 6) from SDCARD.INI
\# 0 = No update
\# 1: Load user program according to entry in group [UserProg] and the PLC prompts for manual reboot.
\# 2: Start firmware update according to entry in group [FirmwareUpdate] and the PLC prompts for manual reboot.
\# 3: Update firmware according to entry in group [FirmwareUpdate] and load user program according to entry in group [UserProg] and the PLC prompts for manual reboot.
\# 4: Load user program according to entry in group [UserProg] and the PLC reboots automatically.
\# 5: Start firmware update according to entry in group [FirmwareUpdate] and the PLC reboots automatically.
\# 6: Update firmware according to entry in group [FirmwareUpdate] and load user program according to entry in group [UserProg] and the PLC reboots automatically.
[FirmwareUpdate] ;
$0=$ No update ;
11 = Update with file specified in module's section <modsec>, component's path key <pathkey> ;
$12=$ Like 11, but check version of file to be updated differs from current one. ;
13 = Like 11, but check version of file to be updated is newer than current one.
CPUPFW=0; ;modsec>=|CPU|, <pathkey>= CPUPFW
BootFW=0;<modsec>=|CPU|, <pathkey>= BootFW
UpdateFW=0;<modsec>=|CPU|, <pathkey>= UpdateFW
DisplayFW=0;<modsec>=|CPU|, <pathkey>=DisplayFW
UpdateHook=0;<modsec>=|CPU|, <pathkey>=UpdateHook
ImportLicense=0;<modsec>=|CPU|, <pathkey>=ImportLicense
ExportLicense=0;<modsec>=|CPU|, <pathkey>=ExportLicense
Coupler0=0;<modsec>=|Coupler0|, <pathkey>=Firmware
Coupler1=0;<modsec>=|Coupler1|, <pathkey>=Firmware
Coupler2=0 ;<modsec>=|Coupler2|, <pathkey>=Firmware
Coupler3=0;<modsec>=|Coupler3|, <pathkey>=Firmware
Coupler4=0;<modsec>=|Coupler4|, <pathkey>=Firmware
Coupler5=0;<modsec>=|Coupler5|, <pathkey>=Firmware
Coupler6=0 ;<modsec>=|Coupler6|, <pathkey>=Firmware
[UserProg] ;
$0=$ No update ;
11 = Update with file specified in module's section <modsec>, component's path key <pathkey>
UserProgram=0 ;Update user program. <modsec>=[CPU], <pathkey>= UserProgram
[CPU];
CPUFW= ;Path/file of CPU's system firmware to update
BootFW= ;Path/file of CPU's boot firmware to update
UpdateFW= ;Path/file of CPU's update firmware to update
UpdateHook $=;$ Path/file of UpdateHook to update
DisplayFW= ;Path/file of Display's firmware to update
ImportLicense= ;Path/file of import license file
ExportLicense= ;Path/file for export license file
UserProgram= ;Path/File of user program to update
[Coupler0]
Firmware= ;Path/file of internal coupler's firmware to update
[Coupler1]
Firmware $=$;Path/file of external coupler's firmware slot 1 to update
[Coupler2]

Firmware $=$;Path/file of external coupler's firmware slot 2 to update
[Coupler3]
Firmware $=$;Path/file of external coupler's firmware slot 3 to update [Coupler4]

Firmware= ;Path/file of external coupler's firmware slot 4 to update [Coupler5]

Firmware $=$;Path/file of external coupler's firmware slot 5 to update [Coupler6]

Firmware $=$;Path/file of external coupler's firmware slot 6 to update

| Example con- <br> tent and <br> description of <br> the SDCARD.INI <br> folder | SDCARD.INI for memory card for update only the system firmware (SystemFW): |
| :--- | :--- |
|  | FunctionOfCard=2 |
|  | [FirmwareUpdate] |
|  | CPUFW=11 |
|  | [CPU] |
|  | CPUFW=/SystemFirmware/AC500_V3_SystemFirmware_V3.7.0.99_zzz.tar.bz2 |

### 6.3.1.4.6 Update $\mathrm{Cl} 52 x-M o d b u s$ firmware

## Precondition

Requirement: A firmware update file is available, e.g. AC500_CI52x_Firmware_V3.2.8.bin.

The CI52x Modbus firmware update is only available in the Automation Builder IP Configuration Tool.

## Installation of the IP configuration tool

1. In Automation Builder click "Tools $\rightarrow$ Installation Manager" to start the Installation Manager.
2. Close any other running instances of Automation Builder. Then, click "Modify" in the Installation Manager.
3. Select the option "IP Configuration Tool" from the list and start the installation of the IP Configuration Tool.

## Firmware update procedure

1. In the IP Configuration Tool click "Scan" to initialize a device scan.
2. From the list select the CI52x-MODTCP device(s) which shall be updated and click "FW Update".
3. Select the firmware update file (e.g. AC500_CI52x_Firmware_V3.2.8.bin) to initialize a signature check and start the update procedure.

During the firmware update the operation of the device(s) is stopped. After the update, all outputs are set to zero.
4. After the update, click "Scan" again to retrieve the firmware version of the device.

## Troubleshooting

| Fault description | Cause | Remedy |
| :---: | :---: | :---: |
| Cl firmware update failed |  | General hints <br> - Close all unused applications on the update PC and do not open Automation Builder or any other applications during the firmware update. <br> - Stop the communication between AC500 PLC and the CI52x devices and disconnect the Ethernet connection of the update PC and the Cl Modbus device(s). <br> - Do not close the IP Configuration Tool during a firmware update and do not switch off a Cl Modbus device during the firmware update. |
| Error 1: <br> Package <br> Timeout | Due to a primitive firmware update protocol a fast and stable network connection is required. Otherwise the update packages cannot be transferred within the requested time and a timeout occurs. | Locate the PC on which the update is performed as near as possible to the stationed Cl Modbus devices. Avoid network switches. |
| Error 2 Unable to read device status | After the firmware update the IP Configuration Tool reads out the status of the updated device in order to check if the update was successful. | Rescan and repeat the update. If this doesn't work, power cycle the device and retry the update. |
| Error 3: IP is not unique | If more than one device hold the same IP address, a firmware update is not possible as the update command is IP based. | Correct the IP address, rescan and repeat the update. If this doesn't work, power cycle the device and retry the update. |
| Error 4: Internal Error | An internal error on the Cl52x Modbus device occurred during the firmware update. | Rescan and repeat the update. If this doesn't work, power cycle the device and retry the update. |
| Error 5: Cannot connect to device | The TCP communication is not sufficient for a connection. Increase the connection quality. | Locate the PC on which the update is performed as near as possible to the stationed Cl Modbus devices. Avoid network switches. |
| Signatur check failed | After the selection of the firmware file (*.bin) a signature check is performed. If either the firmware file or the signature file is corrupt, the signature check fails. | Example file names: <br> - Name of the firmware file: c:IAC500\AC500_CI52x_Firmware_V3.2.8.bin <br> - Correct name of the signature file: c: $\mathrm{IAC500} \mathrm{\backslash AC500} \mathrm{\_CI52x} \mathrm{\_Firmware} \mathrm{\_V3.2.8.bin.sig}$ <br> - Wrong name of the signature file: <br> c:IAC500\AC500_CI52x_Firmware_V3.2.8.sig |


| Fault description | Cause | Remedy |
| :---: | :---: | :---: |
| Indeterminate device firmware version | If the firmware version of the device cannot be determined, an error occurs. | Example: Result OK <br> - PC: 192.168.14.71 / 255.255.255.0 <br> - Device: 192.168.14.10 / 255.255.255.0 <br> Example: Result OK <br> - PC: 192.168.10.71 / 255.255.0.0 <br> - Device: 192.168.14.10 / 255.255.0.0 <br> Example: Result ERROR <br> - PC: 192.168.10.71 / 255.255.255.0 <br> - Device: 192.168.14.10 / 255.255.255.0 |

### 6.3.1.5 Migration of third party devices

After an update of Automation Builder the device repository contains only ABB devices. The third party devices which were installed into previous versions of Automation Builder are not automatically installed in the newest version profile. This has to be triggered by the user.


The feature "Migrate third party devices" is available as of Automation Builder 2.1.1.

1. Click "Tools" in the main menu of Automation Builder.
2. Click "Migrate third party devices" in the drop-down list.
$\Rightarrow$ The window "Version profile selection" appears.
3. Select a version profile in the drop-down list containing previous Automation Builder / Control Builder Plus profiles. The active profile does not appear in the list.
$\Rightarrow$ After selection of a previous version profile, all the third party devices which have been installed inside this version profile are listed.


It is not possible to select or deselect some third party devices. Importing will affect all the third party devices which are listed in the list view.
4. Select [Import].

$\Rightarrow$ During the migration the message window displays success or failure of device migration.


In case of failure during the migration the affected third party device description has to be installed manually via main menu "Tools
$\rightarrow$ Device Repository $\rightarrow$ Install".

In the status bar, the third party device which is on "Migrating: <...>" is displayed on the left side.

The import operation can be cancelled by clicking the "Click here to CANCEL this operation" link on the right side of the status bar. This becomes effective when the migration of the just migrating third party device is finished.
5. To close the dialog select the [Close] button of the Version profile selection.

### 6.3.1.6 Advanced IO device handling

### 6.3.1.6.1 General

Automation Builder provides the Advanced IO Device Handling feature for configuring identical IO device types at multiple instances.
This feature is supported by the following commands that works with IO devices only.

- Generate DUT

² Chapter 6.3.1.6.2 "Generating DUT" on page 1442

- Map to Existing DUT
\# Chapter 6.3.1.6.3 "Mapping to existing DUT" on page 1442
- Release DUT mapping
* Chapter 6.3.1.6.4 "Releasing DUT mapping" on page 1442

These commands work on individual nodes and on Cl level nodes.

### 6.3.1.6.2 Generating DUT

Each device generates two DUTs. One for the input and one for the output. Some devices contain only input or output type. In such cases, the device generates only one DUT of the relevant type.

- Right-click on the desired IO device and select "Generate DUT" to generate a DUT for an IO device.

The following example shows how to generate DUTs at Cl level node.

- In the device tree, right-click on a master node such as PNIO_Controller and select "Generate DUT" to create DUTs for the child nodes.
- The DUTs of child nodes are generated in "Application $\rightarrow$ App $\rightarrow$ IO_Device_Generated_Items" folder.
- Generated DUT considers channels with BYTE datatype as members. If channels with BYTE datatype are not present in the given hierarchy, it adds the members with another higher datatype.
- Channels with BOOL datatype are not considered.


### 6.3.1.6.3 Mapping to existing DUT

This command is enabled for the IO device when the IO device is not mapped and when DUTs of matching size (calculated based on device channel list) are available in "Application $\rightarrow$ App $\rightarrow$ IO_Device_Generated_Items" folder.

1. Right-click on an IO device and select "Map to Existing DUT".
$\Rightarrow$ Enter Instance Name dialog is displayed.
2. Enter the instance name which satisfies IEC naming validations and unique name in global scope.
3. Click [OK] to create a global variable associated with the mappings in DI (PRG).

If you want to view mapped instances, double-click "DI (PRG)".

With the "Map to Existing DUT" command:

- Any device can be mapped only to one input DUT and one output DUT. If you have already mapped an input DUT, only the output DUT is shown in the options list and vice-versa.
- Mapping is also supported at Cl level nodes. To create global variables for Cl level nodes, the address of the first child is considered.


### 6.3.1.6.4 Releasing DUT mapping

This command is enabled on an IO device only when an IO device is mapped either to input, output or both DUTs. You can use this command to release (or revert) mappings and to delete global variables created during 'Map to Existing DUT'.
Right-click on an IO device and select "Release DUT Mapping". The mapped DUT instance is deleted.

### 6.3.1.6.5 Using DUT variables in CODESYS application

1. In the Automation Builder project, double-click "Application" to launch CODESYS application.
$\Rightarrow$ CODESYS application is launched. CODESYS application contains mapped DUT instances.
2. Double-click "PLC_PRG" to create DUT variables.
3. Add DUT variables based on mapped DUTs.

For further information on mapping DUTs, see section ${ }^{\mu}$ Chapter 6.3.1.6.3 "Mapping to existing DUT" on page 1442.
For example, in the PLC_PRG, add analog I/O and digital I/O. If you insert a dot at a position where an identifier should be inserted, then a selection list is open, offering all the input and output variables which are found in the project.
After adding DUT variables, rebuild the program in CODESYS application using "Project
$\rightarrow$ Rebuild".

### 6.3.1.6.6 Support for Cl level node

The user can create DUTs for the entire hierarchy of Cl level node (for example, IO_BUS), by right-clicking on the desired Cl level node and by selecting "Generate DUT". Further, all the DUTs are generated in "Application $\rightarrow$ App $\rightarrow$ IO_Device_Generated_Items" folder.

- The command generates DUT for the node itself and also for all child nodes.
- The DUT generated for the CI level node contains generated DUTs for the child nodes as their members.
- For every execution, the command checks, if any new child node is added and generates DUT.
If you delete child nodes in Cl level node (for example, IO_BUS), the DUTs generated for these child nodes are not deleted automatically. You should delete the DUTs manually in the "Application $\rightarrow$ App $\rightarrow$ IO_Device_Generated_Items" folder if desired.


### 6.3.1.6.7 Configuration check

Configuration check for size is enabled to ensure that all devices are mapped with DUTs of the correct size. In case of any changes in the mapped DUT, configuration check verifies the size of the DUT. If it fails, an error message is displayed in Automation Builder messages window and does not allow to launch the application. This check can be performed in "Create configuration data".

### 6.3.1.6.8 Flexible Configuration

For improved maintenance AC500 V3 PLCs provide the functionality to handle several S500 devices with a flexible configuration. The application example AC500 V3 - Flexible Configuration demonstrates how to configure a flexible configuration scenario in AC500 V3 PLCs.

### 6.3.2 PLC devices and components

### 6.3.2.1 Device repository

The Device Repository of Automation Builder manages the pool of devices that can be used in the PLC configuration.
You install or uninstall devices in the "Device Repository" dialog box. The system installs a device by reading the device description files, which define the device properties for configurability, programmability, and possible connections to other devices.
You can use the devices provided in the device repository by adding them to the device tree of your project.

## Dialog device

 repository1. Click "Tools $\rightarrow$ Device Repository".
$\Rightarrow$ The "Device Repository" dialog box opens.

[Edit Locations]: Changes the default repository location. The devices can be managed at different locations.
[Install] / [Uninstall]: Installs or uninstalls devices.
[Renew device repository]: Updates the device list, e.g. after uninstallation of a device.
[Details]: Provides technical details on the selected device.
2. Select the install location. "System Repository" is set by default.

Installing devices

The device repository cannot be changed manually, e.g. by copying or deleting files. Use always the "Device Repository" dialog to add or remove devices.

1. Click [Install] and select the appropriate file format.
$\Rightarrow$ The "Install Device Description" dialog box opens.

2. Select the file path of the device description.
3. Select the file type filter of the required device description.
$\Rightarrow$ All device descriptions of the selected file type are listed.
4. Select the required device description and click [Open].
$\Rightarrow$ Automation Builder adds the device description to the matching category of your device repository.
If errors occur during installation (for example, missing files that are referenced by the device description), then Automation Builder displays them in the lower part of the "Device Repository" dialog box.

During the installation the device description files and all additional files referenced by that description will be copied to an internal location. Altering the original files will have no further effects to an internal location.

The changes take only effect after reinstalling the corresponding device(s).
The version number shown in the information section of the device should be verified.

## Uninstalling Select the device you want to remove and click [Uninstall]. devices The device is removed from the list.

Uninstalled devices which are used in existing projects are indicated by the symbol ?. The device will not be configured properly.

### 6.3.2.2 PLC start-up

### 6.3.2.2.1 Initialization of AC500 V3 CPU

To initialize an AC500 V3 CPU, you need to download the firmware.
A new CPU has no pre-installed firmware, it is delivered with a boot loader only. You need to download a valid firmware to the CPU. See ${ }^{\mu}$ chapter 6.3.1.4.3 "Installation and update of the AC500 V3 firmware" on page 1422. After download, the functionality of the CPU is given.

### 6.3.2.2.2 PLC runtime licensing

## Introduction

The use of some libraries and devices require the PLC to have a runtime license. If you have purchased such a license, activate the license.
${ }^{\Downarrow}$ Chapter 6.3.2.2.2.2 "Activating a runtime license via license key" on page 1446
If you want to test device functionality or library features in advance, you can activate a demo license in advance.
*y Chapter 6.3.2.2.2.3 "Activating a demo license" on page 1450
The license status of a PLC can be displayed at any time.
② Chapter 6.3.2.2.2.6 "View license information" on page 1453

Activating a run- The linked application note provides step-by-step instructions on how to activate the runtime time license license for the PLC.

AC500 V3 Activating a runtime license

## Activating a runtime license via license key

 GeneralThe use of some libraries and devices require the PLC to have a runtime license.
$\checkmark \quad$ PC and PLC are connected. In case of no connection, perform the activation via memory card ${ }^{\mu} \Rightarrow$ Chapter 6.3.2.2.2.4 "Licensing via memory card" on page 1450.
$\square$ There is a connection to the Internet. In case of no connection, perform the activation on another PC with internet connection $\Longleftrightarrow$ Chapter 6.3.2.2.2.2.2 "Activation without internet connection" on page 1447.

1. Right-click on the PLC and select "PLC runtime licensing" from the "Runtime Licensing" menu.
$\Rightarrow$ A wizard starts. Follow the instructions.
2. Enter the license activation key and select "Next" to finish the licensing procedure.
$\Rightarrow$ The license is activated on the PLC device.
If the license shall be used on another PLC device, the installed license can be returned ${ }^{\mu}$ Chapter 6.3.2.2.2.5 "Returning a license" on page 1452.

## Activation without internet connection

1. If an error occurs when communicating with the ABB license server, or if Automation Builder is running on a PC without internet connection, then it is possible to manually complete the ABB license server interaction by using another PC (with internet connection).


Could not establish a connection to License Central to send the license request.
Click 'Retry' to try again or click 'Next' to save the license request so it can be manually uploaded.
2. In the error dialog select [Next] and save the license activation request file to a storage location the other PC can access, e.g. a file share.
3. In the dialog the web address of the ABB license server is displayed (http:///lc.codemeter.com/32838/depot/index.php). From the PC with internet connection, upload the license activation request file.


The file has been saved to
C:IUsersiTestiDocuments|Automation Builder/PLC AC500 V3 ActivationRequest.xml
Please continue the activation process by uploading the file to
http://c.codemeter.com/32838/depotindex.php


## Automation Builder - Offline Activation

Upload activation request

```
Activating your licenses offline - First step "Upload activation request":
1. Create an activation request flie with the Automation Bulder Activation Wizard.
2. Pick the created activation request file.
3. Click "Upload request and continue"
```

```
Pick activation request file (*.xml)
```

$\square$ Search

## Upload request and continue

4. After the upload, download and save the license activation file from the ABB license server. Transfer this file to the PC without Internet connection.
5. Select [Next] to continue the license activation process. Click [Cancel] to continue the license activation process at a later time $\Longleftrightarrow$ Further information on page 1449.
6. Select [Browse] and select the license activation file (*.WibuCmRaU) from the defined storage location.
$\Rightarrow$ The license is validated by the ABB license server and afterwards activated on the PLC device.

If the license shall be used on another PLC device, the installed license can be returned " $\triangleleft$ Chapter 6.3.2.2.2.5 "Returning a license" on page 1452.
7. To complete the licensing process, a license receipt file must be uploaded to the ABB license server.
Save the license receipt file and upload it manually from a PC with internet connection to http:///lc.codemeter.com/32838/depot/index.php.
$\Rightarrow$ A license confirmation is returned.

```
HOME
English ~ Ponmonpodutiv, ABB
```


## Confirm license activation

```
Upload activation request
```

```
Activating your licenses offline - Third step "Upload activation receipt":
1. Confirming the activation is only supported for Automation Builder 1.1.1 and later. Please ignore this step if you are using Automation Builder
    1.1.0
2. Pick the created activation receipt file.
3. Click "Upload activation receipt".
```

```
Pick activation receipt file (* WibuCmRaC)
    Search.
Upload activation receipt Back
```


## Offline activation

If the runtime licensing process was closed between saving the license activation request file and obtaining the license activation file from the ABB license server, perform an offline activation:

1. Right-click on the PLC node and select "PLC runtime licensing" from the "Runtime Licensing" menu.
$\Rightarrow$ A wizard starts. Follow the instructions.
2. Select the option "Complete offline licensing process".
3. Select [Browse] and select the license activation file (*.WibuCmRaU) from the defined storage location.
$\Rightarrow$ The license is activated on the PLC device.
If the license shall be used on another PLC device, the installed license can be returned ${ }^{\leftrightarrows}$ Chapter 6.3.2.2.2.5 "Returning a license" on page 1452.
4. To complete the licensing process, a license receipt file must be uploaded to the $A B B$ license server.
Save the license receipt file and upload it manually from a PC with internet connection to http:///c.codemeter.com/32838/depot/index.php.
$\Rightarrow$ A license confirmation is returned.

## Activating a demo license

It is possible to try out device features or library features by using a demo license on the PLC. With this, you can use the features for a limited time period.
$\checkmark \quad$ PC and PLC device are connected. In case of no connection, perform the activation via memory card $\Longleftrightarrow$ Chapter 6.3.2.2.2.4 "Licensing via memory card" on page 1450.
$\checkmark$ There is a connection to the Internet. In case of no connection, perform the activation on another PC with Internet connection ${ }^{4} \Rightarrow$ Chapter 6.3.2.2.2.2.2 "Activation without internet connection" on page 1447.

1. Right-click on the PLC node and select "PLC runtime licensing" from the "Runtime Licensing" menu.
$\Rightarrow$ A wizard is started. Follow the instructions.
2. Select the option "Create a demo license" and click [Next] to finish the licensing procedure.
$\Rightarrow$ The demo license is validated by the ABB license server and afterwards activated on the PLC device.

The demo license is valid for all licensed features and the duration of the demo license is 10 days PLC in run.
As it is a demo license, there is no grace period after expiration (product licenses do not have an expiration date).
The demo license can be extended any numbers of times. After expiration, the PLC will not switch back to run mode until a new license is activated or the features requiring a runtime license are removed from the PLC application.
For AC500 V3 it is visible on the display, if the PLC is running on demo license.

## Licensing via memory card

Wibu memory card

## NOTICE!

After removing a Wibu memory card (which holds the AC500 runtime license), the PLC system moves into 'Stop' mode after 24 h .
Ensure to insert the Wibu memory card at the time.

The Wibu memory card is different from the "normal" memory card that is required in the following section. The Wibu memory card is an external product that allows you to use the runtime license on different PLCs if you have several PLCs. Please note the notice at the beginning of the chapter.

When you have no connection between your PC and the PLC device the licensing procedure can be done via a memory card.

On the PC:
Create a license request

There is a connection to the internet.The memory card can be used with AC500 V3 products.

## NOTICE!

If a SDCard.ini file is stored on the memory card, the file will be overwritten.

1. Place the memory card in the PC.
2. Right-click on the PLC node and select "Prepare PLC license SD memory card" from the "Runtime Licensing" menu.
3. From the file system select the root folder of the memory card.
$\Rightarrow$ A success message is displayed when the creation of the memory card files is completed.
The license request files are stored to the selected folder.

## On the PLC: <br> Transfer the <br> license data

1. Insert the memory card into the PLC device and reboot the PLC.
$\Rightarrow$ When the license request file is successfully created by the PLC, "done" is shown on the display of the PLC.
2. Remove the memory card from the PLC.

On the PC:
Enter the
license activation key

For this action, internet connection is required.

1. Place the memory card into the PC.
2. Open the PLC project in Automation Builder. Ensure the PLC is logged out.
3. Right-click on the PLC node and select "PLC runtime licensing" from the "Runtime Licensing" menu.
$\Rightarrow$ A wizard is started. Follow the instructions.
4. Enter the license activation key.
5. From the filesystem, select the root folder of the memory card.
$\Rightarrow$ The previously created license request files are sent to the ABB license server. $A$ license activation is created on the memory card.
6. Remove the memory card from the PC.

## On the PLC: <br> Complete

license activa-
tion for the PLC

1. Insert the memory card into the PLC device and reboot the PLC. $\Rightarrow$ "done" is displayed on the PLC if license activation was successful.
2. Remove the memory card from the PLC

On the PC:
Complete license activation on the license server

Trying to activate a runtime license, that has already been activated (e.g. via an online conncetion), will result in following error in the sdcard.rdy file:
;Result of license file import ImportLicense=13;Internal Error (1)

For this action, internet connection is required.

To complete the licensing process, the license receipt file must be uploaded to the ABB license server.

1. Place the memory card into the PC.
2. Upload the license receipt file manually from a PC with internet connection to http://lc.codemeter.com/32838/depot/index.php.


The license receipt on the memory card is located in the subfolder license
$\Rightarrow$ A license confirmation is returned.

## Returning a license

Returning a
license without memory card

A license which has been installed on a PLC device can be returned and installed on another PLC device.
$\checkmark \quad$ PC and PLC device are connected. In case of no connection, perform the activation via memory card \& "Returning a license via memory card" on page 1452.

1. Right-click on the PLC node and select "Return active license" from the "Runtime Licensing" menu.
$\Rightarrow$ A wizard is started. Follow the instructions.
2. Enter the license activation key and click "Return license".
$\Rightarrow$ The results of the return process will be displayed in the dialog.
The license from the PLC device is removed and can be used now for another PLC device.

## Returning a license via memory card

## NOTICE!

After returning a AC500 runtime license, the PLC system moves into "Stop" mode after 24 h .

1. Insert the memory card in the PC and execute "Runtime Licensing $\rightarrow$ Prepare PLC license SD card" on the PLC node.
2. Place the memory card into the PLC.
3. Perform "power cycle" after a successful update reboot the PLC and connect to the PLC. $\Rightarrow$ The License is removed from the PLC.
4. Place the memory card into the PC.
5. Right-click on the PLC node and select "Return active license" from the "Runtime Licensing" menu.
$\Rightarrow$ A wizard is started. Follow the instructions.
6. Enter the license activation key and click "Return license".

7. Click [Browse] and select the root folder of the memory card.
$\Rightarrow$ Returning of the license is started.
8. Place the memory card in the PLC device and reboot the PLC.
$\Rightarrow$ The license from the PLC device is removed and can be used now for another PLC device.
9. To complete the licensing process, a license receipt file must be uploaded to the ABB license server.

Save the license receipt file and upload it manually from a PC with internet connection to http:///lc.codemeter.com/32838/depot/index.php.
$\Rightarrow$ A license confirmation is returned.

## View license information

To view the license information of AC500 V3 products:

1. In the Automation Builder device tree double-click on the PLC node. $\Rightarrow$ The PLC tab is opened.
2. In the PLC tab select "License Information".

| FPLC_AC500_V3 $\times$ |  |
| :---: | :---: |
| Communication Settings | Required licenses: |
| PLC Settings |  |
| Version information |  |
| Statistics |  |
| Files |  |
| Log | Available licenses: |
| PLC Shell | Login required |
| Users and Groups |  |
| Access Rights |  |
| Symbol Rights |  |
| PM5650-2ETH Hardware | Missinglicenses: |
| CPU-Parameters Parameters | Login required |
| IEC Objects |  |
| I/O mapping list |  |
| I/O-Bus I/O Mapping |  |
| Task Deployment |  |
| Applications |  |
| Backup and Restore |  |
| Status |  |
| Diagnosis |  |
| License Information |  |
| Information |  |

$\Rightarrow$ The project is scanned for required licenses.
If you are logged into a PLC, then the licenses available on the PLC are displayed. Missing required licenses are highlighted.

### 6.3.2.2.3 Connection of devices

## Configuring devices

All installed devices that are available in Automation Builder are listed in the device repository.
Chapter 6.3.2.1 "Device repository" on page 1443

Modify your Automation Builder project by adding device objects. Preset items can be replaced in the same way.

1. In the device tree, right-click an item node. Select [Add object].

2. Select the desired object and click [Add object].
3. Double-click the new object in the device tree to configure the device settings. Depending on the selected item different configuration tabs are available.

| A1523 Parameters | Parameter | Type | Value | Defaut Value | Unit | Description |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| A1523 yo Mapping | - Ifonce modile | Enumeration of 8 ¢TE | No | No |  | This parameter allows to set whether |
|  | - Chedk asply | Enumerstion of BYTE | On | On |  | Oreck supply |
| V/0 mapping list | - Inout 0, charrel conflgaraton | Enumerstion of BYTE | Not used | Not used |  | Anslog inget 0-Cenfiguration of ana |
|  | - Inout 0, cheod chamel | Enumeration of BYTE | Phewsib, Cut wire, Short direut | Pausib, Cut mire, Stort diruit |  | Asalog ingut 0-Check dannel |
| Information | - Inout 1, charnel configraton | Enumeration of SYTE | Not used | Not used |  | Asalog inget 1-Configuration of ana |
|  | - trout 1, ched chamel | Enumeration of BYTE | Phausib, Cut wire, Short circut | Plausib, Cut wire, Short dircuit |  | Analog ingut 1 - Check dannel |
|  | - Inout 2, cramel configraton | Enumeration of BYTE | Not used | Not used |  | Anslog ingut 2-Configuration of ana |
|  | - trout 2, creck cramel | Enumeration of BYTE | Plausib, Cut wire, Short drout | Plauslb, Cut wire, Short circuit |  | Anslog input 2 - Check channel |
|  | - Input 3, clamel conflguraton | Enumeration of BYTE | Not used | Not used |  | Anslog ingut 3-Configuration of ana |
|  | - Inout 3, chedk chame | Enumeration of grte | Plausib, Cut wire, Short circut | Plausib, Cut mire, Stort dircuit |  | Aaslog input 3-Check ciannel |
|  | - Input 4, charnel configuraton | Enumersion of BYTE | Not used | Not uses |  | Anslog ingut 4-Configuration of man |
|  | - thout 4, deed chasnel | Enumeration of BYTE | Phausb, Cut wire, Short dircut | Pausib, Cut wire, Short dirouit |  | Analog ingut 4-Check clannel |
|  | - Inout S, clarnel configraton | Enumeration of BYTE | Not used | Not used |  | Aasalog ingut 5 - Configuration of max |

## Update of AC500 devices

Perform a firmware update to update AC500 V3 devices. «y Chapter 6.3.1.4.3 "Installation and update of the AC500 V3 firmware" on page 1422

## Comparing objects

To compare similar objects within a project (such as the project configuration) select both objects. Right-click and select "Compare Objects" to see the differences.


### 6.3.2.2.4 IP settings

## Configuration of the IP address

The IP address for AC500 devices can be set or changed in Automation Builder using

- the IP configuration tool which is described in the following.
- the Communication Settings. ${ }^{\bullet}$ Chapter 6.3.2.2.4.4 "Configuration of communication via Ethernet (TCP/IP)" on page 1469

As an alternative the IP address can be changed at the hardware device itself. $\Leftrightarrow$ Chapter 6.8.2.5.5 "Description of the function keys" on page 4457

## NOTICE!

For any AC500-ETH device where the IP address can be set, the IP: 0.0.0.0 is invalid and should not be set.
This may cause damage to the PLC and/or the communication module.

## Configuration of the IP settings with the LED display

The IP settings for the PLC can be set directly on the processor module via keypad and LED display.
(3) Chapter 6.8.2.5.5.5 "CFG - configuration" on page 4462

## Configuration of the IP settings with the IP configuration tool

## Introduction

The IP configuration tool can be used

- to set or change the IP address of devices.
- to scan the network for available hardware devices.
- to update the firmware of devices.

This functionality is only supported if the IP configuration tool is used stand-alone.

- to activate certain functionality on hardware devices.

This feature is only available on AC500 V3 devices.
The IP configuration tool is part of Automation Builder and can be called via "Tools
$\rightarrow$ IP-Configuration".
Further the IP configuration tool can be used stand-alone without an Automation Builder application running. The stand-alone variant requires a separate installation via the Installation Manager $\Leftrightarrow$ Chapter 6.3.2.2.4.3.2 "Stand-alone installation" on page 1457.
After the installation, the IP configuration tool is started via .exe file / desktop icon.
Some functionality is only supported if the IP configuration tool is used standalone, e.g. for firmware updates for communication interface devices.

## Stand-alone installation

The IP configuration tool is part of Automation Builder and can be called via "Tools $\boldsymbol{\rightarrow}$ IP-Configuration". A separate installation is only required if the IP configuration tool shall be used stand-alone.

1. Open the Installation Manager in Automation Builder: "Tools $\rightarrow$ Installation Manager".
2. Close all other instances of Automation Builder as only one instance of the program can be executed at a time.

3. Click [Modify] and select the "IP Configuration Tool" from the structure tree.

| 能 ABB Automation Builder 2.4.0 Build 928 - Selection Page | - | $\square$ | $\times$ |
| :---: | :---: | :---: | :---: |
| Autometion Builder |  |  |  |

## Select type of installation:

Premium Edition
Select/Unselect to Install/Uninstall:

```
\(\square \vee\) ABB Automation Builder
    +--\square Professional Tool Chain
    \square-\vee\squarePLC
    #
    \pm}\downarrow\square\mathrm{ Drives
\square- - Tools
    \square-\squareAC500-Tools
    - - - CoDeSys OPC Server 3.5
        - \square CoDeSys OPC Server 3.5
        \checkmark IP Configuration Tool
            - - MultiOnlineChange Tool
    \square-
    --\square HMI
        - Panel Builder }600\mathrm{ Runtime for PC
        - Control Panel - CP600
        - Automation Builder Standalone Tools
        ABB License Manager
        - }\square\mathrm{ Safety PLC - Pluto
```

    V Install also previous product versions
    \(\square\) Automation Builder 2.3
    \(\square\) Automation Builder 2.2
    Automation Builder 2.1
    | Description | Release Notes | Packages |
| :--- | :--- | :--- |

IP Configuration Tool
The standalone tool for IP address configuration

This software package contains the following main components and features:

- ABB IP Configuration Tool
- Not Installed Up to date


4. Click [Continue] to start the installation.
$\Rightarrow$ After a successful installation the IP configuration tool is available as stand-alone tool (.exe).
$\Rightarrow$ To start the IP configuration tool, click the new created desktop icon.

## Network scan

With a network scan all devices that have been found in the network by the scan process are listed, i.e. ABB devices such as AC500 processor modules, AC500 communication interface modules or $A B B$ drives.

1. Start the IP configuration tool in Automation Builder ("Tools $\rightarrow$ IP-Configuration") or start it stand-alone (.exe).
2. The "IP-Configuration" dialog opens. Define the device type for the network scan by selecting the desired option under "Scan Protocol":

- "ABB Net config protocol":

Use this option for AC500 devices such as processor modules, CI5xx-Modbus devices or ABB drives. The device(s) to be scanned must be connected to the PC via a direct Ethernet connection.

- "Profinet Dynamic Configuration Protocol (DCP)":

Use this option for PROFINET communication interface modules. The device(s) to be scanned must be connected to the PC via a direct Ethernet connection (not via CM579).
For the scan, a NPcap driver needs to be installed separately.

- "EtherCAT":

Use this option for EtherCAT communication interface modules. The Ethernet cable must be connected directly to the first EtherCAT slave device of the EtherCAT fieldbus. Ensure that no EtherCAT master device is available on the bus when a scan is performed.
"Emergency" option: Enable this option to check on failures in the EtherCAT assembly during the scan process, i.e. a frame loss or interchanged ports. Errors are displayed. For the scan, a NPcap driver needs to be installed separately.
3. Click [Scan] to start the scan process.

4. All devices that have been found in the network are listed including hardware and connection details. The following details can be changed under "IP settings":
$\Rightarrow$ - "IP Address":
Current IP address of the device.

- "Conf. IP Address":

Configured IP address of the device. A changed IP address will update this column.

- "FW Version":

Current installed firmware version of the device. This field is visible not until a first network scan. If this field is still empty after a network scan, check on connection errors.

The IP address of some devices, e.g. EtherCAT devices cannot be changed.

## Changing the IP address

1. In order to change the IP address of devices perform a network scan.

② Chapter 6.3.2.2.4.3.3 "Network scan" on page 1459
2. Select a device from the list and select the appropriate protocol under "Scan protocol".
"DHCP" or "BOOTP" option: If required, DHCP or BOOTP can be used to receive the IP address for the device from the server.
"IP address", "subnet mask", "Std. gateway": Use these fields to change the IP address settings including the settings for the subnet mask and the standard gateway. Ensure that the combination of connection settings is correct.

## Note for CI52x-Modbus devices

Consider the behavior of CI52x-Modbus devices if the last number of the IP address is set to " 0 ".
(2) "Check last number of IP address" on page 1462
3. Change the settings for the IP configuration and click [Send settings] to transmit the data to the device.

$\Rightarrow$

## Note for PROFINET devices

The device name of PROFINET devices can be edited. If changing the name, ensure the following rules apply:

- Labels must be separated by "."
- Total length: 1 to 240
- Label length: 1 to 63
- Labels can consist of characters [a-z] and numbers [0-9]
- Labels are not allowed to start with "-"
- Labels are not allowed to end with "-"

4. In order to keep all IP changes after a power cycle, the settings can be stored permanently. Confirm the prompted message during the scan process.

Check last number of IP address

This hint is only valid for Cl 52 x -Modbus devices.
Check the last number of the IP address. If it is set to "0", the IP address setting for this last number will be used from the rotary switches on the hardware device.

## Example:



## Firmware update

The firmware of AC500 communication interface modules can be updated with the IP configuration tool.

For this, the IP configuration tool must be used as stand-alone variant.
(y) Chapter 6.3.2.2.4.3.2 "Stand-alone installation" on page 1457

It is not possible to perform a firmware update out of Automation Builder.

- For PROFINET communication interface modules a firmware update is only supported for devices with firmware version $\geq 3$ 3.3.3.
- For EtherCAT communication interface modules a firmware update is only supported for devices with firmware version $\geq 2.1$.4.
- For Modbus communication interface modules a firmware update is only supported for devices with firmware version $\geq 3.2 .13$.


## Requirements: Before the firmware update

- Ensure a fast and stable network connection
- Close all unused applications on the executing PC
- Stop the communication between AC500 PLC and the communication interface module that shall be updated


## During the firmware update

- Do not close the IP configuration tool
- Do not open Automation Builder software or any other application
- Do not switch-off the communication interface module that shall be updated
- Do not disconnect the Ethernet connection of a communication interface module or the executing PC

The firmware update will stop the operation of the affected device(s). Hence, the device(s) will become unresponsive for 1-2 minutes.

## Procedure:

1. Start the IP configuration tool stand-alone (.exe).
2. Perform a network scan.
«y Chapter 6.3.2.2.4.3.3 "Network scan" on page 1459
3. Select the devices that shall be updated from the list and click [Scan] to trigger the scan process.
A multiple selection of several devices is possible via control key, however, ensure to select only devices of the same protocol at a time. Otherwise the firmware update fails.
4. This step is only required for devices that require an installed NPcap driver. In this case an appropriate message including a download link is prompted in the IP-Configuration dialog:

$\Rightarrow$ Click on the displayed link https://nmap.org/download.htmI and download the latest version of the npcap-X.X.exe file.


For those who prefer the command-line zip files (Installation Instructions; Usage Instructions), they are still available. The Zenmap graphical interface is nor includec from a DOS /command window. Or you can download and install a superior command shell such as those included with the free Cygwin system. Also, you need to ru. 2013 Redistributable Package installers which are included in the zip file. The main advantage is that these zip files are a fraction of the size of the executable installe

Latest stable command-line zipfile: nmap-7.91-win32.zip

## Linux RPM Source and Binaries

Many popular Linux distributions (Redhat, Mandrake, Suse, etc) use the RPM package management system for quick and easy binary package installation. We have * RPM packages, though these simple commands usually do the trick:
rne -vhll httns ///nean.arg/dist/nean-7.91-1.xhh fa. mne
$\Rightarrow$ After the download, execute the file as administrator and restart the scan process.
$\Rightarrow$ The devices that have been scanned are listed.
5. Click [Update Firmware] to start the firmware update for the selected devices.

6. For $\mathrm{CI} 50 \mathrm{x}, \mathrm{CI} 51 \mathrm{x}$ and CI 52 x devices a signature check is started. Select the appropriate firmware update file (*.bin) for the device(s). Example: C:\AC500\AC500_CI52x_Firmware_V3.2.8.bin.

After a successful signature check the firmware update file (*.bin) and the respective signature file (*.bin.sig) are transferred to the device. This can last up to 3 minutes.

If the signature check fails, check the availability of the *.bin file and the *bin.sig file.
7. A status check followed by a device reboot followed by a second status check is performed automatically.

After the firmware update all outputs of the updated devices are set to ' 0 '.
8. After a successful firmware update the update status or the new firmware version is displayed in the "FW Version" field.
If this field is empty, there possibly is a connection error between the device and the executing PC.

Exception: For EtherCAT devices an empty "FW Version" field does not indicate a connection error.

$\Rightarrow$ If the firmware update fails

- check the requirements for the update procedure.
» "Requirements:" on page 1463
- check the hints for trouble-shooting.
- perform a network scan and repeat the update. If the error still persists power cycle the device and try the update again.


## Blink functionality

This function activates flashing of the backlight of an AC500 LED display.

1. From the menu, select "Tools $\rightarrow$ IP-Configuration".
2. Click [Scan] to trigger the scan process for devices in the network.
$\Rightarrow$ A progress bar shows the progress. The IP settings of a selected device is displayed below the list and can be edited.
3. Adjust your desired time and click [Blink] to activate flashing.


## Troubleshooting for IP configuration tool

Firewall excep- On a standard Windows 7 installation without third party firewall or security tools installed the IP tions: configuration tool should work properly.

The Automation Builder setup installs rules or exceptions for the built-in Windows firewall to allow IPConfig to receive the responses for the IPConfig scan.
To check the Windows firewall is set correctly check the firewall settings.

Windows 7I On the network that is used for communication with the PLC, set "Incoming connections" to Windows 10:

Networks at home or work where you know and trust the people and devices on the network

| Windows Firewall state: | On |
| :--- | :--- |
| Incoming connections: | Block all connections to programs that are not on the <br> list of allowed programs |
| Active home or work (private) networks: | Network |
| Notification state: | Notify me when Windows Firewall blocks a new <br> program |

Networks in public places such as airports or coffee shops

Windows Firewall state:
Incoming connections:

Active public networks:
Notification state:

On
Block all connections to programs that are not on the list of allowed programs


Unidentified network
Notify me when Windows Firewall blocks a new program

If a third party firewall is used these exceptions must be configured manually.

$$
\begin{aligned}
& \text { Either exceptions for applications can be entered: Automation Builder and IP } \\
& \text { configuration tool must be added as application. } \\
& \text { Or the protocol and the port number must be given (for IPConfig: UDP protocol } \\
& \text { and port number } 24576 \text { ). }
\end{aligned}
$$

| Fault description | Cause | Remedy |
| :--- | :--- | :--- |
| Error - general |  | Ensure that all requirements have been con- <br> sidered before and during the update proce- <br> dure. |
| The "FW Version" field <br> is empty after the net- <br> work scan or the firm- <br> ware version has not <br> been updated after the <br> update procedure (only <br> for Modbus devices and <br> PROFINET devices) | Connection error between the device <br> and the executing PC. |  |


| Fault description | Cause | Remedy |
| :--- | :--- | :--- |
| Signature check failed | During the firmware update of CI50x, <br> Cl51x and CI52x devices a signature <br> check is started. The update proce- <br> dure expects a firmware update file <br> (*.bin) and a signature file (*.bin.sig) <br> in the same directory. Without a signa- <br> ture file the signature check will fail. | Example: <br> Firmware update file: <br> C:\AC500\AC500_CI52x_Firmware_V3.2.8.bin <br> Signature file:C:\AC500\AC500_CI52x_Firm- <br> ware_V3.2.8.bin.sig |
| Error: Package timeout | A timeout error may occure due to an <br> instable network. | Keep the executing PC as near as possible <br> to the devices that shall be updated. Avoid net- <br> work switches. |
| Error: Unable to read <br> device status | A read error may occure due to errors <br> in the firmware update protocol. After <br> the firmware update the IP configura- <br> tion tool reads out the status of the <br> updated device in order to check if the <br> update was successful. | Perform the update again. |
| Error: IP is not unique | If an IP address is obtained by more <br> than one device an error occures. A <br> firmware update is not possible. | Change the IP addresses. |
| Error: Error State | Internal device error during the firm- <br> ware update. | Che TCP communication is not suffi- |
| Error: Can't connect to <br> device | Increase the connection quality. Keep the exe- <br> cuting PC as near as possible to the devices <br> cient. |  |

## Configuration of communication via Ethernet (TCP/IP)

## General

Programming via Ethernet is only possible on a PC with Ethernet board and installed network. Programming can be done via the internal (onboard) Ethernet communication module.

An application note describes the configuration of an AC500 V3 PLC for EtherNet/IP communication.


## Enter a known PLC IP address

1. Right-click the top node "PLC_AC500 <...>" and select "Communication Settings" from the context menu.
$\Rightarrow$ Dialog box "Communication Settings <...>" appears.

2. Enter your PLC IP Address and click [OK].

## Enter PLC IP address by scanning devices

1. Right-click the top node "PLC_AC500 <...>" and select "Communication Settings" from the context menu.
$\Rightarrow$ Dialog box "Communication Settings <...>" appears.

2. Click [...].
$\Rightarrow$ Dialog box "Communication Settings <...>" appears.
3. Click [Scan], select your desired PLC and click [OK].
$\Rightarrow$ Entry is transferred to the dialog box "Communication Settings <...>".
Click [OK].

4. Click to log in the "PLC_AC500_V3" project.

## Enter PLC IP address by [Advanced Settings...]

If a remote gateway instead of a local one has to be used it can be configured in the [Advanced Settings...].

1. Right-click the top node "PLC_AC500 <...>" and select "Communication Settings" from the context menu.
$\Rightarrow$ Dialog box "Communication Settings <...>" appears.

2. Enable checkbox "Use advanced settings" and click [Advanced Settings...].
$\Rightarrow$ Tab "Communication Settings" opens.
3. Check gateway or change if required.
$\Rightarrow$ Successful connection is indicated by green dot on the gateway icon.
4. 



Check IP address or change if required.
5. Press ENTER to confirm changed IP address.
$\Rightarrow$ Successful communication is indicated by green dot on the PLC icon.
6. Or instead of the last two steps:

## Set the IP address via a scan.

Click [Scan Network], select your desired PLC and click [OK].

$\Rightarrow$ Successful connection is indicated by green dot on the gateway icon.
7. Click to log in the "PLC_AC500_V3" project.

## Connect to PLC via hostname

Automation Builder supports the connection to PLCs via hostname.

To access a V3 PLC via hostname, the user should add the prefix "dns:" in the editor view "Communication Settings" of that PLC.


This enables a login to a V3 PLC via hostname.

Firmware update of a V3 PLC via Automation Builder requires the configuration of the IP－Address in bytes and will not work with any hostname．

## 6．3．2．2．5 Online program modification

A fast online program modification of the user program is possible without interrupting the running operation．If data areas should be saved during power OFF／ON，they can be stored in the flash EPROM．An optional battery saves data in the RAM．

## 6．3．2．3 Processor modules

## 6．3．2．3．1 Configure a processor module in the device tree

1．Add a processor module to your project．$\Rightarrow$ Chapter 6．3．1．1．2＂Creating a new project＂ on page 1398
2．Double－click the PLC node in the device tree．
$\Rightarrow$ This will open a new window with tabs for the device configuration：
－＂Communication Settings＂＂Chapter 6．4．1．21．2．8．3＂Tab＇Communication Set－ tings＇＂on page 2427
－＂PLC Settings＂${ }^{\wedge}>$ Chapter 6．4．1．21．2．8．10＂Tab＇PLC Settings＇＂on page 2439
－＂Version information＂$乡>$ Chapter 6．3．1．4．2．2．1．1＂Version information＂ on page 1419
－＂Statistics＂${ }^{\text {² }}$ Chapter 6．9．2．4．2＂Statistics＂on page 4646
－＂Files＂${ }^{\Perp}$ Chapter 6．4．1．21．2．8．8＂Tab＇Files＂＇on page 2437
－＂Log＂＂Chapter 6．9．2．4．3＂Log＂on page 4647
－＂PLC Shell＂$\Longleftrightarrow$ Chapter 6．3．5．6＂PLC shell commands＂on page 1756
－＂Users and Groups＂${ }^{\text {• }}$ Chapter 6．4．1．21．2．8．14＂Tab＇Users and Groups＂＂ on page 2450
－＂Access Rights＂＂a Chapter 6．4．1．21．2．8．15＂Tab＇Access Rights＂＂on page 2453
－＂Symbol Rights＂${ }^{\circ}$ Chapter 6．4．1．21．2．8．16＂Tab＇Symbol Rights＂＇on page 2458
－＂PM5xxx Hardware＂${ }^{\mu}$ Chapter 6．3．2．3．2＂Changing the processor module type＂ on page 1475
－＂CPU Parameters＂$\left.{ }^{\star}\right\rangle$ Chapter 6．4．1．21．2．8．4＂Tab＇Parameters＂＂on page 2433
－＂IEC Objects＂边 Chapter 6．4．1．21．2．8．13＂Tab＇＜device name＞IEC Objects＇＂ on page 2449
－＂I／O mapping list＂出 Chapter 6．3．2．13．8＂I／O mapping list＂on page 1572
－＂I／O－Bus I／O Mapping＂${ }^{4}$ Chapter 6．9．2．5＂Live values in views with I／O compo－ nents＂on page 4649
－＂Task Deployment＂${ }^{\Downarrow}$ Chapter 6．4．1．21．2．8．18＂Tab＇Task deployment＂＂ on page 2459
－＂Applications＂${ }^{\Downarrow}$ Chapter 6．4．1．21．2．8．5＂Tab＇Applications＇＂on page 2434
－＂Backup and Restore＂乡 Chapter 6．4．1．21．2．8．6＂Tab＇Backup and Restore＂＂ on page 2435
－＂Status＂乐 Chapter 6．9．2．4．5＂Status＂on page 4648
－＂Diagnosis＂${ }^{4}$ Chapter 6．9．1．4．4＂Device diagnosis＂on page 4613
－＂Diagnosis History＂${ }^{4}$ Chapter 6．9．1．4．5＂Diagnosis history＂on page 4614
－＂License Information＂« Chapter 6．3．2．2．2．6＂View license information＂ on page 1453
－＂Information＂General information about the device（name，vendor，version etc．）
3．Select the＂CPU Parameters＂tab to configure the parameters for the processor module． を Table on page 1555
4. Use the "PM5xxx Hardware" tab for later on changes of "Terminal Base Type" or "Processor Module Type" $\leadsto$ Chapter 6.3.2.3.2 "Changing the processor module type" on page 1475.
5. Select the "//O mapping list" tab to create mapping variables with better usability support compared to the tree structured view. $\Rightarrow$ Chapter 6.3.2.13.8 "//O mapping list" on page 1572
6. Select the "Backup and Restore" tab to create a backup or restore the project. $\Rightarrow$ Chapter 6.4.1.21.2.8.6 "Tab 'Backup and Restore'" on page 2435
7. Select the "Diagnosis" tab to know what errors have occurred in the project. 出 Chapter 6.9.1.4.4 "Device diagnosis" on page 4613

| Parameter | Default | Value | Description |
| :---: | :---: | :---: | :---: |
| Error LED | On | On | The error LED lights up for errors of all classes, no fail-safe function activated. |
|  |  | Off by E4 | Warnings (E4) are not indicated by the error LED, no fail-safe function activated. |
|  |  | Off by E3 | Warnings (E4) and minor errors (E3) are not indicated by the error LED, no fail-safe function activated. |
|  |  | POU control | Control LED ERR with POU PmErrLedSet |
| Check battery | On | On | The presence of the battery and the battery status are checked. If no battery is available or the battery is empty, a warning (E4) is generated and the ERR LED lights up. |
|  |  | Off | The presence of the battery is not checked. No warning (E4) is generated. The LCD display "Batt" (triangle) can not be acknowledged! This also applies if a battery is installed but empty. |
| Stop on error class | Diagnosis of at least error class 2 | Diagnosis of at least error class 2 | In case of a fatal or severe error (E1-E2), the user program is stopped. |
|  |  | Diagnosis of at least error class 3 | In case of a fatal, severe or minor error (E1-E3), the user program is stopped. |
|  |  | Diagnosis of at least error class 4 | In case of a fatal, severe or minor error (E1-E3) or a warning (E4) the user program is stopped. |
| Diagnosis - Add PLC name to node name | Off | Off | Diagnosis - Add PLC name to node name. |
|  |  | On |  |
| PLC behavior after voltage dip | Halt | Halt | Behavior of the PLC after short voltage dip: reboot or halt. |
|  |  | Reboot |  |
| Diagnosis history | On | On | Enable the diagnosis history. |
|  |  | Off | Disable the diagnosis history. |
| Max. Diagnosis history entries | 1000 | 1000 | Max. number of entries kept by the diagnosis history. |
| Missed cycle behavior | Next | Next | Skip the current cycle and start task in time on next cycle. |
|  |  | ASAP | Start the task immediately. |


| Parameter | Default | Value | Description |
| :--- | :--- | :--- | :--- |
| Communication <br> Schema | Default | Default | Balanced priority for communication <br> via communication modules (CMSs) and <br> onboard Ethernet communication. |
|  |  | Communica- <br> tion modules | Priority and high performance for commu- <br> nication module (CM) based communica- <br> tion via sync tasks. Lower priority for <br> onboard Ethernet and I/O bus. |
|  | Onboard <br> Ethernet | Priority for onboard Ethernet communica- <br> tion (e.g. via Modbus TCP). Lower pri- <br> ority for communication via communication <br> modules (CMs). |  |
|  | Real time <br> onboard <br> Ethernet | Very high priority for onboard Ethernet <br> communication (e.g. EtherCAT PROFINET <br> Ethernet). Low priority for communication <br> via communication modules (CMs). |  |
|  | Off | Off | Not automated reboot after E2 error. |
|  | On | Automated reboot after E2 error. |  |

### 6.3.2.3.2 Changing the processor module type

## General

In a project, you can change the target system by changing the type of processor module or terminal base type. If possible, the device configuration of fieldbuses and interfaces is kept and switched over to the device configuration of the new module.
Target change options:

- between platforms: from V2 platform to V3 platform (and vice versa)
- between module types: from AC500 (standard) to AC500-eCo (and vice versa)
- a combination of changed platform and changed module type


## Target change from a V2 processor module to a V3 processor module

Target change options:

- AC500 V2 processor module to AC 500 V 3 processor module
- AC500 V2 processor module to AC500-eCo V3 processor module
- AC500-eCo V2 processor module to AC500-eCo V3 processor module
- AC500-eCo V2 processor module to AC500 V3 processor module

Procedure: 1. Close CODESYS.
2. Double-click the "PLC_AC500_V2 <...>" node and open the "PM5<...> Hardware" tab.
3. Enable "Change to AC500 V3 PLC" and select the desired V3 processor module from the "PM5xx Type" drop-down list.

4. Click [Create V3 PLC].
$\Rightarrow$ The new V3 processor module is displayed in the navigation tree.
$\Rightarrow$ Change the node name of the processor module, if desired.

In case of a target change from AC500-eCo V2 to AC500-eCo V3, the I/O bus and Ethernet configuration is kept.

## Target change from a V3 processor module to another V3 processor module

Target change options:

- AC500 V3 processor module to AC500 V3 processor module
- AC500 V3 processor module to AC500-eCo V3 processor module
- AC500-eCo V3 processor module to AC500 V3 processor module
- AC500-eCo V3 processor module to AC500-eCo V3 processor module

Procedure: 1. Close CODESYS.
2. Double-click the "PLC_AC500_V3 <...>" node and open the "PM5<...> Hardware" tab.
3. Select the desired V3 processor module from the "PM5xx Type" drop-down list.


Fig. 271: Change_Hardware_V3

4．Ensure the correct＂Terminal Base Type＂is selected and click［Change PM／TB type］．
$\Rightarrow$ If possible，the device configurations from the previous processor module will be kept and switched over to the new processor module．
The device configurations that cannot be kept are listed in a prompted information dialog．


By default，all device configurations which cannot be switched over will be copied to a＂device pool＂section in the navigation tree（option＂Copy all objects that cannot be added to the new PLC into a device pool for further access＂）．If required，this backuped configuration can be used in another project or in another processor module configuration．
If the checkbox is deactivated all device configurations that cannot be switched will be lost after the execution of the target change．

$\int$ PLC＿AC500＿V3＿eco＿unplaced＿devices（Device Pool）
Fis OnBoard＿IO（12DI／6DOR／2DC）
－- Interfaces
且追 TA5120＿2AI＿UI（TA5120－2AI－UI）
蚟迫 TA5126＿2AO＿UI（TA5126－2AO－UI）
－ 10 OnBoard＿RTC

Target change from AC500－eCo V3 to AC500 V3

The configuration of the onboard I／Os，the option board slots and the onboard RTC cannot be changed－over to the new module．

Target change from AC500 V3 to AC500－eCo V3


ETH1 configuration is kept even if the configured protocols are not allowed for the selected AC500－eCo V3 PLC．In this case error messages are displayed in the messages window．


#### Abstract

Libraries which are not used anymore are not deleted with the target change. Libraries of option boards are kept in the Library Manager even if no longer available at the target module.


6.3.2.3.3 Changing the processor module type for AC500-eCo V3 CPU

It is not possible to change from an AC500 V3 processor module to an AC500eCo V3 processor module!

Changing an AC500-eCo V3 processor module to another AC500-eCo V3 processor module is possible and the same limitation as listed before are applying, only the available or possible feature from the new processor module will be kept from the old processor module.

## Close CODESYS.

1. Double-click the "PLC_AC500_V3 <...>" node.

2. Open the "PM50xx Hardware" tab and select the new "PM50xx Type" from the drop-down list.

### 6.3.2.3.4 Parameters of the processor module

## Automated reboot after E2 error

The parameter "Automated reboot after E2 error" allows to set the behavior of the CPU in case of severe errors (class E2).

| (1) PLC_AC500_V3 $\times$ |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Communication Settings | Parameter | Type | Value | Default Value | Unit | Description |
| PLC Settings | - Error Led | Enumeration of BYTE | on | on |  | Error LED: On: Auto control of LED ERR Off by E4: Auto control, c |
|  | - Check battery | Enumeration of BYTE | on | On |  | Check battery state (ff off and no battery then no error messsge |
| Version information | - Stop on error dass | Enumeration of BYTE | Diagnosis of at least error dass 2 | Diagnosis of atleasterror dass 2 |  | Prevents application from running in case there is at least one activ |
|  | - Diagnosis - Add PLC name to node name | Enumeration of BYTE | off | off |  | Diagnosis - Add PLC name to node name |
| Statistic | - PLC behaviour after voltage dip | Enumeration of BTIE | Halt | Halt |  | Behaviour of the PLC after short voltage dip: reboot or halt |
|  | - Diagnosis history | Enumeration of BYTE | On | On |  | Enables /disables the diagnosis history |
| Files | - Max. diagnosis history entries | WORD(0. 500000 | 1000 | 1000 |  | Max. number of entries kept by the diagnosis history |
|  | - Missed cyde behavior | Enumeration of BYTE | Next | Next |  | Behavior when a task is not able to keep its schedule: Exeate ASA |
| Log | - Communicaton Schema | Enumeration of BYTE | Default | Deffult |  | Communicaton Schema: - Default: Balanced prioity for communica |
|  | Automated reboot after E2error | Enumeration of BYTE | off | off |  | Automated reboot after E2 error |
| PLC Shell |  |  |  |  |  |  |
| Users and Groups |  |  |  |  |  |  |
| Access Rights |  |  |  |  |  |  |
| Symbol Rights |  |  |  |  |  |  |
| PM5675-2ETH Hardware |  |  |  |  |  |  |
| CPU-Parameters Parameters |  |  |  |  |  |  |

If the default setting "Off" is used, no automated reebot after an E2 error is performed. If the setting "On" is used, an automated reebot after an E2 error is performed.

## PLC behaviour after voltage dip

The parameter "PLC behaviour after voltage dip" allows to set the behavior of the CPU in case of short voltage dips.


If the default setting "Halt" is used, the CPU is changed to STOP mode if a short voltage dip >10 ms occurs. A new powercycle is required.
If the setting "Reboot" is used, CPU will reboot after power supply has recovered to nominal value.

Floating point values
A calculation with floating points can lead to the following values:

0 (zero) If a calculation results in an underrun, the value is set to 0 (result near 0 , but not presentable). Depending on the sign bit, it can be a positive zero or a negative zero. The operator "=" of -0 and 0 returns TRUE.

Infinity If a calculation results in an overrun, the value is set to Infinity (the result is not presentable). Depending on the sign bit, it can be a positive infinity (Infinity) or negative infinity (-Infinity).

If Infinity is converted into another data type it results in the maximum value of the other data type (e.g.. conversion into DWORD with REAL_TO_DWORD: 16\#FFFFFFFF, into DINT with REAL_TO_DINT: 16\#7FFFFFFF).
If -Infinity is converted into another data type it results in the maximum value of the other data type (e.g.. conversion into DWORD with REAL_TO_DWORD: 16\#00000000, into DINT with REAL_TO_DINT: 16\#80000000).

Except for:
TRUE := REAL_TO_BOOL(Infinity);
'\#Inf' := REAL_TO_STRING(Infinity);
'-\#Inf' := REAL_TO_STRING(-Infinity);

| Examples: | Infinity | -Infinity |
| :---: | :---: | :---: |
|  | Infinity := $1.0 / 0.0$ | -Infinity : $=-1.0$ / 0.0 |
|  | Infinity : = Infinity + Infinity | -Infinity := -lnfinity -Infinity |
|  | Infinity := Infinity + 1.0 | -Infinity := -Infinity + 1.0 |
|  | Infinity := LREAL_TO_REAL(Infinity) | -Infinity := LREAL_TO_REAL(-Infinity) |

NaN If a calculation results in an undefined value the result is set to NaN (Not a Number). The result of each calculation with NaN is NaN . The operators "<", "<=", ">" and ">=" return FALSE if either or both operands are NaN .
Operator "=" returns FALSE if one operand is NaN .
Operator "<>" returns TRUE if one operand is NaN .
If NaN is converted into another data type the result is 0 .
Except for:
TRUE := REAL_TO_BOOL (NaN) ;
'\#NaN' : = REAL_TO_STRING (NaN) ;
Examples: $\quad \mathrm{NaN}:=\operatorname{SQRT}(-2.0)$
$\mathrm{NaN}:=0.0 / 0.0$
NaN := Infinity -Infinity
$\mathrm{NaN}:=0.0$ * Infinity
NaN := Inifnity / Infinity

The result of an operation can be checked with the following program parts:

| Check for NaN (REAL): | Check for NaN (LREAL): |
| :---: | :---: |
| ```rX: REAL; IF (rX <> rX) THEN (* rX is a NaN *) ...; END_IF;``` | ```lrX: LREAL; IF (lrX <> lrX) THEN (* IrX is a NaN *) ...; END_IF;``` |


| Check for Infinity (REAL): | Check for Infinity (LREAL): |
| :---: | :---: |
| Infinity is represented with sign bit 0 , exponent of all 1 s and a fraction of all 0 s. -Infinity is represented with sign bit 1 , exponent of all 1 s and a fraction of all 0 s . |  |
| ```rX: REAL; prX: POINTER TO REAL; pdwX: POINTER TO DWORD; prX := ADR(rX); pdwX := prX; IF (pdwX^ = 16#7F800000) THEN (* rX is Infinity *) ...; END_IF; IF (pdwX^ = 16#FF800000) THEN (* rX is -Infinity *) ...; END_IF;``` | ```lrX: LREAL; plrX: POINTER TO LREAL; plwX: POINTER TO LWORD; plrX := ADR(lrX); plwX := plrX; IF (plwX^ = 16#7FF0000000000000) THEN (* IrX is Infinity *) ...; END_IF; IF (plwX^ = 16#FFF0000000000000) THEN (* IrX is -Infinity *) ...; END IF;``` |

### 6.3.2.4 AC500-eCo V3 onboard I/Os

According to the used AC500-eCo V3 processor module, the onboard I/Os are different and the functionality of the I/Os are adapted to the processor module type.

| Onboard I/O combination | $\begin{aligned} & \text { PM5012-x- } \\ & \text { ETH } \end{aligned}$ | $\begin{aligned} & \text { PM5032-x- } \\ & \text { ETH } \end{aligned}$ | $\begin{aligned} & \text { PM5052-x- } \\ & \text { ETH } \end{aligned}$ | $\begin{gathered} \text { PM5072- } \\ \text { T-2ETH(W) } \end{gathered}$ | $\begin{aligned} & \text { PM5082- } \\ & \text { T-2ETH } \end{aligned}$ |
| :---: | :---: | :---: | :---: | :---: | :---: |
| 6 DI, digital input 24 V DC / 4 DO, digital output transistor 24 V DC / 0.5 A | X |  |  |  |  |
| 6 DI, digital input 24 V DC / 4 DO, digital output relay $240 \mathrm{~V} \mathrm{AC} / 2 \mathrm{~A}$ | X |  |  |  |  |
| 12 DI , digital input 24 V DC / 8 DO, digital output transistor 24 V DC / 0.5 A / 2 DC, digital in/out configurable 24 V DC, 24 VDC / 0.5 A |  | X | X | X | X |
| 12 DI , digital input 24 V DC / 6 DO, digital output relay $240 \mathrm{~V} \mathrm{AC} / 2 \mathrm{~A} / 2$ DC, digital in/out configurable 24 V DC, 24 V DC / 0.5 A |  | X | X |  |  |

### 6.3.2.5 Configure the onboard I/O channel

The onboard I/Os support the following channels functions according to the processor module type:

| Onboard I/O <br> type | Channel func- <br> tion | PM5012-T-ETH | PM5012-R-ETH | Channel name <br> when available |
| :---: | :---: | :---: | :---: | :---: |
| Digital input <br> channel total <br> thereof as | Digital input | 6 | 6 | DIO ... DI5 |
| Fast input x, <br> max. 5 kHz | Digital input | 6 | 6 | DIO ... DI5 |
|  | Interrupt input | 4 | 4 | DIO ... DI3 |
|  | Fast counter | 2 | 2 | DI4 ... DI5 |
| Digital output <br> channel total <br> thereof as | Digital output | 4 | 4 | DOO ... DO3 |
|  | Digital output | 4 | 4 | DOO ... DO3 |
|  | Limit switch | 4 | - | DO0 ... DO3 |
|  | PWM output | 4 | - | DOO ... DO3 |


| Onboard I/O type | Channel function | PM5032-T-ETH <br> PM5052-T-ETH <br> PM5072- <br> T-2ETH(W) <br> PM5082-T-2ETH | PM5032-R-ETH <br> PM5052-R-ETH | Channel name when available |
| :---: | :---: | :---: | :---: | :---: |
| Digital input channel total thereof as | Digital input | 12 | 12 | DIO ... DI11 |
| Fast input x, max. 5 kHz | Digital input | 4 | 4 | DI0 ... DI3 |
|  | Interrupt input | 4 | 4 | DIO ... DI3 |
|  | Touch/Reset | 4, together with dedicated encoder | 4, together with dedicated encoder | DI0 ... DI3 |
| Fast input x, max. 100 kHz | Digital input | 4 | 4 | DI4 ... DI7 |
|  | Encoder input | 2, with A/B tracks | 2, with A/B tracks | DI4 ... DI7 |
|  | Fast counter | 4 | 4 | DI4 ... DI7 |
| Standard input | Digital input | 4 | 4 | DI8 ... DI11 |
| Digital output channel total thereof as | Digital output | 8 | 6 | $\begin{aligned} & \text { DOO ... DO7 } \\ & \text { DOO ... DO5 } \end{aligned}$ |
| Fast output x , max. 5 kHz | Digital output | 4 | - | DO0 ... DO3 |
|  | Limit switch | 4 | - | DO0 ... DO3 |
|  | PWM output | 4 | - | DO0 ... DO3 |
| Fast output x , max. $100 \ldots . .$. | Digital output | 4 | - | DO4 ... DO7 |
|  | Limit switch | 4 | - | DO4 ... DO7 |
|  | PWM output | 4 | - | DO4 ... DO7 |


| Onboard I/O <br> type | Channel func- <br> tion | PM5032-T-ETH <br> PM5052-T-ETH <br> PM5072- <br> T-2ETH(W) | PM5032-R-ETH <br> PM5052-R-ETH <br> PM5082-T-2ETH | Channel name <br> when available |
| :---: | :---: | :---: | :---: | :---: |
|  | PTO output | 2, pair of output, <br> in CW/CCW or <br> Pulse/Direction <br> mode | - | DO4 ... DO7 |
|  |  | or <br> or 4 as Pulse <br> with standard <br> outputs as Direc- <br> tion |  |  |
| Digital in/output <br> configurable <br> channel total <br> thereof as | Digital in/output | 2 | 2 | DC12 ... DC13 |
| Standard dig. <br> channel | Digital In/output | 2 | 2 | DC12 ... DC13 |
| Fast output, max. <br> 100 ... 200 kHz | Limit switch | - | 2 | DC12 ... DC13 |
|  | PWM output | - | 2 | DC12 ... DC13 |
|  | PTO output | - | 1, pair of output, <br> in CW/CCW or <br> Pulse/Direction <br> mode | DC12 ... DC13 |



Basic CPU:
PM5012-x-ETH

For all CPU versions the configuration of the input channels is the same. The configuration of the output channels is only available on CPU version with transistor output channels:


Version with relay outputs, same configuration for the input channels, no configuration for the output channels relay:

| 6DI/4DO-R Parameter | Parameter <br> - Run on config fault | Typ | Wert | Standardwert | Einheit | Beschreibung |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 6DI/4DO-R E/A-Abbild |  | Enumeration of BYTE | No | No |  | Start PLC program even on configuration fault |
|  | -- Digital inputs 24 VDC |  |  |  |  |  |
| 6DI/4DO-R IEC-Objekte | - Input 0, input delay | Enumeration of BYTE | 8 ms | 8 ms |  | Fast input 0 - Input delay |
|  | - Input 0, channel configuration | Enumeration of BYTE | Input/Interrupt | Input/Interrupt |  | Fast input 0 - Configuration (max. 5 kHz ) |
| IO Mapping List | - Input 1, input delay | Enumeration of BYTE | 8 ms | 8 ms |  | Fastinput 1-Input delay |
|  | - Input 1, channel configuration | Enumeration of BYTE | Input/Interrupt | Input/IIterrupt |  | Fast input 1-Configuration (max. 5 kHz ) |
|  | - Input 2, input delay | Enumeration of BYTE | 8 ms | 8 ms |  | Fastinput 2 - Input delay |
|  | - Input 2, channel configuration | Enumeration of BYTE | Input/Interrupt | Input/IIterrupt |  | Fast input 2 - Configuration (max. 5 kHz ) |
|  | - Input 3, input delay | Enumeration of BYTE | 8 ms | 8 ms |  | Fastinput 3 - Input delay |
|  | - Input 3, channel configuration | Enumeration of BYTE | Input/Interrupt | Input/Interrupt |  | Fast input 3 - Configuration (max. 5 kHz ) |
|  | - Input 4, input delay | Enumeration of BYTE | 8 ms | 8 ms |  | Fastinput 4-Input delay |
|  | - Input 4, channel configuration | Enumeration of BYTE | Input | Input |  | Fast input 4-Configuration (max. 5 kHz ) |
|  | - Input 5, input delay | Enumeration of BYTE | 8 ms | 8 ms |  | Fastinput 5 - Input delay |
|  | - Input 5, channel configuration | Enumeration of BYTE | Input | Input |  | Fast input 5-Configuration (max. 5 kHz ) |

The following parameter can be configured:

| Onboard I/O type | Parameter | Channel name | Default value | Value | Description |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Digital inputs | Input $X$, input delay | Channel 0..5 | 8 ms | No delay | Configures input with no delay |
|  |  |  |  | 1 ms | Configures 1 ms input delay |
|  |  |  |  | 8 ms | Configures 8 ms input delay |
|  |  |  |  | 32 ms | Configures 32 ms input delay |
|  | Input X, channel configuration | Channel 0.. 3 | Input/Interrupt | Input/Interrupt | Configures the channel as normal digital or interrupt input |
|  | The configuration /function of the | Channel $4 . .5$ | Input | Input | Configures the channel as normal digital input |
|  | following channels is realized using function blocks in the program |  |  | Encoder 0 track-A or B | Configures the pair of channels as encoder input track A or B. When that value is configured then both channels are reserved for that functionality |
|  |  |  |  | Forward counter | Configures the channel as forward counter |

The configuration of output channel is only available on the CPU with transistor outputs

| Digital out- <br> puts | Output X, <br> channel con- <br> figuration | Channel 0..3 | Output | Output | Configures the <br> channel as digital <br> output |
| :--- | :--- | :--- | :--- | :--- | :--- |

Standard CPU: PM5032-x-ETH, PM5052-x-ETH Pro CPU:
PM5072-T-2ETH, PM5082-T-2ETH

For all CPU versions the configuration of the input channels is the same. The configuration of the output channels is only available on CPU version with transistor output channels, the digital configurable In/Output channels are present on both version (transistor or relay output) but with different features configurable:


For the CPU with relay outputs, the digital configurable Input/Output channels have specific functionalities:

| Onboard I/O type | Parameter | Channel name | Default value | Value | Description |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Digital inputs | Input X, input delay | Channel$0 . .11$ | 8 ms | No delay | Configures input with no delay |
|  |  |  |  | 1 ms | Configures 1 ms input delay |
|  |  |  |  | 8 ms | Configures 8 ms input delay |
|  |  |  |  | 32 ms | Configures 32 ms input delay |
| Fast inputs max. 5 kHz | Input X, channel configuration | Channel 0.. 1 | Input/Interrupt | Input/Interrupt | Configures the channel as normal digital or interrupt input |
|  |  |  |  | Touch/Reset 0 | Configures the pair of adjacent channels as Touch/Reset inputs together with encoder 0 |
|  |  | Channel 2.. 3 | Input/Interrupt | Input/Interrupt | Configures the channel as normal digital or interrupt input |


| Onboard I/O type | Parameter | Channel name | Default value | Value | Description |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  | Touch/Reset 1 | Configures the pair of adjacent channels as Touch/Reset inputs together with encoder 1 |
| Fast inputs max. 100/200 kHz | The function of the following channels is realized using function blocks in the program | Channel $4 . .5$ | Input | Input | Configures the channel as normal digital input |
| Max. frequency 200 kHz | When that value is configured then both channels are reserved for that functionality |  |  | Encoder 0 track-A or B | Configures the pair of adjacent channels as encoder 0 input track A or B. |
| Max. frequency 100 kHz |  |  |  | Forward counter | Configures the channel as forward counter |
| Fast inputs max. $100 / 200 \mathrm{kHz}$ |  | Channel $6 . .7$ | Input | Input | Configures the channel as normal digital input |
| Max. frequency 200 kHz | When that value is configured then both channels are reserved for that functionality |  |  | Encoder 1 track-A or B | Configures the pair of adjacent channels as encoder 1 input track A or B . |
|  |  |  |  | Touch/Reset | Configures the pair of adjacent channels as Touch/Reset inputs together with encoder 0 |
| Max. frequency 100 kHz |  |  |  | Forward counter | Configures the channel as forward counter |
| The following configuration of output channel is only available on the CPU with transistor |  |  |  |  |  |
| Fast outputs, max. 5 kHz | Output X, channel configuration | Channel $0 . .3$ | Output | Output | Configures the channel as digital output |


$\left.$| Onboard I/O <br> type | Parameter | Channel <br> name | Default <br> value | Value |
| :--- | :--- | :--- | :--- | :--- |
|  | The configu- <br> ration / func- <br> tion of the <br> following <br> channels is <br> realized <br> using func- <br> tion blocks in <br> the program |  |  | Dimit switch | | Configures the |
| :--- |
| channel as limit |
| switch output | \right\rvert\, 

### 6.3.2.6 Mapping of the onboard I/O channels



1. Double-click "OnBoard_IO" in the device tree.
$\Rightarrow A$ tab opens in the editor view.

| (4) OnBoard_IO $\times$ |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| 12DI/8DO-T/2DC Parameters | Find |  | Filter Show all |  | - |
| 12DI/8D0-T/2DC I/0 Mapping | Variable | Mapping | Channel | Address | Type |
|  | - Digital inputs 24 VDC |  |  |  |  |
| 12DI/8D0-T/2DC IEC Objects | - |  | Slower inputs | \%IB0 | BYTE |
|  | 4 |  | Digital input DIO | \%IX0.0 | BOOL |
| I/O mapping list | * |  | Digital input DI1 | \%IX0.1 | BOOL |
|  | * |  | Digital input DI2 | \%IX0.2 | BOOL |
|  | * |  | Digital input DI3 | \%IX0. 3 | BOOL |
|  | - * |  | Fast inputs | \%IB1 | BYTE |
|  | * |  | Digital input DI4 | \%IX1.0 | BOOL |
|  | 4 |  | Digital input DI5 | \%IX1.1 | BOOL |
|  | ${ }^{40}$ |  | Digital input DI6 | \%IX1.2 | BOOL |
|  | 4 |  | Digital input DI7 | \%IX1.3 | BOOL |
|  | - ${ }^{4}$ |  | Standard inputs | \%IB2 | BYTE |
|  | * |  | Digital input DI8 | \%IX2.0 | BOOL |
|  | $\cdots$ |  | Digital input DI9 | \%IX2.1 | BOOL |
|  | * |  | Digital input DI 10 | \%IX2.2 | BOOL |
|  | * |  | Digital input DI11 | \%IX2.3 | BOOL |
|  | - Digital outputs $24 \mathrm{VDC} / 0,5 \mathrm{~A}$ transistor |  |  |  |  |
|  | -* |  | Slower outputs | \%QB0 | BYTE |
|  | ${ }^{0}$ |  | Digital output DOO | \%QX0.0 | BOOL |
|  | ${ }^{*}$ |  | Digital output DO1 | \%QX0.1 | BOOL |
|  | ${ }^{*}$ |  | Digital output DO2 | \%QX0.2 | BOOL |
|  | ${ }^{*}$ |  | Digital output DO3 | \%QX0.3 | BOOL |
|  | - ${ }^{\circ}$ |  | Fast outputs | \%QB1 | BYTE |
|  | ${ }^{*}$ |  | Digital output DO4 | \%QX1.0 | BOOL |
|  | ${ }^{*}$ |  | Digital output DO5 | \%QX1.1 | BOOL |
|  | K |  | Digital output DO6 | \%QX1.2 | BOOL |
|  | K |  | Digital output DO7 | \%QX1.3 | BOOL |
|  | - Digital configurable In/outputs $24 \mathrm{VDC} / 0,5 \mathrm{~A}$ transistor |  |  |  |  |
|  | - |  | Digital inputs - Transistor | \%IB3 | BYTE |
|  | - |  | Digital input DC12 | \%IX3.0 | BOOL |
|  | 4 |  | Digital input DC13 | \%IX3.1 | BOOL |
|  | $={ }^{*}$ |  | Digital outputs - Transistor | \%QB2 | BYTE |
|  |  |  | Digital output DC12 | \%QX2.0 | BOOL |
|  |  |  | Digital output DC13 | \%QX2.1 | BOOL |

2. Select "12DI/8DO-T/2DC I/O Mapping".
$\Rightarrow$ Here, you will map variable names (symbols) for the channels you will need in the program.
Digital input variables

| 12DI/8DO-T/2DC Parameters | Find Filter | Show all |  | - $\}$ Add FB for IO Channel |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| 12DI/8DO-T/2DC I/O Mapping | Variable | Mapping | Channel | Address | Type |
|  | -- Digital inputs 24 VDC |  |  |  |  |
| 12DI/8DO-T/2DC IEC Objects | - * |  | Slower inputs | \%IB0 | BYTE |
|  | *DI_00_OnBoard_IO_I0 | - | Digital input DIO | \%IX0.0 | BOOL |
| I/O mapping list | * |  | Digital input DI1 | \%IX0.1 | BOOL |
|  | * |  | Digital input DI2 | \%IX0.2 | BOOL |
|  | 4 |  | Digital input DI3 | \%IX0.3 | BOOL |

3. Open the list of the digital inputs.
4. Fill in the variable names:

| Channel | Type | Variable |
| :--- | :--- | :--- |
| Digital input DIO | BOOL | xDI_00_OnBoard_IO_10 |

Digital output variables

5. Open the list of the digital outputs.
6. Fill in the variable names:

| Channel | Type | Variable |
| :--- | :--- | :--- |
| Digital output DO0 | BOOL | xStartDrilling1 |

The suggested name convention is based on the "Hungarian notation". A name prefix is describing variable type: e.g., "x" = variable of type BOOL, "w" = WORD, "i" = INT (integer) etc. This increases the code readability and is helpful for program analysis.

### 6.3.2.7 Configuration of the onboard I/Os of AC500-eCo V3 PLC

### 6.3.2.7.1 Digital inputs from the onboard I/Os

Depending on the processor module used, several configurations are possible for the onboard I/Os mostly different per group of channels.

| Functionality to be realized | Processor module type | Digital input channels | Configuration of the channel to be selected in Automation Builder | Dedicated function block to be used in the user program | Comments |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Digital inputs | $\begin{aligned} & \text { PM5012-x- } \\ & \text { ETH } \end{aligned}$ | I0 ... 13 | Input/Interrupt | Not needed | Input delay can be sometimes configured according to channels type |
|  |  | $14 \ldots 15$ | Input |  |  |
|  | PM5032-x- <br> ETH, <br> PM5052-x- <br> ETH, | $10 \ldots 13$ | Input/Interrupt | Not needed | Input delay can be sometimes configured according to channels type, for the PLC with relay outputs, the digital |
|  |  | $14 \ldots 17$ | Input |  |  |
|  | $\begin{array}{\|l} \text { PM5072- } \\ \text { T-2ETH(W) } \end{array}$ |  |  |  |  |
|  | PM5082- <br> T-2ETH |  |  |  |  |


| Functionality <br> to be realized | Processor <br> module type | Digital input <br> channels | Configura- <br> tion of the <br> channel to be <br> selected in <br> Automation <br> Builder | Dedicated <br> function <br> block to be <br> used in the <br> user program | Comments |
| :--- | :--- | :--- | :--- | :--- | :--- |
|  |  | $18 \ldots$ I11 | Always inputs, <br> not configu- <br> rable |  | configurable <br> channels <br> have some |
| other configu- |  |  |  |  |  |
| rable features. |  |  |  |  |  |$|$


| (1) OnBoard_IO $\times$ |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 120I/8D0-T/2DC Parameters | Parameter | Type | Value | Default value | Unit | Description |
| 12DI/8DO-T/20C L/O Mapping <br> 12DI/8DO-T/2DC IEC Objects <br> I/o mapping list | - Run on config fault <br> - Digita inputs 24 VDC | Enumeration of Bite | No | No |  | Start PLC program even on configuration fault |
|  |  |  |  |  |  |  |
|  | - Input 0 , input delay | Enumeration of ByTE | 8 ms | 8 ms |  | Fast input 0-Input delay |
|  | - Input 0, chamel confguration | Enumeration of ByTE | Input/Interrupt | Input/Interrupt |  | Fast input 0-Confguration (max. 5 k KZ ) |
|  | - Input 1, input delay | Enumeration of ByTE | 8 ms | 8 ms |  | Fast input 1-Input delay |
|  | - Input 1, chamel configuration | Enumeration of BYTE | Input/Interrupt | Input/Interrupt |  | Fast input 1-Configuration (max. 5 k krz ) |
|  | - Input 2, input delay | Enumeration of ByTE | 8 ms | 8 ms |  | Fast input 2 - Input delay |
|  | - Input 2, chamel conffguration | Enumeration of BYTE | Input/Interrupt | Inout/interrupt |  | Fast input 2-Configuration (max, 5 k kz ) |
|  | - Input 3 , input delay | Enumeration of ByTE | 8 ms | 8 ms |  | Fast input 3-Input delay |
|  | - Input 3, chamel configuration | Enumeration of BYTE | Input/Interrupt | Input/Interrupt |  | Fast input 3-Configuration (max, 5 k kz ) |
|  | - Input 4, input delay | Enumeration of SyTE | 8 ms | 8 ms |  | Fast input 4-Input delay |
|  | - Input 4 , chamel confguration | Enumeration of ByTE | Input | Input |  | Fast input 4-Confguration: Forward Counter (max. 100 KHz ), Encoder (max. 200 KHz ) |
|  | - Input5, 5 inut delay | Enumeration of ByTE | 8 ms | 8 ms |  | Fast input 5-Input delay |
|  | - Input 5, channel confguration | Enumeration of ByTE | Input | Input |  | Fastinput 5 - Confguration: Forward Counter (max. 100 KHz 2 , Encoder (max. 200 KHz ) |
|  | - Input6, input delay | Enumeration of PYTE | 8 ms | 8 ms |  | Fast input 6 - Input delay |
|  | - Input 6 , chamel configuration | Enumeration of BrTE | Input | Input |  | Fastinput 6 - Configuration: Forward Counter (max. 100 kHz ), Encoder (max. 200 kHz ) |
|  | - Input 7 , input delay | Enumeration of SYTE | 8 ms | 8 ms |  | Fast input 7 - Input delay |
|  | - Input 7 , chamel configuration | Enumeration of BYTE | Input | Input |  | Fastinput 7 - Configuration: Forward Counter (max. 100 kHz ), Encoder (max. 200 kHz ) |
|  | - Digital outputs $24 \mathrm{VDC} / 0,5 \mathrm{~S}$ tansistor |  |  |  |  |  |
|  | - Output 0 , chamel configuration | Enumeration of BYTE | Outut | Outut |  | Fast output 0-Configuration (max. 5 kHz ) |
|  | - Output 1 , channel Configuration | Enumeration of ByTE | Output | Output |  | Fastoutput 1-Configuration (max. 5 KHz ) |
|  | - Output 2 , channel confguration | Enumeration of SYTE | Output | Outut |  | Fast output 2 - Confguration (max. 5 WHz ) |
|  | - Output 3 , channel confguration | Enumeration of Brit | Output | Ouput |  | Fastoutput 3-Confguration (max. 5 k Hz ) |
|  | - Output 4 , channel Configration | Enumeration of BYTE | Output | Output |  | Fast output 4-Configuration: PWM (max. 100 kHz ), PTO (max. 200 kHz ) |
|  | - Output 5 , channel configuration | Enumeration of BYTE | Output | Ouput |  | Fast output 5 - Configuration: PWM (max. 100 kHz ), PTO (max. 200 kHz ) |
|  | - Output 6 , channel confguration | Enumeration of BYTE | Output | Outut |  | Fast output 6 - Configuraton: PWM (max, 100 kHz ), PTO (max. 200 kHz ) |
|  | - Output 7 , channel configuration | Enumeration of BYTE | Output | Ouput |  | Fast output 7 - Confguration: PWM (max. 100 kHz ), PTO (max. 200 kHz ) |
|  | -- Digital configurable In/uututs $24 \mathrm{VDC} / 0,5 \mathrm{~A}$ transistor |  |  |  |  |  |
|  | - Input/ Output DC 12, channel conffguration | Enumeration of ESTE | Input/Output | Input/Output |  | Digital configurable channel 12 - Configuration |
|  | - Input/ Output DC13, channel configuration | Enumeration of byte | Input/Output | Input/Output |  | Digital configurable channel 13 - Confouration |

### 6.3.2.7.2 Fast counters in the onboard I/Os

(4) General details on fast counters.

③ Details on the configuration of the onboard I/Os channels.
Depending on the configuration for the input channels of the onboard I/O from the processor module different functionality are possible which must be used together with the dedicated function block of the user program.

| Functionality to be realized | Processor module type | Digital input channels | Configuration of the channel to be selected in Automation Builder | Dedicated function block to be used in the user program | Comments |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Forward counter | $\begin{aligned} & \text { PM5012-x- } \\ & \text { ETH } \end{aligned}$ | 10 ... 13 | Not relevant for the functionality | OBIOForwardCounter | Up to 2 forward counters with up to 5 kHz can be used, the other inputs can be used for other purpose |
|  |  | $14 . . .15$ | Forward counter |  |  |
|  | PM5032-x-ETH,PM5052-x-ETH,PM5072-T-2ETH(W)PM5082-T-2ETH | 10 ... 13 | Not relevant for the functionality | OBIOForwardCounter | Up to 4 forward counters with up to 100 kHz can be used, the other inputs can be used for other purpose |
|  |  | 14...17 | Forward counter |  |  |
|  |  | 18 ... 111 | Not relevant for the functionality |  |  |



### 6.3.2.7.3 A/B Encoder in the onboard I/Os

Depending on the configuration for the input channels of the onboard I/O from the processor module different functionality are possible which must be used together with the dedicated function block of the user program.
$\left.\begin{array}{|l|l|l|l|l|l|}\hline \begin{array}{l}\text { Functionality } \\ \text { to be realized }\end{array} & \begin{array}{l}\text { Processor } \\ \text { module type }\end{array} & \begin{array}{l}\text { Digital input } \\ \text { channels }\end{array} & \begin{array}{l}\text { Configura- } \\ \text { tion of the } \\ \text { channel to be } \\ \text { selected in } \\ \text { Automation } \\ \text { Builder }\end{array} & \begin{array}{l}\text { Dedicated } \\ \text { function } \\ \text { block to be } \\ \text { used in the } \\ \text { user program }\end{array} & \text { Comments } \\ \hline \begin{array}{lll}\text { A/B encoder 5 } \\ \text { kHz with } \\ \text { touch/reset } \\ \text { inputs }\end{array} & \begin{array}{l}\text { PM5012-x- } \\ \text { ETH }\end{array} & 10 & \text { Touch/Reset0 }\end{array} \begin{array}{l}\text { OBIOEnco- } \\ \text { derCounter }\end{array} \begin{array}{l}\text { The function- } \\ \text { ality uses the } \\ \text { 4digital }\end{array}\right\}$

| Functionality to be realized | Processor module type | Digital input channels | Configuration of the channel to be selected in Automation Builder | Dedicated function block to be used in the user program | Comments |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | 14 | Encoder 0 Track A |  |  |
|  |  | 15 | Encoder 0 Track B |  |  |
| Up to $2 \mathrm{~A} / \mathrm{B}$ encoders 200kHz possible with touch/reset standard inputs | PM5032-xETH, <br> PM5052-x- <br> ETH, <br> PM5072- <br> T-2ETH(W) <br> PM5082- <br> T-2ETH | 10 | Touch/Reset0 | OBIOEncoderCounter | The functionality uses up to the 4 digital fast inputs 200 kHz and the 45 kHz , the other inputs can be used for other purpose. Select encoder x track A for an input (14 or I7) automatically selects the adjacent input for B track |
|  |  | 11 | Touch/Reset0 |  |  |
|  |  | 12 | Touch/Reset1 |  |  |
|  |  | 13 | Touch/Reset1 |  |  |
|  |  | 14 | Encoder 0 <br> Track A |  |  |
|  |  | 15 | Encoder 0 <br> Track B |  |  |
|  |  | 16 | Encoder 1 <br> Track A |  |  |
|  |  | 17 | Encoder 1 <br> Track B |  |  |
|  |  | 18... 111 | Not relevant for the functionality |  |  |
| One A/B encoder 200 kHz with touch/reset | PM5032-xETH, <br> PM5052-xETH, <br> PM5072- <br> T-2ETH(W) <br> PM5082- <br> T-2ETH | 10 | Not relevant for the functionality | OBIOEncoderCounter | The functionality uses the 4 digital fast inputs 200 kHz , the other inputs can be used for other purpose. Select encoder x track A for the input (14) automatically selects the adjacent input for B track |
|  |  | 11 | Not relevant for the functionality |  |  |
|  |  | 12 | Not relevant for the functionality |  |  |
|  |  | 13 | Not relevant for the functionality |  |  |
|  |  | 14 | Encoder 0 <br> Track A |  |  |
|  |  | 15 | Encoder 0 <br> Track B |  |  |
|  |  | 16 | Touch/Reset |  |  |
|  |  | 17 | Touch/Reset |  |  |
|  |  | 18 ... 111 | Not relevant for the functionality |  |  |

## Example with one encoder

| Parameter <br> Run on config fault | Type <br> Enumeration of BYTE | Value No | Default Value <br> No | Unit | Description <br> Start PLC program even on configuration fault |
| :---: | :---: | :---: | :---: | :---: | :---: |
| -- Digital inputs 24 VDC |  |  |  |  |  |
| - Input 0, input delay | Enumeration of BYTE | 8 ms | 8 ms |  | Fast input 0 - Input delay |
| - Input 0, channel configuration | Enumeration of BYTE | Input/Interrupt | Input/Interrupt |  | Fast input 0 - Configuration (max. 5 kHz ) |
| - Input 1, input delay | Enumeration of BYTE | 8 ms | 8 ms |  | Fast input 1 - Input delay |
| - Input 1, channel configuration | Enumeration of BYTE | Input/Interrupt | Input/Interrupt |  | Fast input 1-Configuration (max. 5 kHz ) |
| Input 2, input delay | Enumeration of BYTE | 8 ms | 8 ms |  | Fast input 2 - Input delay |
| - Input 2, channel configuration | Enumeration of BYTE | Input/Interrupt | Input/Interrupt |  | Fast input 2 - Configuration (max. 5 kHz ) |
| - Input 3, input delay | Enumeration of BYTE | 8 ms | 8 ms |  | Fast input 3 - Input delay |
| - Input 3, channel configuration | Enumeration of BYTE | Input/Interrupt | Input/Interrupt |  | Fast input 3 - Configuration (max. 5 kHz ) |
| - Input 4, input delay | Enumeration of BYTE | 8 ms | 8 ms |  | Fast input 4 -Input delay |
| - Input 4, channel configuration | Enumeration of BYTE | Encoder 0 Track-A | Input |  | Fast input 4 - Configuration: Forward Counter (max. 100 kHz ), Encoder (max. 200 kHz ) |
| Input 5, input delay | Enumeration of BYTE | 8 ms | 8 ms |  | Fast input 5 - Input delay |
| -- Input 5, channel configuration | Enumeration of BYTE | Encoder 0 Track-B | Input |  | Fast input 5 - Configuration: Forward Counter (max. 100 kHz ), Encoder (max. 200 kHz ) |
| - Input 6, input delay | Enumeration of BYTE | 8 ms | 8 ms |  | Fast input 6 - Input delay |
| - Input 6, channel configuration | Enumeration of BYTE | Reset 0 | Input |  | Fast input 6 - Configuration: Forward Counter (max. 100 kHz ), Encoder (max. 200 kHz ) |
| Input 7, input delay | Enumeration of BYTE | 8 ms | 8 ms |  | Fast input 7 - Input delay |
| - Input 7, channel configuration | Enumeration of BYTE | Touch 0 | Input |  | Fast input 7 - Configuration: Forward Counter (max. 100 kHz ), Encoder (max. 200 kHz ) |

## Example with <br> two encoders

| Parameter <br> - Run on config fault | Type <br> Enumeration of BYTE |
| :---: | :---: |
| -- Digital inputs 24 VDC |  |
| - Input 0, input delay | Enumeration of BYTE |
| - Input 0, channel configuration | Enumeration of BYTE |
| Input 1, input delay | Enumeration of BYTE |
| - Input 1, channel configuration | Enumeration of BYTE |
| - Input 2, input delay | Enumeration of BYTE |
| * Input 2, channel configuration | Enumeration of BYTE |
| - Input 3, input delay | Enumeration of BYTE |
| - Input 3, channel configuration | Enumeration of BYTE |
| - Input 4, input delay | Enumeration of BYTE |
| - Input 4, channel configuration | Enumeration of BYTE |
| - Input 5, input delay | Enumeration of BYTE |
| - Input 5, channel configuration | Enumeration of BYTE |
| - Input 6, input delay | Enumeration of BYTE |
| - Input 6, channel configuration | Enumeration of BYTE |
| - Input 7, input delay | Enumeration of BYTE |
| - Input 7, channel configuration | Enumeration of BYTE |


| Value <br> No | Default Value <br> No | Unit | Description <br> Start PLC program even on configuration fault |
| :---: | :---: | :---: | :---: |
| 8 ms | 8 ms |  | Fast input 0 - Input delay |
| Reset 0 | Input/Interrupt |  | Fast input 0 - Configuration (max. 5 kHz ) |
| 8 ms | 8 ms |  | Fast input 1 - Input delay |
| Touch 0 | Input/Interrupt |  | Fast input 1 - Configuration (max. 5 kHz ) |
| 8 ms | 8 ms |  | Fast input 2 - Input delay |
| Reset 1 | Input/Interrupt |  | Fast input 2 - Configuration (max. 5 kHz ) |
| 8 ms | 8 ms |  | Fast input 3 - Input delay |
| Touch 1 | Input/Interrupt |  | Fast input 3 - Configuration (max. 5 kHz ) |
| 8 ms | 8 ms |  | Fast input 4-Input delay |
| Encoder 0 Track-A | Input |  | Fast input 4-Configuration: Forward Counter (max. 100 kHz ), Encoder (max. 200 kHz ) |
| 8 ms | 8 ms |  | Fast input 5 - Input delay |
| Encoder 0 Track-B | Input |  | Fast input 5 - Configuration: Forward Counter (max. 100 kHz ), Encoder (max. 200 kHz ) |
| 8 ms | 8 ms |  | Fast input 6 - Input delay |
| Encoder 1 Track-A | Input |  | Fast input 6 - Configuration: Forward Counter (max. 100 kHz ), Encoder (max. 200 kHz ) |
| 8 ms | 8 ms |  | Fast input 7 - Input delay |
| Encoder 1 Track-B | Input |  | Fast input 7 - Configuration: Forward Counter (max. 100 kHz ), Encoder (max. 200 kHz ) |

### 6.3.2.7.4 Configuration of interrupt inputs

Depending on the configuration for the input channels of the onboard I/O from the processor module different functionality are possible which must be used together with the dedicated function block of the user program.

| Functionality to be realized | Processor module type | Digital input channels | Configuration of the channel to be selected in Automation Builder | Dedicated function block to be used in the user program | Comments |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Interrupt inputs | $\begin{aligned} & \text { PM5012-x- } \\ & \text { ETH } \end{aligned}$ | I0 ... I3 | Input/Interrupt | OBIO InterruptInfo OBIOInterruptPara | Up to 4 interrupt input channels can be used, the other inputs can be used for other purpose |
|  |  | 14 ... 15 | Not relevant for the functionality |  |  |
|  | $\begin{aligned} & \text { PM5032-x- } \\ & \text { ETH, } \\ & \text { PM5052-x- } \\ & \text { ETH, } \\ & \text { PM5072- } \\ & \text { T-2ETH(W) } \\ & \text { PM5082- } \\ & \text { T-2ETH } \end{aligned}$ | I0 ... I3 | Input/Interrupt | OBIO InterruptInfo OBIOInterruptPara | Up to 4 interrupt input channels can be used, the other inputs can be used for other purpose |
|  |  | $14 \ldots 111$ | Not relevant for the functionality |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |



### 6.3.2.7.5 Creating an interrupt task

After configuring the parameter, the user needs to create a new task with the "Type" set to "External" and the "External event" set to "OnBoard_Binary_Input".


Please see "Use the onboard I/Os as interrupt input with dedicated interrupt task" in the system technology how to use the function. $\Rightarrow$ Chapter 6.8.2.15.3.4 "Use the onboard I/Os as interrupt input with dedicated interrupt task" on page 4550

### 6.3.2.7.6 Configuration of digital outputs

According to the processor module type, the digital outputs have several functionalities. To use them as digital output and as default configuration the following configuration is needed:


| [fif OnBoard_10 $\times$ |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 12DI/8DO-T/2DC Parameters | Parameter <br> - Run on config fault | Enumeration of ByTE | ValueNo | Default Value No | Unit | Description |
| 1201/8DO-T/2DC I/0 Mapping 12DI/8DO-T/2DC IEC Objects I/0 mapping list |  |  |  |  |  | Start PLC program even on configuration fault |
|  | +- Digital inputs 24 VDC |  |  |  |  |  |
|  | -- Digital outputs $24 \mathrm{VDC} / 0,5 \mathrm{~A}$ transistor |  |  |  |  |  |
|  | - Output 0 , channel configuration | Enumeration of BYTE | Output | Output |  | Fast output 0-Configuration (max. 5 kHz ) |
|  | - Output 1, channel configuration | Enumeration of BYTE | Output | Output |  | Fast output 1-Configuration (max, 5 kHz ) |
|  | - Output 2, channel configuration | Enumeration of BYTE | Output | Output |  | Fastoutput 2 - Configuration (max. 5 kHz ) |
|  | - Output 3, channel configuration | Enumeration of BYTE | Output | Output |  | Fast output 3 - Configuration (max. 5 kHz ) |
|  | - Outuut 4, channel configuration | Enumeration of BYTE | Limit Switch | Output |  | Fast output 4-Configuration: PWM (max. 100 kHz ), PTO (max. 200 kHz ) |
|  | - Output 5 , channel configuration | Enumeration of BYTE | Output | Output |  | Fast output 5 - Configuration: PWM (max. 100 kHz ), PTO (max. 200 kHz ) |
|  | - Output 6 , channel conffguration | Enumeration of BYTE | PWM | Output |  | Fast output 6 - Configuration: PWM (max. 100 kHz ), PTO (max. 200 kHz ) |
|  | - Output 7 , channel configuration | Enumeration of SYTE | рто0 | Output |  | Fast output 7-Configuration: PWM (max. 100 kHz ), PTO (max. 200 kHz ) |
|  | - Digital configurable In/uututs $24 \mathrm{VDC} / 0,5 \mathrm{~A}$ tansistor |  | PTO 0 |  |  |  |
|  | - Input / Output DC12, channel configuration | Enumeration of BYTE | Input/Output | Input/Output |  | Digital configurable channel 12 -Configuration |
|  | - Input / Output DC13, channel configuration | Enumeration of BYTE | Input/Output | Input/Output |  | Digital configurable channel 13 - Configuration |

### 6.3.2.7.7 Configuration of outputs as limit switch

The AC500-eCo V33 processor modules provide according to the output variants transistor or relay some output which can be used as limit switch. For the process modules with relay outputs, only the digital configurable channels provide this functionality.

| Functionality to be realized | Processor module type | Digital output channels | Configuration of the channel to be selected in Automation Builder | Dedicated function block to be used in the user program | Comments |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Limit switch | PM5012-T- ETH | O0 ... O3 | Limit Switch | OBIOLimitSwitch | Up to 4 limit switches |
|  | $\begin{aligned} & \text { PM5012-R- } \\ & \text { ETH } \end{aligned}$ | NO0 ... NO1 | Not relevant for the functionality |  | Relay outputs without other functions |
|  |  | NO2 ... NO3 | Not relevant for the functionality |  |  |
|  | PM5032-TETH, <br> PM5052-TETH, <br> PM5072- <br> T-2ETH(W) <br> PM5082- <br> T-2ETH | O0 ... O3 | Limit Switch | OBIO- <br> LimitSwitch | Up to 8 limit switches |
|  |  | O4 ... O7 | Limit Switch |  |  |
|  |  | C12 ... C13 | Not relevant for the functionality |  |  |
|  | PM5032-RETH, <br> PM5052-RETH | NO0 ... NO2 | Not relevant for the functionality | OBIOLimitSwitch | Relay outputs without other functions |
|  |  | NO3 ... NO5 | Not relevant for the functionality |  |  |
|  |  | C12 ... C13 | Limit Switch |  | Up to 2 limit switches |


| [fi OnBoard_10 x |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 12DI//8D-T/2DC Parameters | Parameter <br> - Run on config fault | Enumeration of bite | ValueNo | Default Value <br> No | Unit | Description |
| 12DI/8DO-T/2DC I/O Mapping 12DI/8DO-T/2DC IEC Objects I/O mapping list |  |  |  |  |  |  |
|  | +- Digital inputs 24 VDC |  |  |  |  |  |
|  | -- Digital outputs $24 \mathrm{VDC} / 0,5 \mathrm{~A}$ transistor |  |  |  |  |  |
|  | - Output 0 , channel configuration | Enumeration of BYTE | Output | Output |  | Fast output 0-Configuration (max. 5 kHz ) |
|  | - Output 1, channel configuration | Enumeration of BYTE | Output | Output |  | Fast output 1-Configuration (max. 5 kHz ) |
|  | - Output 2, channel configuration | Enumeration of BYTE | Output | Output |  | Fast output 2-Confguration (max. 5 kHz ) |
|  | - Output 3, channel configuration | Enumeration of BYTE | Output | Output |  | Fast output 3 - Confguration (max. 5 kHz ) |
|  | - Output 4, channel configuration | Enumeration of BYTE | Limit Switch | Output |  | Fast output 4-Configuration: PWM (max, 100 kHz ), PTO (max. 200 kHz ) |
|  | - Output 5 , channel configuration | Enumeration of BYTE | Output | Output |  | Fast output 5 - Confguration: PWM (max, 100 kHz ), PTO (max. 200 kHz ) |
|  | - Output 6 , channel conffguration | Enumeration of BYTE | PWM | Output |  | Fast output 6-Configuration: PWM (max. 100 kHz ), PTO (max. 200 kHz ) |
|  | - Output 7 , channel configuration | Enumeration of BYTE | рто0 | Output |  | Fast output 7-Confguration: PWM (max, 100 kHz ), PTO (max. 200 kHz ) |
|  | - Digital configurable In/outputs $24 \mathrm{vCC} / 0,5 \mathrm{~A}$ transistor |  | PTO 0 |  |  |  |
|  | - Input / Output DC 12, channel configuration | Enumeration of BYTE | Input/Output | Input/Output |  | Digital configurable channel 12 -Configuration |
|  | - Input / Output DC 13, hannel configuration | Enumeration of BYTE | Input/Output | Input/Output |  | Digital configurable channel 13 - Configuration |

### 6.3.2.7.8 Operating the limit switch output with user program

The OBIOLimitSwitch function block of the library must be used to operate the outputs with help of user program.

### 6.3.2.7.9 Configuration of PWM outputs (Pulse Width Modulation)

The AC500-eCo V3 processor modules provide up to 8 PWM output channels with a maximum frequency of 100 kHz ... 200 kHz . The parameter of PWM output channel of onboard I/O must be configured before it can be used. User should take these steps to configure the PWM output function.

| Functionality to be realized | Processor module type | Digital output channels | Configuration of the channel to be selected in Automation Builder | Dedicated function block to be used in the user program | Comments |
| :---: | :---: | :---: | :---: | :---: | :---: |
| PWM outputs | $\begin{aligned} & \text { PM5012-T- } \\ & \text { ETH } \end{aligned}$ | O0 ... O3 | PWM | OBIOPwm | Up to 4 PWM with 5 kHz |
|  | $\begin{aligned} & \text { PM5012-R- } \\ & \text { ETH } \end{aligned}$ | NO0 ... NO1 | Not relevant for the functionality |  | Relay outputs without other functions |
|  |  | NO2 ... NO3 | Not relevant for the functionality |  |  |
|  | PM5032-T- <br> ETH, <br> PM5052-T- <br> ETH, <br> PM5072- <br> T-2ETH(W) <br> PM5082- <br> T-2ETH | O0 ... O3 | PWM | OBIOPwm | Up to 4 PWM with 5 kHz |
|  |  | O4 ... 07 | PWM / PTOx LS Pulse |  | Up to 4 PWM with 100... 200 kHz |
|  |  | C12 ... C13 | Not relevant for the functionality |  |  |
|  | $\begin{aligned} & \text { PM5032-R- } \\ & \text { ETH, } \\ & \text { PM5052-R- } \\ & \text { ETH } \end{aligned}$ | NO0 ... NO2 | Not relevant for the functionality | OBIOPwm | Relay outputs without other functions |


| Functionality <br> to be realized | Processor <br> module type | Digital output <br> channels | Configura- <br> tion of the <br> channel to be <br> selected in <br> Automation <br> Builder | Dedicated <br> function <br> block to be <br> used in the <br> user program | Comments |
| :--- | :--- | :--- | :--- | :--- | :--- |
|  |  | $\mathrm{NO} \ldots \mathrm{NO}$ | Not relevant <br> for the func- <br> tionality |  |  |
|  |  | $\mathrm{C} 12 \ldots \mathrm{C} 13$ | PWM |  | Up to 2 PWM <br> with 100 kHz <br> only on these <br> channels |


6.3.2.7.10 Operating the PWM output with user program

The OBIOPwm function block of the library must be used to operate the PWM outputs with help of user program.
6.3.2.7.11 Configuration of PTO outputs (HW fast outputs for Pulse Train Output)

The AC500-eCo V3 processor modules provide up to 2 PTO hardware dedicated output channels with a maximum frequency of 200 kHz . The parameter of PTO output channel of onboard I/O must be configured before it can be used. User should take these steps to configure the PTO output function.

The PTO outputs can be used with 2 different modes either Pulse / Direction or Cc/Ccw mode.
Please refer to the technical data of the onboard I/Os ${ }^{*}>$ Chapter 5.2.1.1.2.7.8 "Technical data of the PTO outputs" on page 217.
The PTO channels are always requiring 2 consecutive output channels for the function.

| Functionality <br> to be realized | Processor <br> module type | Digital output <br> channels | Configura- <br> tion of the <br> channel to be <br> selected in <br> Automation <br> Builder | Dedicated <br> function <br> block to be <br> used in the <br> user program | Comments |
| :--- | :--- | :--- | :--- | :--- | :--- |
| PTO outputs | PM5012-T- <br> ETH | $\mathrm{O} 0 \ldots \mathrm{O} 3$ | Not possible | - | No PTO avail- <br> able |
|  | PM5012-R- <br> ETH | $\mathrm{NO} \ldots \mathrm{NO} 1$ |  |  |  |
|  |  |  |  |  |  |


| Functionality to be realized | Processor module type | Digital output channels | Configuration of the channel to be selected in Automation Builder | Dedicated function block to be used in the user program | Comments |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  | PM5032-T- <br> ETH, <br> PM5052-T- <br> ETH, <br> PM5072- <br> T-2ETH(W) <br> PM5082- <br> T-2ETH | O0 ... O3 | Not relevant for the functionality | OBIOPulseTrainOutput OBIOMotionPTO | Up to 2 PTO with Pulse/ Direction or Cw/Ccw mode |
|  |  | O4 ... 07 | PTOO HS Pulse/Cw -> O4 and automatically for O 5 also <br> PTO1 HS Pulse/Cw -> 06 and automatically for 07 also |  |  |
|  |  | C12 ... C13 | Not relevant for the functionality |  |  |
|  | PM5032-R- <br> ETH, <br> PM5052-R- | NOO ... NO2 | Not relevant for the functionality | OBIOPulseTrainOutput OBIOMo | Relay outputs without other functions |
|  |  | NO3 ... NO5 | Not relevant for the functionality | tionPTO |  |
|  |  | C12 ... C13 | PTO 1 -> C12 and automatically for C13 also |  | Up to 1 PTO with Pulse/ Direction or Cc/Ccw mode |



### 6.3.2.7.12 Operating the PTO hardware output with user program

The OBIOPulseTrainOutput function block of the library can be used to operate the PTO outputs with help of user program. This FB allows to control the output in PTO mode. The OBIOMotionPTO function block is a dedicated Motion control block to realize point-to-point movement or velocity control of a motion axis $\Rightarrow$ Chapter 6.8.2.15.3.6 "Use the onboard I/Os as pulse-train output (PTO)" on page 4553.
6.3.2.7.13 Configuration of SW PTO (PWM) outputs (HW fast outputs and standard outputs with software dedicated function block)

The AC500-eCo V3 processor modules could also provide up to 4 PTO (PWM) software output channels. To use that mode, the parameter of output channel of onboard I/O must be configured before it can be used. User should take these steps to configure this special PTO output function. The PTO outputs channels can be only used as Pulse / Direction mode. Please refer to the technical data ${ }^{\#}$ Chapter 5.2.1.1.2.7.8 "Technical data of the PTO outputs" on page 217 « ${ }^{4}$ Chapter 5.2.1.1.2.7.9 "Technical data of the PWM outputs" on page 217.
The PTO channel is using a digital fast output configured as PWM output to generate the Pulse output and a standard digital output to indicate the direction. A dedicated PTO motion block will then control the channel to realize the functionality.
Up to 4 PTO can be then provided each using 2 digital outputs.
A mixed configuration of one HW PTO channel (e.g. Output $04 \ldots 05$ ) and Pulse/Direction or $\mathrm{Cc} / \mathrm{Ccw}$ mode together with up to 2 other software PTO Channels (e.g. O6, O7 + dedicated output) and only Pulse/Direction mode is then possible. Please refer to the technical data $\star$ Further information on page 217.
To achieve such a software PTO mode the following channel configuration must be done.

| Functionality to be realized | Processor module type | Digital output channels | Configuration of the channel to be selected in Automation Builder | Dedicated function block to be used in the user program | Comments |
| :---: | :---: | :---: | :---: | :---: | :---: |
| PTO outputs (HW fast outputs PWM and software PTO) | $\begin{aligned} & \text { PM5012-T- } \\ & \text { ETH } \end{aligned}$ | O0 ... O3 | Not possible | - | No PTO available |
|  | PM5012-R- <br> ETH | NO0 ... NO1 |  |  |  |
|  |  | NO2 ... NO3 |  |  |  |
|  | $\begin{aligned} & \text { PM5032-T- } \\ & \text { ETH, } \\ & \text { PM5052-T- } \\ & \text { ETH, } \\ & \text { PM5072- } \\ & \text { T-2ETH(W) } \\ & \text { PM5082- } \\ & \text { T-2ETH } \end{aligned}$ | O | Output / PTOO Dir | OBIOMotionPWM | The 4 software PTO channels will use the fast outputs O4... O7 PWM to generate the Pulse signal of each SW PTO and the outputs O0... O3 will generate the direction signals |
|  |  | 01 | Output / PTO1 Dir |  |  |
|  |  | O 2 | Output / PTO2 Dir |  |  |
|  |  | O3 | Output / PTO3 Dir |  |  |
|  |  | O4 | PWM / PTOO LS Pulse or CPU revision 1 PTO HS Single 0 |  |  |
|  |  | O5 | PWM / PTO1 LS Pulse or CPU revision 1 PTO HS Single 1 |  |  |
|  |  | 06 | PWM / PTO2 LS Pulse or CPU revision 1 PTO HS Single 2 |  |  |


| Functionality to be realized | Processor module type | Digital output channels | Configuration of the channel to be selected in Automation Builder | Dedicated function block to be used in the user program | Comments |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | 07 | PWM / PTO3 LS Pulse or CPU revision 1 PTO HS Single 3 |  |  |
|  |  | C12 ... C13 | Not relevant for the functionality |  |  |
|  | PM5032-R- <br> ETH, <br> PM5052-R- <br> ETH | NO0 ... NO2 | Not relevant for the functionality | OBIOMotionPWM | Relay outputs without other functions |
|  |  | NO3 ... NO5 | Not relevant for the functionality |  |  |
|  |  | C12 ... C13 | PWM |  | Theoretically possible up to 2 software PTO with Pulse/Direction but need an additional digital ouput module for the direction signal |

## 2 PTO Cw/Ccw



| (fifl OnBoard_10 $\times$ |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 12DI/8D0-T/2DC Parameters | Parameter | Type | Value | Default Value | Unit | Description |
| 12DI/8DO-T/2DC I/O Mapping | - Run on confg fault | Enumeration of BYTE | No | No |  | Start PLC program even on configuration fault |
|  | *-D Digital inputs 24 VDC |  |  |  |  |  |
| 12DI/8DO-T/2DC IEC Objects | -- Digital outputs $24 \mathrm{VDC/} / 0,5 \mathrm{Stansistor}$ |  |  |  |  |  |
|  | - Output 0 , channel configuration | Enumeration of BYTE | Outut / PTOO Dir | Output/PTOO Dir |  | Fast output 0-Configuration: Output/PTOO Dir, Limit switch, PWM (max. 5 kHz ) |
| I/O mapping list | - Output 1, channel configuration | Enumeration of BYTE | Output/PTO1 Dir | Output/PTO1Dir |  | Fast output 1 - Configuration: Output/PTO1 Dir, Limit switch, PWM (max. 5 kHz ) |
|  | - Output 2, channel configuration | Enumeration of BYTE | Output / PTO2 Dir | Output/PTO2Dir |  | Fast output 2 - Configuration: Output / PTO2 Dir, Limit switch, PWM (max. 5 kHz ) |
|  | - Output 3, channel configuration | Enumeration of BYTE | Output/PTO3 Dir | Output/PTO3 Dir |  | Fasto output 3 - Configuration: Output/PTO3 Dir, Linit switch, PWM (max. 5 kHz ) |
|  | - Output 4 , channel configuration | Enumeration of BYTE | Outut $\checkmark$ | Outut |  | Fast output 4-Configuration: Output, Linit switch, PWM / PTOO LS Pulse (max. 100 kHz ), P.. |
|  | - Output 5, channel configuration | Enumeration of BYTE | Output | Output |  | Fast output 5 - Configuration: Output, Linit switch, PWM / PTO1LS Pulse (max. 100 kHz ), P.. |
|  | - Output 6 , channel configuration | Enumeration of BYTE | Limit Switch | Outut |  | Fast output 6-Configuration: Output, Linit switch, PWM / PTO2 LS Pulse (max. 100 kHz ), P.. |
|  | - Outuut 7 , channel configuration | Enumeration of BYTE | PTOO HS Puse/ ${ }^{\text {cm }}$ | Output |  | Fast output 7 - Configuration: Output, Linit switch, PWM / PTO3 LS Pulse (max. 100 kHz ), P... |
|  | *- Digital configurable In/outputs $24 \mathrm{vDC} / 0,5 \mathrm{At}$... |  | Pro HS Single 0 |  |  |  |


| (fill OnBoard_Io x |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 120I/800-T/20C Parameters | Parameter <br> - Run on config fault | Type | Value | Default Value | Unit | Description |
| 120I/800-T/2DC I/O Mapping |  | Enumeration of BYTE | No | No |  | Start PLC program even on configuration fault |
|  | - - Digital inputs 24 VDC |  |  |  |  |  |
| 12DI/800-T/20C IEC Objects | -- Digital outputs $24 \mathrm{VDC} / 0,5 \mathrm{~A}$ transistor |  |  |  |  |  |
|  | - Output 0 , channel configuration | Enumeration of BYTE | Output/PTOOI $\sim$ | Output/PTOO Dir |  | Fast output 0-Configuration: Output/PTOO Dir, Limit switch, PWM (max. 5 kHz ) |
| I/O mapping list | - Output 1, channel conffguration | Enumeration of BYTE | Output/PTOOD ${ }^{\text {dit }}$ | Output/PTO1 Dir |  | Fast output 1-Configuration: Output/PTO1 Dir, Limit switch, PWM (max. 5 kHz ) |
|  | - Output 2, channel conffguration | Enumeration of BYTE |  | Output/PTO2Dir |  | Fast output 2 - Conffguration: Output/PTO2 Dir, Limit switch, PWM (max. 5 kHz ) |
|  | - Output 3, channel conffguration | Enumeration of BYTE | Outut/PTO3Dir | Output/PTO3 Dir |  | Fast output 3 - Configuration: Output/PTO3 Dir, Limit switch, PWM (max. 5 kHz ) |
|  | - Output 4, channel configuration | Enumeration of BYTE | Output | Ouput |  | Fast output 4-Configuration: Output, Limit switch, PWM / PTOO LS Pulse (max. 100 KHz ), P.. |
|  | - Output 5 , channel configuration | Enumeration of BYTE | Output | Output |  | Fast output 5 - Conffguration: Output, Limit switch, PWM / PTO1 LS Pulse (max. 100 kHz ), P.. |
|  | - Output 6, channel configuration | Enumeration of BYTE | Output | Ouput |  | Fast output 6 - Configuration: Output, Limit switch, PWM / PTO2 L. Pulse (max. 100 kHz ), P.. |
|  | - Output 7 , channel configuration | Enumeration of BYTE | Output | Outut |  | Fast output 7 - Configuration: Output, Limit switch, PWM / PTO3 L. Puilse (max. 100 KHz ), P.. |
|  | $\pm$ - Digital configurable In/outputs $24 \mathrm{VDCC} / 0,5 \mathrm{t}$... |  |  |  |  |  |

PTO on CPU with Relay


### 6.3.2.7.14 Operating the software PTO output channels with user program

The OBIOMotionPWM function block is a dedicated motion control block to realize point-topoint movement or velocity control of a motion axis. This block will then control the output channels as PTO mode Pulse/Direction only up to 100 kHz . See the dedicated chapter in the system technology ${ }^{\leftrightarrows}$ Chapter 6.8.2.15.3.6 "Use the onboard I/Os as pulse-train output (PTO)" on page 4553.

### 6.3.2.8 Option board for processor modules PM50xx

### 6.3.2.8.1 Select the option board

Depending on processor module type, up to 3 option board slot are available and for each several option board modules are available.


To add an option board on the processor module, select the desired OptionSlot to be configured and attach the needed option board from the list.

There is no limitation on the number of same option board used on the CPU, no dedicated slot for a specific function and no specific order to place the option board module.
Depending on the type of option board selected and attached to the CPU some further configuration of channels or function may be needed.

### 6.3.2.8.2 Attach an option board for digital I/O extension

```
us Project$peoject' - Actomation Bulder 2.3 - Premium
Ele Idit View Project Byidd goline Rebug Jools Widow Help BaCNet
```




Just select and attached the module, no other channel configuration is needed. The I/O channels are directly mapped in the I/O mapping and variables can then be defined.

### 6.3.2.8.3 Attach an option board for analog I/O extension

The desired analog I/O option board type for the desired option board slot must be selected and added.

The option board may require some other channel configuration according to your need.
Following example shows how to add a TA5120-2AI-UI analog option board and the desired channel types which can be selected.

| [] PLC_AC500_V3 10 TA5120_2AI_UI $\times$ |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| TA5120-2AI-UI Parameters | Parameter <br> - Run on config fault | Type <br> Enumeration of BYTE | Value No |  | Default Value <br> No | Unit | Description |
|  |  |  |  |  | Start PLC program even on configuration fault |  |
| 5120-2AI-UI I/O Mapping | - Check supply | Enumeration of BYTE |  | On |  | On |  | Check supply |
| I/0 mapping list | - Input 0 , channel configuration | Enumeration of BYTE | Not used | $\checkmark$ | Not used |  | Analog input 0-Configuration of analog input channel |
| I/O mapping list | - Input 0 , check channel | Enumeration of BYTE | Notused |  | Plausib, Cut wire, Short dircuit |  | Analog input 0-Check channel |
| TA5120-2AI-UI IEC Objects | - Input 1, channel configuration | Enumeration of BYTE | ${ }^{0} 0.10 .10 \mathrm{~V}$ |  | Not used |  | Analog input 1-Configuration of analog input channel |
|  | - Input 1, check channel | Enumeration of BYTE | $4 . .20 \mathrm{~mA}$ |  | Plausi, Cut wire, Short circuit |  | Analog input 1-Check channel |
| Diagnosis |  |  |  |  |  |  |  |
| Information |  |  |  |  |  |  |  |

### 6.3.2.8.4 Attach an option board for COMx serial communication

The desired serial interface option board type for the desired option board slot must be selected and added.
The option board may require some other channel configuration according to your need.
Following example shows how to add a TA5142-RS485I isolated interface and the desired protocol.





The desired protocol has also to be added according to your needs, e.g. Modbus RTU client:


The parameter for the serial interface can also be adapted like baudrate, data bit, stop bit or parity.


### 6.3.2.9 Onboard Ethernet configuration

### 6.3.2.9.1 General

Onboard Ethernet is provided for device types with -ETH extension.

### 6.3.2.9.2 Configuration of the IP settings with the IP configuration tool

 IntroductionThe IP configuration tool can be used

- to set or change the IP address of devices.
- to scan the network for available hardware devices.
- to update the firmware of devices. This functionality is only supported if the IP configuration tool is used stand-alone.
- to activate certain functionality on hardware devices.

This feature is only available on AC500 V3 devices.
The IP configuration tool is part of Automation Builder and can be called via "Tools $\rightarrow$ IP-Configuration".
Further the IP configuration tool can be used stand-alone without an Automation Builder application running. The stand-alone variant requires a separate installation via the Installation Manager $\Leftrightarrow$ Chapter 6.3.2.9.2.2 "Stand-alone installation" on page 1507.
After the installation, the IP configuration tool is started via .exe file / desktop icon.

Some functionality is only supported if the IP configuration tool is used standalone, e.g. for firmware updates for communication interface devices.

## Stand-alone installation

The IP configuration tool is part of Automation Builder and can be called via "Tools $\rightarrow$ IP-Configuration". A separate installation is only required if the IP configuration tool shall be used stand-alone.

1. Open the Installation Manager in Automation Builder: "Tools $\rightarrow$ Installation Manager".
2. Close all other instances of Automation Builder as only one instance of the program can be executed at a time.

3. Click [Modify] and select the "IP Configuration Tool" from the structure tree.

| 能 ABB Automation Builder 2.4.0 Build 928 - Selection Page | - | $\square$ | $\times$ |
| :---: | :---: | :---: | :---: |
| Autometion Builder |  |  |  |

## Select type of installation:

Premium Edition
Select/Unselect to Install/Uninstall:

```
\(\square \vee\) ABB Automation Builder
    +--\square Professional Tool Chain
    \square-\vee\squarePLC
    #
    \pm}\downarrow\square\mathrm{ Drives
\square- - Tools
    \square-\squareAC500-Tools
    - - - CoDeSys OPC Server 3.5
        - \square CoDeSys OPC Server 3.5
        \checkmark IP Configuration Tool
            - - MultiOnlineChange Tool
    \square-
    --\square HMI
        - }\square\mathrm{ Panel Builder }600\mathrm{ Runtime for PC
        - Control Panel - CP600
        - Automation Builder Standalone Tools
        ABB License Manager
        - }\square\mathrm{ Safety PLC - Pluto
```

    V Install also previous product versions
    \(\square\) Automation Builder 2.3
    \(\square\) Automation Builder 2.2
    Automation Builder 2.1
    | Description | Release Notes | Packages |
| :--- | :--- | :--- |

IP Configuration Tool
The standalone tool for IP address configuration

This software package contains the following main components and features:

- ABB IP Configuration Tool
- Not Installed Up to date


4. Click [Continue] to start the installation.
$\Rightarrow$ After a successful installation the IP configuration tool is available as stand-alone tool (.exe).
$\Rightarrow$ To start the IP configuration tool, click the new created desktop icon.

## Network scan

With a network scan all devices that have been found in the network by the scan process are listed, i.e. ABB devices such as AC500 processor modules, AC500 communication interface modules or $A B B$ drives.

1. Start the IP configuration tool in Automation Builder ("Tools $\rightarrow$ IP-Configuration") or start it stand-alone (.exe).
2. The "IP-Configuration" dialog opens. Define the device type for the network scan by selecting the desired option under "Scan Protocol":

- "ABB Net config protocol":

Use this option for AC500 devices such as processor modules, CI5xx-Modbus devices or ABB drives. The device(s) to be scanned must be connected to the PC via a direct Ethernet connection.

- "Profinet Dynamic Configuration Protocol (DCP)":

Use this option for PROFINET communication interface modules. The device(s) to be scanned must be connected to the PC via a direct Ethernet connection (not via CM579).
For the scan, a NPcap driver needs to be installed separately.

- "EtherCAT":

Use this option for EtherCAT communication interface modules. The Ethernet cable must be connected directly to the first EtherCAT slave device of the EtherCAT fieldbus. Ensure that no EtherCAT master device is available on the bus when a scan is performed.
"Emergency" option: Enable this option to check on failures in the EtherCAT assembly during the scan process, i.e. a frame loss or interchanged ports. Errors are displayed. For the scan, a NPcap driver needs to be installed separately.
3. Click [Scan] to start the scan process.

4. All devices that have been found in the network are listed including hardware and connection details. The following details can be changed under "IP settings":
$\Rightarrow$ - "IP Address":
Current IP address of the device.

- "Conf. IP Address":

Configured IP address of the device. A changed IP address will update this column.

- "FW Version":

Current installed firmware version of the device. This field is visible not until a first network scan. If this field is still empty after a network scan, check on connection errors.

The IP address of some devices, e.g. EtherCAT devices cannot be changed.

## Changing the IP address

1. In order to change the IP address of devices perform a network scan.
${ }^{\Downarrow}{ }^{\Downarrow}$ Chapter 6.3.2.9.2.3 "Network scan" on page 1509
2. Select a device from the list and select the appropriate protocol under "Scan protocol".
"DHCP" or "BOOTP" option: If required, DHCP or BOOTP can be used to receive the IP address for the device from the server.
"IP address", "subnet mask", "Std. gateway": Use these fields to change the IP address settings including the settings for the subnet mask and the standard gateway. Ensure that the combination of connection settings is correct.

## Note for CI52x-Modbus devices

Consider the behavior of CI52x-Modbus devices if the last number of the IP address is set to " 0 ".
«2, "Check last number of IP address" on page 1512
3. Change the settings for the IP configuration and click [Send settings] to transmit the data to the device.

$\Rightarrow$

## Note for PROFINET devices

The device name of PROFINET devices can be edited. If changing the name, ensure the following rules apply:

- Labels must be separated by "."
- Total length: 1 to 240
- Label length: 1 to 63
- Labels can consist of characters [a-z] and numbers [0-9]
- Labels are not allowed to start with "-"
- Labels are not allowed to end with "-"

4. In order to keep all IP changes after a power cycle, the settings can be stored permanently. Confirm the prompted message during the scan process.

Check last number of IP address

This hint is only valid for Cl 52 x -Modbus devices.
Check the last number of the IP address. If it is set to "0", the IP address setting for this last number will be used from the rotary switches on the hardware device.

## Example:



## Firmware update

The firmware of AC500 communication interface modules can be updated with the IP configuration tool.

For this, the IP configuration tool must be used as stand-alone variant.
(y) Chapter 6.3.2.9.2.2 "Stand-alone installation" on page 1507

It is not possible to perform a firmware update out of Automation Builder.

- For PROFINET communication interface modules a firmware update is only supported for devices with firmware version $\geq 3$ 3.3.3.
- For EtherCAT communication interface modules a firmware update is only supported for devices with firmware version $\geq 2.1$.4.
- For Modbus communication interface modules a firmware update is only supported for devices with firmware version $\geq 3.2 .13$.


## Requirements: Before the firmware update

- Ensure a fast and stable network connection
- Close all unused applications on the executing PC
- Stop the communication between AC500 PLC and the communication interface module that shall be updated


## During the firmware update

- Do not close the IP configuration tool
- Do not open Automation Builder software or any other application
- Do not switch-off the communication interface module that shall be updated
- Do not disconnect the Ethernet connection of a communication interface module or the executing PC

The firmware update will stop the operation of the affected device(s). Hence, the device(s) will become unresponsive for 1-2 minutes.

## Procedure:

1. Start the IP configuration tool stand-alone (.exe).
2. Perform a network scan.
${ }^{\mu}$ ) Chapter 6.3.2.9.2.3 "Network scan" on page 1509
3. Select the devices that shall be updated from the list and click [Scan] to trigger the scan process.
A multiple selection of several devices is possible via control key, however, ensure to select only devices of the same protocol at a time. Otherwise the firmware update fails.
4. This step is only required for devices that require an installed NPcap driver. In this case an appropriate message including a download link is prompted in the IP-Configuration dialog:

$\Rightarrow$ Click on the displayed link https://nmap.org/download.html and download the latest version of the npcap-X.X.exe file.


For those who prefer the command-line zip files (Installation Instructions; Usage Instructions), they are still available. The Zenmap graphical interface is nor includec from a DOS /command window. Or you can download and install a superior command shell such as those included with the free Cygwin system. Also, you need to ru: 2013 Redistributable Package installers which are included in the zip file. The main advantage is that these zip files are a fraction of the size of the executable installe

Latest stable command-line zipfile: nmap-7.91-win32.zip

## Linux RPM Source and Binaries

Many popular Linux distributions (Redhat, Mandrake, Suse, etc) use the RPM package management system for quick and easy binary package installation. We have * RPM packages, though these simple commands usually do the trick:

$\Rightarrow$ After the download, execute the file as administrator and restart the scan process.
$\Rightarrow$ The devices that have been scanned are listed.
5. Click [Update Firmware] to start the firmware update for the selected devices.

6. For $\mathrm{CI} 50 \mathrm{x}, \mathrm{CI} 51 \mathrm{x}$ and CI 52 x devices a signature check is started. Select the appropriate firmware update file (*.bin) for the device(s). Example: C:\AC500\AC500_CI52x_Firmware_V3.2.8.bin.

After a successful signature check the firmware update file (*.bin) and the respective signature file (*.bin.sig) are transferred to the device. This can last up to 3 minutes.

If the signature check fails, check the availability of the *.bin file and the *bin.sig file.
7. A status check followed by a device reboot followed by a second status check is performed automatically.

After the firmware update all outputs of the updated devices are set to ' 0 '.
8. After a successful firmware update the update status or the new firmware version is displayed in the "FW Version" field.
If this field is empty, there possibly is a connection error between the device and the executing PC.

Exception: For EtherCAT devices an empty "FW Version" field does not indicate a connection error.

$\Rightarrow$ If the firmware update fails

- check the requirements for the update procedure.
"Requirements:" on page 1513
- check the hints for trouble-shooting.
- perform a network scan and repeat the update. If the error still persists power cycle the device and try the update again.


## Blink functionality

This function activates flashing of the backlight of an AC500 LED display.

1. From the menu, select "Tools $\rightarrow$ IP-Configuration".
2. Click [Scan] to trigger the scan process for devices in the network.
$\Rightarrow$ A progress bar shows the progress. The IP settings of a selected device is displayed below the list and can be edited.
3. Adjust your desired time and click [Blink] to activate flashing.


## Troubleshooting for IP configuration tool

Firewall excep- On a standard Windows 7 installation without third party firewall or security tools installed the IP tions: configuration tool should work properly.

The Automation Builder setup installs rules or exceptions for the built-in Windows firewall to allow IPConfig to receive the responses for the IPConfig scan.
To check the Windows firewall is set correctly check the firewall settings.

Windows 7I On the network that is used for communication with the PLC, set "Incoming connections" to Windows 10:

Networks at home or work where you know and trust the people and devices on the network

| Windows Firewall state: | On |
| :--- | :--- |
| Incoming connections: | Block all connections to programs that are not on the <br> list of allowed programs |
| Active home or work (private) networks: | Network |
| Notification state: | Notify me when Windows Firewall blocks a new <br> program |

Networks in public places such as airports or coffee shops

Windows Firewall state:
Incoming connections:

Active public networks:
Notification state:

On
Block all connections to programs that are not on the list of allowed programs


Unidentified network
Notify me when Windows Firewall blocks a new program

If a third party firewall is used these exceptions must be configured manually.

$$
\begin{aligned}
& \text { Either exceptions for applications can be entered: Automation Builder and IP } \\
& \text { configuration tool must be added as application. } \\
& \text { Or the protocol and the port number must be given (for IPConfig: UDP protocol } \\
& \text { and port number } 24576 \text { ). }
\end{aligned}
$$

| Fault description | Cause | Remedy |
| :--- | :--- | :--- |
| Error - general |  | Ensure that all requirements have been con- <br> sidered before and during the update proce- <br> dure. |
| The "FW Version" field <br> is empty after the net- <br> work scan or the firm- <br> ware version has not <br> been updated after the <br> update procedure (only <br> for Modbus devices and <br> PROFINET devices) | Connection error between the device <br> and the executing PC. |  |


| Fault description | Cause | Remedy |
| :--- | :--- | :--- |
| Signature check failed | During the firmware update of CI50x, <br> Cl51x and CI52x devices a signature <br> check is started. The update proce- <br> dure expects a firmware update file <br> (*.bin) and a signature file (*.bin.sig) <br> in the same directory. Without a signa-- <br> ture file the signature check will fail. | Example: <br> Firmware update file: <br> C:\AC500\AC500_CI52x_Firmware_V3.2.8.bin <br> Signature file:C:VAC500\AC500_CI52x_Firm- <br> ware_V3.2.8.bin.sig |
| Error: Package timeout | A timeout error may occure due to an <br> instable network. | Keep the executing PC as near as possible <br> to the devices that shall be updated. Avoid net- <br> work switches. |
| Error: Unable to read <br> device status | A read error may occure due to errors <br> in the firmware update protocol. After <br> the firmware update the IP configura- <br> tion tool reads out the status of the <br> updated device in order to check if the <br> update was successful. | Perform the update again. |
| Error: IP is not unique | If an IP address is obtained by more <br> than one device an error occures. A <br> firmware update is not possible. | Change the IP addresses. |
| Error: Error State | Internal device error during the firm- <br> ware update. | Increase the connection quality. Keep the exe- <br> cuting PC as near as possible to the devices <br> that shall be updated. Avoid network switches. |
| Error: Can't connect to <br> device | The TCP communication is not suffi- <br> cient. |  |

### 6.3.2.9.3 Switch functionality of Ethernet interfaces ETH1/ETH2

As of SystemFW 3.1.0 the Ethernet interfaces ETH1/ETH2 can be configured as an Ethernet switch.

The default setting is "Two separate interfaces".


The change of the PLC Boot parameter ETH1 / ETH2 mode will become active after a PLC reboot.

Create and download a Boot project before rebooting the PLC!

| Parameter | Value | Description |
| :--- | :--- | :--- |
| ETH1 / ETH2 mode | Two separate interfaces | Two separate Ethernet inter- <br> faces ETH1 and ETH2 |
|  | Switch functionality ETH1- <br> ETH2 | Switch between ETH1 and <br> ETH2 |



If the Switch functionality ETH1-ETH2 is active, only the Ethernet inter-
face ETH1 is available (see ${ }^{\wedge}$ Chapter 6.8.2.5.5.5 "CFG - configuration" on page 4462). Any protocols configured under Ethernet interface ETH2 must be deleted. Otherwise a compile error will be created.

The setting of ETH1 / ETH2 mode can be checked on LED display with soft key <CFG> (see \& Chapter 6.8.2.5.5.5 "CFG - configuration" on page 4462).

### 6.3.2.10 Onboard CAN configuration

AC500 V3 PLCs provide the following methods for CAN integration:

- Onboard CAN interface
- CANopen master-slave arrangement (with CM598-CN as a master device)

Table 300: Differences in supported protocols

|  | Onboard CAN | CM598-CN |
| :--- | :--- | :--- |
| CANopen Manager | X | X |
| CAN 2A/2B | X | X |
| J1939 | X |  |

Onboard CAN interface is not available on AC500-eCo V3!

Supported pro- Onboard CAN interface supports the following protocols
tocols

- CANopen Manager: Connection of CI581 and CI582 without additional I/O modules
- CAN 2A/2B
- J1939

Configuration in Automation Builder is described in chapter 'CANopen' ${ }^{*} \Rightarrow$ Chapter 6.3.2.11.1.1
"CM598-CAN - CANopen Manager communication module" on page 1521.
Further information can be found in chapter 'CAN onboard' $\Leftrightarrow$ Chapter 6.3.2.16 "CAN onboard" on page 1594

### 6.3.2.11 Communication modules

### 6.3.2.11.1 CANopen

CM598-CAN - CANopen Manager communication module

## Configuration of the communication module

$$
\begin{aligned}
- & \text { Click menu "Tools } \rightarrow \text { Options" and select "Device editor" in the "Options" } \\
& \text { window. } \\
- & \text { Enable first checkbox "Show generic device configuration views" and click } \\
& {[\text { OK]. }}
\end{aligned}
$$

Append a CM598-CAN

1. Right-click on your desired Slot below node "Extension_Bus" and click "Add object". $\Rightarrow$ Dialog "Replace object:" appears.
2. Click "CM598_CAN" in the list and click [Replace object].
3. Double-click "CM598_CAN (CM598-CAN)" to get the "CM598-CAN Parameters" in the editor window.

The following parameters are available:

| Parameter | Default value | Value | Description |
| :--- | :--- | :--- | :--- |
| Run on config <br> fault | No | No | In case of a configuration error, the <br> user program is not started. |
|  |  | Yes | The user program is started inde- <br> pendent of a faulty configuration <br> of the CM598-CAN communication <br> module. |


| Parameter | Default value | Value | Description |
| :--- | :--- | :--- | :--- |
| CANopen Sync <br> mode | Sync Bus only | Sync Bus only | The "CANopen Manager" sends sync <br> messages (if enabled) to CAN bus <br> only. The properties of the sync mes- <br> sage are configured in the "CANopen <br> Manager" parameters. |
|  |  | Sync Bus and <br> task | "CANopen Manager" sends sync <br> messages to CAN bus and triggers <br> IEC task. Chapter 6.3.2.11.1.1.4 <br> "Configuration of SYNC task" <br> on page 1532 |

The tab "CAN Bus" contains the basic settings of the CAN bus and special settings for the CAN 2.0 B protocol.

The settings at "29 Bit COB-ID" are only valid for CAN 2.0 B protocol. Ensure the option "Enable 29 Bit COB-ID" is enabled. Otherwise no CAN 2.0 B frames can be received. With the other parameters at "Enable 29 Bit COB-ID" the receive filter is configured.

SAE J1939 An example project which demonstrates using and handling the SAE J1939 protocol in AC500 V3 PLCs is given in the application example AC500 SAE J1939 protocol.

| Parameter | Default value | Value | Description |
| :---: | :---: | :---: | :---: |
| Bus parameters |  |  |  |
| Transmission rate | $250 \mathrm{kBit} / \mathrm{s}$ | $10 \mathrm{kBit} / \mathrm{s}$ <br> $20 \mathrm{kBit} / \mathrm{s}$ <br> 50 kBit/s <br> $100 \mathrm{kBit} / \mathrm{s}$ <br> 125 kBit/s <br> 250 kBit/s <br> $500 \mathrm{kBit} / \mathrm{s}$ <br> 800 kBit/s <br> 1000 kBit/s | Transmission speed in [kBit/s] |
| Node settings |  |  |  |
| Stop in case of monitoring error | Disabled | Disabled | The manager does not stop in case of a monitoring error (Node Guarding or Heartbeat Error). A loss of communication to one node has no influence to other nodes. The manager tries to reestablish the communication to the error affected nodes. |
|  |  | Enabled | If this function is enabled, the manager will also stop the communication to all responding and active nodes. |
| Send "Global Start Node" | Enabled | Disabled | No "Global Start Node" message is sent after configuring the nodes |
|  |  | Enabled | A "Global Start Node" message is sent after configuring the nodes. This synchronize all nodes again. |
| 29 Bit COB-ID |  |  |  |
| Enable 29 bit COB-ID | Disabled | Disabled | 29 bit CAN-IDs are disabled, but 11 bit CAN-IDs are still enabled. |
|  |  | Enabled | 29 bit CAN-IDs are additional enabled. |


| Parameter | Default value | Value | Description |
| :--- | :--- | :--- | :--- |
| Acceptance mask | 0 | 29 bit | Specifies the bits of a <br> CAN-ID which will be <br> evaluated by the filter. <br> For instance, with an <br> acceptance mask <br> Ox1FFFFFFF all bits <br> are evaluated. |
| Acceptance code | 0 | 29 bit | Specifies the bits of <br> a CAN-ID which has <br> to be set to pass the <br> filter. Only those bits <br> which are set in the <br> acceptance mask are <br> relevant. |

The tab "CM598-CAN I/O Mapping" contains bus cycle option used for CANopen configuration. The following parameters are available:

| Parameter | Default value | Value | Description |
| :--- | :--- | :--- | :--- |
| Bus cycle task | Use parent bus cycle <br> setting | Use parent bus cycle <br> settings | Settings from "PLC <br> settings" tab are used. |
|  | Task | Name of the task that <br> triggers the bus cycle |  |

The tab "CM598-CAN IEC Objects" contains the created instance of the IO driver.

| CM598-CAN Parameters | Add... | Edit... $\times$ Delete $\rightarrow$ Go to variable |  |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| CAN Bus | Variable | Mapping | Type |
| CM598-CAN IEC Objects | CM598_CAN |  |  |

## Configuration of the CANopen Manager

By adding "CM598-CN 'to configuration, "CANopen Manager" is added automatically. Double click at "CANopen Manager" to show related parameters/information in a new window.
The tab "General" contains the following parameters:

| Parameter | Default value | Value | Description |
| :--- | :--- | :--- | :--- |
| General |  |  |  |
| Node-ID | 1 | $1-127$ | Unique node id for "CANopen <br> Manager". |


| Parameter | Default value | Value | Description |
| :--- | :--- | :--- | :--- |
| Check and fix <br> configuration |  |  | This button opens the dialog that <br> allows checking and fixing the cur- <br> rent configuration. |
| Autostart <br> "CANopen <br> Manager" |  |  | Not supported ("CANopen Manager" <br> always starts automatically). |
| Polling of <br> optional slaves |  |  | Not supported |
| Start slaves |  |  | Not supported |
| NMT start all (if <br> possible) |  |  | Not supported |
| NMT error <br> behavior |  |  | Not supported |
| Guarding |  |  | Disabled |
| Enable hearbeat <br> producing | Disabled | The heartbeat generation is dis- <br> abled. Nodeguarding is enabled <br> instead. Note that CANopen remote <br> devices can also be configured as <br> heartbeat producers. |  |


| Parameter | Default value | Value | Description |
| :--- | :--- | :--- | :--- |
| COB-ID (Hex) |  |  | Not supported |
| Producer time <br> (ms) |  |  | Not supported |

Dialog "Check and fix Configuration"
If you insert several devices below the "CANopen Manager", then error messages may report multiple assigned Node-IDs or invalid COB-IDs. The "Check and Fix Configuration" button opens a dialog for solving these conflicts.
For conflicts with Node-IDs or PDO COB-IDs, you can click "Edit Conflicts" to open a dialog with detailed information.

| "Doubled node number" | List of all devices with identical IDs. In the field <br> of the "Node-ID" column, you can enter new <br> node numbers for the affected devices. |
| :--- | :--- |
| "Incorrect and double assignment of PDO | The COB-IDs that are generated automatically <br> from the device description files may not be <br> permitted. All incorrect entries are listed with <br> the respective device names, Node-IDs, and <br> indexes. There are three options for correcting <br> invalid COB-IDs: <br> -Correct the displayed formula for calcu- <br> lating the COB-IDs so that a valid COB-ID <br> results. You can change the formula in the <br> respective table element. <br> - Accept the automatic suggestion for the <br> COB-ID by clicking the respective button. |
| - Accept all automatic suggestions by |  |
| clicking the "Use Suggested COB-ID" |  |
| button. |  |

Corrected entries are removed from the displayed list automatically.
Solve timing problems automatically by using the "Automatic Repair". The command modifies all timing values to compatible values. (The time should be a multiple of the task time.)
The tab "Live list" can be used to check CANopen network for devices. The tab is only shown in online mode. If communication module is up and running after clicking button "Scan" all remote devices are listed showing their node-Ids and types.

## Configuration of the CANopen remote devices

1. To add CANopen remote devices, right-click on "CANopen Manager" and click "Add objects".
$\Rightarrow$ A dialog is shown where you can choose a CANopen remote device.
2. Double-click at a CANopen remote device to show related parameters/information in a new window.

The tab "General" contains the following parameters:

| Parameter <br> General |  | Value | Description |
| :--- | :--- | :--- | :--- |
| Node-ID |  | Next unused value | $1-127$ |


| Parameter | Default value | Value | Description |
| :--- | :--- | :--- | :--- |
| Life time factor (only if <br> nodeguarding is ena- <br> bled) | Disabled | Disabled | The remote device doesn't <br> send heartbeat messages. |
|  |  | Enabled | The remote device sends <br> heartbeat messages. |
| Producer time (ms) <br> (only if heartbeat pro- <br> ducing is enabled) | 10 | $1-65535$ | Time interval in which <br> heartbeat messages are <br> sent. |
| Heartbeat consuming |  |  | Opens a "Heartbeat <br> Consuming Properties" <br> dialog. There you activate <br> the CANopen devices that <br> you want to watch. <br> The number of possible <br> devices to be monitored is <br> defined in the EDS file. To <br> do this, you must select <br> the "Enable"check box and <br> enter the Node-ID of the <br> device and the required <br> values in the "Heartbeat <br> time" field (in milliseconds). <br> When a remote device |
| is monitoring, a green |  |  |  |
| check mark is displayed on |  |  |  |
| the "Heartbeat Consuming" |  |  |  |
| button. |  |  |  |$|$

Within the tab "PDO Mapping" PDOs for the CANopen remote device can be configured.
The PDOs are splitted into RPDOs and TPDOs. For each PDO corresponding I/Os are shown.
The following options are available for TPDOs and RPDOs:

| Parameter | Default value | Value | Description |
| :---: | :---: | :---: | :---: |
| COB-ID | calculated | $\begin{aligned} & 0 \times 180-0 \times 57 F \\ & 0 \times 681-0 \times 6 \mathrm{DF} \end{aligned}$ | The COB-ID is filled in automatically dependend to the node id and number of PDO but can be changed manually. |
| Type | Asynchronous device profile specific | Asynchronous device profile specific | The PDO is transmitted according to the CiA device profile. |
|  |  | Acyclic - synchronous | When a change is made, the PDO is transmitted synchronously, but not periodically. |
|  |  | Asynchronous manufacturer specific | The PDO is transmitted only after specific events. |
|  |  | Cyclic - synchrounus | PDO is transmitted using SYNC mechanism. |
|  |  | Synchronous only RTR <br> (TPDOs only) | After a synchronization message, the PDO is updated, but not transmitted. Transmission is by explicit request only (Remote Transmission Request). |
|  |  | Asynchronous only RTR <br> (TPDOs only) | The PDO is updated and transmitted by explicit request only (Remote Transmission Request). |
| Num of syncs (only for Type "Cyclic synchronous") | 1 | 1-240 | Number of SYNC messages before PDO is sent. |

The following options are only available for TPDOs.

| Parameter | Default value | Value | Description |
| :--- | :--- | :--- | :--- |
| Inhibit time (x 100 $\mu \mathrm{s}$ ) | 0 | $0-65535$ | Devicedependant. The <br> inhibit time is the min- <br> imum time between two <br> messages of a specific <br> PDO. You can use this set- <br> ting for preventing PDOs <br> from being sent too often <br> when their values are <br> edited. Default: "0". Pos- <br> sible values: 0-65535. |
| Event time (x 1ms) <br> (only for types <br> "asynchronous - <br> manufacturer specific" <br> and "asynchrounus - <br> device profile") | 0 | Device-dependant. Indi- <br> cate the time span that <br> should be between two |  |
| PDO transmissions PDOs |  |  |  |
| (in milliseconds). |  |  |  |


| Parameter | Default value | Value | Description |
| :--- | :--- | :--- | :--- |
| RTR | Disabled | Disabled | RTR( Remote transmis- <br> sion request) not used. |
|  |  | Enabled | A RTR frame can be used <br> for interrogating the PDO <br> externally. |

The "Bootconfiguration" tab allows the configuration of the boot up behavior of the CANopen remote device.

The following options are available and can be set by either clicking on the graphics or selecting the corresponding check box on the right side of the editor window. The graphic illustrates the order of execution:

| Parameter | Default value | Value | Description |
| :---: | :---: | :---: | :---: |
| Node ID-active | Enabled | Disabled | The remote device is inactive. |
|  |  | Enabled | The remote device is active. |
| Node BootUp |  |  |  |
| Node reset (send the reset-node command) | Enabled | Disabled | No specific node reset communication command is sent. |
|  |  | Enabled | The manager sends the CANopen specific node reset communication command. |
| Check node type and profile (compare the configured profile and type object 1000 H with real value) | Disabled | Disabled | The content of the node object 1000 H are not compared with the current parameters. |
|  |  | Enabled | The "CANopen Manager" compares the content of the mandatory node object 1000 H . If the values are different, the manager will report a parameterization error. |
| Configuration Guard Protocol (configure the guard time and lifetime factor) | Enabled | Disabled | The guard time and lifetime factor of the remote device configuration are not written. |
|  |  | Enabled | A CANopen has 2 specific register responsible for the Node guarding protocol. The "CANopen Manager" writes the guard time and Life-Time factor of the node configuration into the corresponding objects of the remote device during startup. |


| Parameter | Default value | Value | Description |
| :--- | :--- | :--- | :--- |
| Configuration SYNC <br> COB-ID (configure the <br> COB-ID for the syn- <br> chronization-telegram) | Enabled | Disabled <br> The "CANopen Manager" <br> does write the SYNC <br> COB-ID of the config- <br> uration into the corre- <br> sponding objects of the <br> remote deivce. |  |


| Parameter | Default value | Value | Description |
| :--- | :--- | :--- | :--- |
| Initiate PDO data <br> (remote request all <br> TxPDOs and send <br> current RxPDOs once <br> after) | Enabled | Disabled | PDOs are not written <br> and read by the manager <br> automatically. |
|  | Enabled | This item selects if <br> the installed PDOs <br> shall be automatically <br> written and read by <br> the "CANopen Manager" <br> directly after the startup <br> once. This ensures that <br> the latest output data <br> which can be found within <br> the "CANopen Manager" <br> output process data area <br> is sent to the remote <br> device and that the latest <br> remote device input data <br> is read from the remote <br> device and be placed into <br> the input process data <br> area. |  |

The tab "CI58x IEC Objects" contains the created instance used for diagnosis access. Only available for CI 58 x devices.

## Configuration of SYNC task

To create a SYNC task, which is triggered after each SYNC message, several steps are needed:

1. Set CANopen sync mode to "Sync Bus and Task" at CM598 communication module.
2. Enable "SYNC producing" at CANopen manager.
3. Set at least one PDO to transmission type "cyclic synchronous".
4. Build the project.
$\Rightarrow$ A new task is created automatically which is called "ABB SyncTask <slot>" there you can add POUs which should be executed if the task is triggered.



Fig. 272: Example of SYNC task configuration

## Configuration of the protocols CAN 2.0 A / CAN 2.0 B

The communication module CM598-CAN can be used to realize CAN bus based networks in combination with library ABB_CM598Can_AC500.library.

To enable the support for the desired protocol it must be appended to CM598-CAN.

1. Right-click "CM598_CAN (CM598-CAN)" in the device tree and select "Add device" in the context menu.
$\Rightarrow$ Window "Add object below: CM598_CAN_1" appears.
2. Select "CAN 2.0 A" or "CAN 2.0 B" from the list.

Parameteriza- The CAN data transmission requires a buffer for the incoming data that can be read with tion function blocks of library ABB_CM598Can_AC500.library.

1. Right-click "CAN_2_OA_11_bit_identifier_ (CAN 2.0A)" or "CAN_2_OB_29_bit_identifier_ (CAN 2.0B)" and select "Add object".
2. Select "Buffer for CAN 2A" for CAN 2.0A. Or select "Buffer for CAN 2B" for CAN 2.0B from the list.
3. Double-click on "Buffer_for_CAN_2A (Buffer for CAN2A)" or "Buffer_for_CAN_2B (Buffer for CAN2B)" in the device tree to open the "Buffer configuration" in the editor window.

The following parameters are available:

| Parameter | Default value | Value | Description |
| :--- | :--- | :--- | :--- |
| Identifier | 0 | CAN 2A: 0 .. 2047 <br> CAN 2B: 0 ... <br> 536870911 | The value of the CAN <br> identifier that is com- <br> pared with the identi- <br> fier of the incoming <br> telegrams. The tele- <br> grams will be added <br> to the buffer if the <br> identifier matches. |
| Number of receive <br> buffers | 1 | $1 \ldots 16$ | The size of the buffer <br> in number of tele- <br> grams. |


| Parameter | Default value | Value | Description |
| :--- | :--- | :--- | :--- |
| Behavior on receive <br> buffer overflow | Overwrite | Overwrite | The oldest telegram <br> in the buffer is <br> overwritten by the <br> incoming telegram. |
|  |  | Discard | Incoming telegrams <br> are discarded as long <br> as the buffer is full. |
| Enable triggering of <br> IEC task | No | Disables the triggering <br> of the execution of the <br> related IEC task. |  |
|  |  | No | Enables the triggering <br> of the execution of <br> the related IEC task <br> as soon as a CAN <br> frame with the speci- <br> fied CAN-ID arrives <br> E Chapter 6.8.3.1.1 <br> "Triggering of event <br> tasks with CAN-IDs" <br> on page 4565. |

## Configuration of the CANopen manager

Chapter 6.3.2.16.1.2 "CANopen manager (master)" on page 1594

### 6.3.2.11.2 PROFINET

CM579-PNIO - PROFINET IO communication module

## Configuration of the communication module



Configuration is valid as of Automation Builder 2.2.0 and CPU firmware 3.2.0.
Until Automation Builder 2.5.x and CPU firmware 3.5.x the minimum firmware version 2.8.4.0 is required for communication module CM579-PNIO.

As of Automation Builder 2.6.0 and CPU firmware 3.6.0 the minimum firmware version 3.0.0.21 is required for communication module CM579-PNIO.
${ }^{\Downarrow}$ Description of configuration with Automation Builder $<2.2 .0$

As of Automation Builder 2.6 .0 some component changes of the PROFINET IO controller stack require a manual upgrade of projects created in Automation Builder versions < 2.6.0. How to perform those steps is described in the application note AC500 V3-Upgrade of projects with PROFINET.

Append a
CM579-PNIO

1. Right-click on your desired Slot below node "Extension_Bus" and click "Add object". $\Rightarrow$ Dialog "Replace object:" appears.
2. Click "CM579_PNIO" in the list and click [Replace object].
3. Double-click "CM579_PNIO (CM579-PNIO)" to get the "CM579-PNIO Parameters" in the editor window.
The following parameters are available:

| Parameter | Default | Value | Description |
| :--- | :--- | :--- | :--- |
| Run on config <br> fault | No | No | In case of a configuration error, the <br> user program is not started. |
|  | Yes | The user program is started inde- <br> pendent of a faulty configuration <br> of the CM579-PNIO communication <br> module. |  |

4. Click on tab "CM579-PNIO I/O Mapping" to open the "Bus Cycle Options" in the editor window.

The following parameters are available:

| Parameter | Default | Value | Description |
| :--- | :--- | :--- | :--- |
| Bus cycle task | Use parent bus <br> cycle settings | Use parent bus <br> cycle settings | Settings from "PLC Settings" tab <br> are used. |
|  | Task | Name of the task that triggers the <br> bus cycle. |  |

In the tab "CM579-PNIO IEC-Objects" the list of used IEC Objects for information instantiated I/O driver function block class is shown.

The library "CM579PROFINET" will be added automatically to "Library
Manager" when a CM579-PNIO is appended.

## Configuration of the PROFINET IO controller

## PROFINET IO controller - Configuration

The PROFINET IO controller node appears automatically below the added communication module CM579-PNIO.

- Double-click on "PNIO_Controller" and open the tab "General" in the editor window.

The following parameters are available:

| Parameter | Default | Value | Description | Parameter |
| :--- | :--- | :--- | :--- | :--- |
| Station name | cm579-pnio | Up to 240 char- <br> acters | Network name of the <br> PROFINET IO controller <br> station. Must be conform to <br> PROFINET standard. | Station name |
| IP parameters | IP-Address | 192.168 .0 .1 | Valid IP address | IP address of the <br> PROFINET IO controller <br> station. |
| Subnet mask | 255.255.255. <br> 0 | Valid subnet <br> mask | Network mask of the <br> PROFINET IO controller <br> station. | Subnet mask |


| Parameter | Default | Value | Description | Parameter |
| :---: | :---: | :---: | :---: | :---: |
| Default gateway | 0.0.0.0 | Valid gateway address | Gateway address of the PROFINET IO controller station. | Default gateway |
| Address settings for device |  |  |  |  |
| First IPAddress | 192.168.0.2 | Valid IP address | First IP address of the PROFINET IO devices. This parameter determines the address range of the PROFINET IO devices in combination with parameter Last IP address. | First IP address |
| Last IP- <br> Address | $\begin{aligned} & 192.168 .0 .25 \\ & 4 \end{aligned}$ | Valid IP address | Last IP address of the PROFINET IO devices. This parameter determines the address range of the PROFINET IO devices in combination with parameter First IP address. | Last IP address |
| Subnet mask | $\begin{aligned} & 255.255 .255 . \\ & 0 \end{aligned}$ | Valid subnet mask | Network mask of the PROFINET IO devices. | Default subnet mask |
| Default gateway | 0.0.0.0 | Valid gateway | Gateway address of the PROFINET IO devices. | Default gateway address |
| Bus State |  |  |  |  |
| Bus State in Stop | Stop |  | Defines the behavior of the fieldbus when the PLC is in Stop. In "PLC Settings"IO update and behavior of outputs in stop can be configured. <br> * ${ }^{4}$ Chapter 6.4.1.21.2.8.10 "Tab 'PLC Settings"' on page 2439 | Bus State in Stop |
|  |  | Stop | The protocol of the fieldbus is stopped. There is no communication between PROFINET controller and devices. |  |
|  |  | Operation | The protocol of the fieldbus is executed. The communication between PROFINET controller and devices is enabled. |  |

## PROFINET IO controller - Parameters

The tab "PROFINET-IO-Controller Parameters" is a generic view of all PROFINET IO controller parameters. It is normally hidden and is normally not needed for configuration.

Use tab "PROFINET-IO-Controller Parameters" only, if you need to change a parameter, which is not visible in other dialogs.

1. Click "Tools $\rightarrow$ Options" and select "Device editor".
$\Rightarrow$ The "Device editor" dialog opens.
2. Enable checkbox "Show generic device configuration views" and click [OK]
$\Rightarrow$ The tab is now available.

## Configuration of PROFINET IO devices

## Add PROFINET IO device

1. Right-click on node "PNIO_Controller (PROFINET-IO-controller)" and click "Add object." $\Rightarrow$ A list with all installed PROFINET IO devices appears.
2. Slelect the desired device and click [Add object].
$\Rightarrow$ The device is added to the PROFINET IO controller in the device tree.

## PROFINET IO device - Configuration

Double-click on "PNIO-Device" to open the device configuration in the editor window. The following parameters are available:

With "Device live list" view the PROFINET IO bus can be scanned for available devices and the device names and IP addresses can be assigned.

| Parameter | Default | Value | Description | Parameters |
| :---: | :---: | :---: | :---: | :---: |
| Identification |  |  |  |  |
| Station name <br> ② Chapter 6.9.2.6.1.4 " PROFINET scan and comparison view" on page 4651 | Device-specific | Up to 240 characters | This is a system wide unique name for addressing the device. Must be conform to PROFINET standard. | Slave parameters -> Identification -> Station name |
| Communication |  |  |  |  |
| Send clock (ms) | Device-specific | $\begin{aligned} & 1 \\ & 2 \\ & 4 \end{aligned}$ | The Send cycle is divided into several phases. The Send clock specifies the length of a phase. It is a multiple of time base $31.25 \mu \mathrm{~s}$. <br> Send clock and Reduction ratio determine the Send cycle: <br> Send cycle $=$ Send clock $x$ Reducation ratio | Slave parameters -> Send clock |


| Parameter | Default | Value | Description | Parameters |
| :---: | :---: | :---: | :---: | :---: |
| Reduction ratio | Device-specific | 1... 16384 | The Reduction ratio determines how often the IO data is sent within the Send cycle. <br> Send clock and Reduction ratio determine the Send cycle: <br> Send cycle= Send clock x Reducation ratio | Slave parameters -> Reduction ratio |
| Phase | 1 | 1 ... Reduction ratio | Defines the part of the Send cycle at which the IO data is sent. | Phase |
| Watchdog | 3 | 1 ... 65535 | Time that is used to monitor the communication between IO controller and IO device. | Watchdog interval |
| RT class |  |  |  |  |
| RT class | RT Class 1 | RT Class 1 (legacy) | Defines the Realtime Class of cyclic data. Currently only RT Class 1 (legacy) and RT Class 1 are supported. | Slave parameters -> RT Class |
|  |  | RT Class 1 |  |  |
| IP Parameter |  |  |  |  |
| IP-Address | 192.168.0.2 | Valid IP address | IP address of the PROFINET IO controller station. | Slave parameters -> Identification -> IP address |
| Subnetmask | $\begin{aligned} & 255.255 .255 . \\ & 0 \end{aligned}$ | Valid subnet mask | Network mask of the PROFINET IO controller station. | Slave parameters -> Identification -> Subnet mask |
| Default gateway | 0.0.0.0 | Valid gateway address | Default gateway address of the PROFINET IO controller station. | Slave parameters -> Identification -> Default gateway address |

The following parameter is only supported up to Automation Builder version 2.5.

| Parameter | Default | Value | Description | Parameter |
| :--- | :--- | :--- | :--- | :--- |
| VLAN ID | 0 | $0 \ldots 4095$ or <br> $0 \ldots 32767$ | In case of VLAN <br> usage the param- <br> eter VLAN ID <br> represents the ID <br> of the virtual net- <br> work. | Slave parameters <br> -> VLAN ID |
|  |  | For VLAN type <br> 802.1Q the range <br> is 0..4095 while <br> VLAN type ISL <br> accepts values <br> from 0 to 32767. <br> The supported <br> type depends on <br> the used device. |  |  |

## PROFINET IO device - PNIO parameters

The tab "PNIO Parameters" is a generic view of all PROFINET IO device parameters. It is normally hidden and is normally not needed for configuration.


Use tab "PNIO Parameters" only, if you need to change a parameter, which is not visible in other dialogs.

Activating tab

1. Click "Tools $\rightarrow$ Options" and select "Device editor".
$\Rightarrow$ The "Device editor" dialog opens.
2. Enable checkbox "Show generic device configuration views" and click [OK]
$\Rightarrow$ The tab is now available.

## Configuration of 3rd party PROFINET IO devices

Before a 3rd party PROFINET IO device can be used, the provided GSDML file has to be installed in the "Device Repository".

Installation Go to "Tools $\rightarrow$ Device Repository $\rightarrow$ Install".

Configuration $\Leftrightarrow$ Chapter 6.3.2.11.2.1.3 "Configuration of PROFINET IO devices" on page 1537.

## I/O mapping of the PROFINET IO devices

Open I/O map- 1. Double-click on the "PNIO_Controller" or below on the "<...>PNIO-Device" or below on the ping list "I/O-Module" in the device tree.
2. Select tab "I/O mapping list" to show the list of I/O channels.

The content of the list depends on the selected node.
For instance:

- When the "PNIO_Controller" node is selected all I/O channels of all configured devices are shown.
- When a "PNIO-Device" is selected all I/O channels of the configured modules are shown.

An IEC variable for an I/O channel that is available in the "Application" can be defined by double-clicking in column "Variable".

| PNIO_Controller |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| General <br> PROFINET-IO-Controller Parameters | $\geqslant$ ¢ $\times$ Clear mappings $\mid \boldsymbol{T}$ |  |  |  |  |  |
|  | , Object Name | Variable | Channel | Address | Type |  |
|  | CI501_IO (CI501_PNIO_DeviceU) |  | Fast Counter: Actual value 1 | \%ID3 | UDINT | * |
| I/O mapping list | CI501_10 (CI501_PNIO_Device |  |  |  |  |  |
|  |  | Fast Counter: Actual value 2 | \%ID4 | UDINT |  |  |
|  | CI501_IO (CI501_PNIO_Device |  |  |  |  |  |
| ) |  | Fast Counter: State byte 1 | \%IB20 | USINT |  |  |
| PROFINET-IO-Controller I/O Mapping | CI501_IO (CI501_PNIO_Device |  |  |  |  |  |
|  |  | Fast Counter: State byte 2 | \%1821 | USINT |  |  |
|  | CI501_1O (CI501_PNIO_Devicel) |  | Analog Output Channel 0 | 96QW0 | INT | 1. |
| Status | CI501_IO (CI501_PNIO_Devicel) | Input_C1501 | Analog Output Channel 1 | $96 \mathrm{QW1}$ | INT | 1. |
| LicenseRequiredNotification | CI501_IO (CT501_PNIO_Devicel) |  | 8 -bit digital output | \%QQB4 | USINT |  |
|  | CI501_10 (CI501_PNIO_Devicel) |  | 8 -bit digital output - Channel 0 | $96 Q \times 4.0$ | BOOL | 3. |
| Information | CI501_IO (CI501_PNIO_Devicel) |  | 8 -bit digital output - Channel 1 | \%QX4.1 | BOOL | 3. |
|  | CI501_IO (CI501_PNIO_Devicel) |  | 8 -bit digital output - Channel 2 | $96 Q \times 4.2$ | BOOL | 3. |
|  | CI501_IO (CI501_PNIO_Device |  |  |  |  |  |
|  |  | 8 -bit digital output - Channel 3 | \% 6 Q 4.3 | BOOL | 3. |  |
|  | CI501_IO (CI501_PNIO_Devicel) |  | 8 -bit digital output - Channel 4 | $96 Q \times 4.4$ | BOOL | 3. |
|  | CI501_IO (CI501_PNIO_Devicel) |  | 8 -bit digital output - Channel 5 | \% 6 Q44.5 | BOOL | 3. |

## Provider and consumer status

Provider and consumer status functionality is available as of Automation Builder2.7.0 and CPU firmware 3.7.0.

The provider and consumer status of a PROFINET submodule is in the I/O editor available.
For each submodule 4 additional I/O channels exists that can be used to read or write the provider and consumer status. These channels are updated with the cycle of the specified buscycle task.

| Channel | Direction | Description |
| :--- | :--- | :--- |
| "Input PS" (submodule name) | input | Provider status of the input <br> channels |
| "Output CS" (submodule <br> name) | input | Consumer status of output <br> channels |
| "Output PS"(submodule <br> name) | output | Provider status of the output <br> channels |
| "Input CS" (submodule name) | output | Consumer status of the input <br> channels |

The channels "Output PS" and "Input CS" are handled by the IO driver of the communication module CM579-PNIO when no mapping variable is assigned. The status value is set to "Bad_By_Controller" when the PLC is in "Stop" and to "Good" when the PLC is in "Run".
In case of a mapping variable is assigned the correct status value has to be set in the PLC application.

The channels "Input PS" and "Output CS" contains the status values as received form the PROFINET device. If the Application Relation (AR) between PROFINET device and controller is closed the status is set to "Bad_By_Controller".

| Disgnomas <br> General | Find <br> Varisble $*$ | Fiter Showall |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | Mopping | Channel Andog Inout Oiarnel o | Address 5NWS. | 3pe NT | Defout Value | Unt | Description |
| Waso Modile Parameters | * |  | Andog trat Oomel 1 | stwss | 2 ar |  |  |  |
|  | $*$ |  | Nosog Trat Orame 2 | *ww\% | art |  |  |  |
| Pw20 Module Jo Mapping | * |  | Ansog Ihat Cramel 3 | Swws | NT |  |  |  |
|  | - ${ }^{*}$ |  | 3bsdotar nout | $\times 5176$ | Usist |  |  |  |
| Mos Module ItC Objects <br> 1/0 mapping list | * |  | Fant Cartar: Mcusi value : | *N030 | Leort |  |  |  |
|  | * |  | Fast Carter kcuas vilu 2 | V1031 | $100 \pi$ |  |  |  |
|  | * |  | Fest Cornser: Stase byte 1 | 588128 | USET |  |  |  |
| 2formaton | $*$ |  | Fsat Courser: Stase brite 2 | *2812\% | Usidt |  |  |  |
|  | $*$ |  | toutps | N*18130 | Oumerston of ert |  |  |  |
|  | $*$ |  | apsutcs | K1813: | Erueaton ol erte |  |  |  |
|  | * |  | Maks Cutut Ormelo | Nows6 | sor |  |  |  |
|  |  |  | msibg Oitut Chwel 1 | Kows | 3 mr |  |  |  |
|  | *** |  | 8itidotal autut | Sceits | USEPT |  |  |  |
|  | " |  | Fest Coanter: Start vile 1 | *coso | Leent |  |  |  |
|  | " |  | Fast Canses: End vale 1 | $\times \mathrm{cosin}$ | Leort |  |  |  |
|  | * |  | Fast Carmel Start vive? | scous | U000T |  |  |  |
|  | " |  | Fest Canser: End vaice 2 | *6033 | UCONT |  |  |  |
|  | " |  | Fast Canto: Contol byte 1 | *5c138 | usist |  |  |  |
|  | " |  | Fsot Courter Contod byte 2 | *-9137 | Usivt |  |  |  |
|  | ") Provderstonmousut | , | aputips | *98138 | Sumerson of erte | Bad.by_Contoler |  |  |
|  | * Corometsatatrout | - | toutcs | * 5 ceis\% | Burerson sforte | Sos,jel_Contoter |  |  |

## Remanent data

The remanent data of the communication module CM579-PNIO is stored in the PLC filesystem. It will be read and evaluated by the communication module during configuration.
When a new project is loaded or the communication module is replaced the data is not deleted. It can be deleted either manually in the file browser or in the Automation Builder menue "Online $\rightarrow$ Reset Origin" command.


## Shared device

> Shared device functionality is available as of Automation Builder 2.6.0 and CPU firmware 3.6.0.

The shared device functionality allows it to use the same IO device with multiple IO controllers at the same time. The sharing of an IO device is done on submodule level. A submodule of PROFINET IO device must be uniquely assigned to a PROFINET IO controller.
The example below shows the configuration of 2 IO controllers using a CM589-PNIO-4 as shared device. The modules or submodules that are assigned to an IO controller are shown in black style. The unassigned ones are in gray style.


The following workflow applies to an IO device configuration:

1. Add an IO device to an IO controller.
2. Add IO modules and submodules according to IO device hardware.
3. Setup required parameters of IO device.
4. Repeat the IO device configuration in the same way for further IO controllers.
5. Select the assigned IO submodules for each IO controller with disconnect/connect module function in context menu.
6. Define one IO controller to be responsible to connect to DAP.

To disconnect DAP and port submodules from IO controller use option "Shared device":

1. Double-click on "PNIO-Device" to open the device configuration in the editor window.
2. Select tab "Options" to show option "Shared device".


| Parameter | Default | Value | Description |
| :--- | :--- | :--- | :--- |
| Shared device | unchecked | unchecked | DAP and Port sub- <br> modules are con- <br> nected. |
|  |  | checked | DAP and Port sub- <br> modules are discon- <br> nected. |

Rules to be followed on using PROFINET "Shared device" configurations:

- All configurations of the IO device at IO controllers using this IO device have to look the same.
- IP configuration (IP address, subnet mask) have to be the same.
- The station names has to be the same.
- Each IO submodule has to be connected at one single IO controller only.

Automation Builder is not able to check these details for consistency.

## CM589-PNIO / CM589-PNIO-4-PROFINET IO device communication module

Configuration is valid as of CPU firmware 3.5 .0 ("CM589-PNIO") respectively CPU firmware 3.6.0 ("CM589-PNIO-4") and "CM589-PNIO" FW 1.6.2.20.

The configuration of the "CM589-PNIO" PROFINET I/O module has to be done in the following steps:

- Parameterization of the AC500 communication module interface $\Leftrightarrow$ "Parameterization - communication module interface" on page 1543
- Parameterization of the PROFINET IO device protocol stack "Parameterization PROFINET I/O stack" on page 1544
- Configuring PROFINET IO device module structure « "Configuring PROFINET IO structure" on page 1544
- Parameterization of the PROFINET IO device modules " "Parameterization-PROFINET IO device modules" on page 1545
- Mapping of the I/Os " "Mapping of the I/Os" on page 1546
- Configuration specific to CM589-PNIO-4 « "Configuration CM589-PNIO-4" on page 1547


Configuration procedure for "CM589-PNIO" and "CM589-PNIO-4" are basically the same. Thus this description refers to "CM589-PNIO" only. Differences between "CM589-PNIO" and "CM589-PNIO-4" will be highlighted explicitly if necessary!

Parameteriza- For connecting a PLC as "PROFINET-IO-Device", plug "CM589-PNIO"/ "CM589-PNIO-4" at the tion - communication module interface "Extension_Bus" node.
Double-click on "CM589-PNIO" to open the "CM589-PNIO" configuration in the editor window. The following parameter is available:

| Parameter | Value | Default Value | Description |
| :--- | :--- | :--- | :--- |
| Run on config <br> fault | No | No | In case of a configuration error，the <br> user program is not started． |
|  | Yes | The user program is started even in <br> case of configuration error． |  |

Click on tab＂CM589－PNIO I／O Mapping＂to get＂Bus Cycle Options＂in the editor window． The following parameter is available：

| Parameter | Value | Default Value | Description |
| :--- | :--- | :--- | :--- |
| Bus cycle task | Use parent bus <br> cycle settings | Use parent bus <br> cycle settings | Settings from＂PLC settings＂tab are <br> used． |
|  | Task | Name of task that triggers the bus <br> cycle． |  |

Click on tab＂CM589－PNIO IEC Objects＂．Here the IO driver instance of communication module is specified．

Parameteriza－All parameters for＂PROFINET－IO－Device＂protocol configuration are set automatically by tion－PROFINET Automation Builder．These parameters are displayed just for information and in read－only I／O stack mode．Double－click on＂PROFINET－IO－Device＂in tree view will show the parameter set in tab ＂PROFINET－IO－Device Parameters＂．

| Devices $\quad$－ $\mathbf{4}$ | 圆 StartPage PlC＿A | ＿V3 K CM589＿PNIO | 4a－4＿Byte＿Input | （1）PNIO＿Device | x |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| －－PLCWAND | I／O mapping list | Parameter | Type | Value | Default Value | Unit | Description |
| －F Fill PLC Logic | PROFINET－IO－Device Parameters | －Config version | DWORD | 16\＃01000000 | 16\＃01000000 |  |  |
| $\pm$ Application |  | －Vendor ID | UDINT | 26 | 26 |  |  |
| ＋－io＿bus | Diagnosis | －Device－Id | UDINT | 28 | 28 |  |  |
| ${ }^{+}$．．Interfaces |  | －Maximum input data | UDINT | 1440 | 1440 |  |  |
| －䗦 Extension＿Bus | Status | －Maximum output data | UDINT | 1440 | 1440 |  |  |
| － $\mathrm{m}^{6}$ CM589 PNIO（CM588PNIO） |  | －Station name | STRING | ＇cm589－pn－00＇ | ＇cm589－pn－00＇ |  |  |
| 60 PNIO＿Device（PROFINET－IO－D | Information | －SetiP Para | Enumeration of BYTE | No | No |  |  |
|  |  | －IP address | ARRAY［0．3］OF BYTE | ［ $0,0,0,0]$ | ［0，0，0，0］ |  |  |
| －緒 PNIO＿Device＿1 ${ }^{\text {（PROFINET－IO }}$ |  | －Subnet Mask | ARRAY［0．．3］OF BYTE | ［ $0,0,0,0]$ | ［ $0,0,0,0]$ |  |  |
|  |  | －Default gateway | ARRAY $[0.3]$ OF BYTE | $[0,0,0,0]$ | ［ $0,0,0,0]$ |  |  |
|  |  | －Max diag records | UINT | 32 | 32 |  |  |
|  |  | －Max AR | UDINT | 0 | 0 |  |  |
|  |  | －Instance－Id | UINT | 1 | 1 |  |  |
|  |  | －Diag on link down | Enumeration of BYTE | Yes | Yes |  |  |
|  |  | －Activate IOxS | Enumeration of BYTE | Disabled | Disabled |  |  |
|  |  | －List of APIs | ARRAY［0．．4］OF UDINT | ［ $1,0,0,0,0$ ］ | $[1,0,0,0,0]$ |  |  |


－＂Station name＂：The default name is displayed．The real name used on acting at the field bus is combined out of this default name and the used setting of the rotary switches at the CM589 module（＂cm589－pnio－00＂，＂00＂ will be replaced by rotary switch value）or the name set via PROFINET set name service．
－Parameter＂IP address＂，＂Subnet Mask＂，＂Default gateway＂：The default values are displayed here．These values are not used as communication settings．PROFINET IO controller supplies the IO devices with IP settings on communication establishing．

Configuring PROFINET IO structure
＂CM589－PNIO＂provides I／O data as modules with different data types and directions．Create an application specific I／O structure by compiling an appropriate combination of modules．

To assign I／O modules to＂PROFINET－IO－Device＂node open＂Add object＂dialog．


The maximum image size available for I/O data depends on "CM589-PNIO" modules firmware revision. If firmware revision run by connected "CM589-PNIO" module does not support the number of I/O data configured, download configuration will fail. Currently see "Calculating size of I/O data" on page 1546 how to calculate number of I/O data occupied by certain configuration.
PROFIsafe modules can only be used if a SM560-S-FD1 (for "CM589-PNIO") respectively SM560-S-FD4 (for "CM589-PNIO-4") was configured in project. For details see safety documentation.

Parameterization - PROFINET IO device modules

PROFINET IO device modules do not need user configuration. All needed parameters are set automatically by Automation Builder. Double-click on a module node shows the parameter set just for information. This parameter set is identical for all module types and is displayed in read-only mode.


- "API": Shows the API which is used by the "CM589-PNIO" modules. As API 0 is supported only "CM589-PNIO" modules do not provide configuration capabilities for this parameter.
- "Slot number", "SubSlot-Number", "Offset in DPM", inputs/outputs: Will be set to default values on inserting a module. On creating configuration data Automation Builder calculates real values and overwrites the defaults.
- "Offset IOPS" provider/consumer: Not used and set to 0 values.
- "PROFIsafe module": "FALSE" if module is not a safety module.


## Calculating size of I/O data

PROFINET defines I/O data and status information to be exchanged between IO controller and IO device. The status information is called "Provider Status" and "Consumer Status". Both (I/O data and status information) have to be considered on calculating allocated memory in input and output image.

- The number of status bytes depends on the type of module used.
- The different types of modules input, output and in/output have to be considered different.
- Some status bytes are reserved for predefined submodules have to be considered additionally.

A configured I/O module allocates memory space at the corresponding I/O image for data and status bytes. Additionally memory is allocated at the opposite directions I/O image to store further status bytes. E.g. an input module allocates memory at the input image but additionally it allocates one byte for status at the output image. Summarized size of input and output data and status has to fit to the corresponding image.

Table 301: Overview of I/O module types and corresponding status bytes

| Module <br> type | Input data |  |  | Output data |  |  |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
|  | Inputs | Provider <br> status <br> inputs | Consumer <br> status out- <br> puts | Outputs | Provider <br> status out- <br> puts | Consumer <br> status <br> inputs |
| Reserved | 0 input <br> bytes | 4 bytes | 0 bytes | 0 bytes | 0 bytes | 4 bytes |
| Input <br> module <br> (e.g. 4 byte <br> input) | n input <br> bytes | 1 byte | 0 bytes | 0 bytes | 0 bytes | 1 byte |
| Output <br> module | 0 input <br> bytes | 0 bytes | 1 byte | n output <br> bytes | 1 byte | 0 bytes |
| Input/output <br> module | $n$ input <br> bytes | 1 byte | 1 byte | $n$ output <br> bytes | 1 byte | 1 byte |

Following expressions calculate allocated sizes of input and output data:

| Size input $=$ | Input + status +4 bytes (reserved status) |
| :--- | :--- |
| Size output $=$ | Output + status +4 bytes (reserved status) |

- $\quad$ Input $=$ Summarized number input bytes all modules.
- Output = Summarized number output bytes all modules.
- Status $=$ Count input modules + count output modules +2 * count input/output modules.

[^29]Symbolic names for variables, inputs and outputs for further details on mapping inputs and outputs.
② Chapter 6.3.2.13.7 "Symbolic names for variables, inputs and outputs" on page 1572

Configuration "CM589-PNIO-4" adds PROFINET "Shared Device" feature to already described "CM589-PNIO"
device functionality. Thus "CM589-PNIO-4" is able to communicate to 4 different PROFINET IO controllers in parallel.

Shared device usage of a "CM589-PNIO-4" does not need to be considered on doing the configuration of the "CM589-PNIO-4" coupler device itself. It has to be considered only on doing the configuration of "CM589-PNIO-4" as child to the "CM579-PNIO" IO controller. See "CM579-PNIO" configuration for details.

## Diagnosis and

 debugging for AC500 V3 products
## "CM589-PNIO" - PROFINET device diagnosis

The diagnosis messages of communication module "CM589-PNIO" are displayed in tab "Diagnosis" of node "CM589-PNIO" in device tree of Automation Builder. Within PLC application they can be read with the diagnosis methods of IO driver or function block "Diag".

* ${ }^{4}$ Chapter 6.9.1.5.4.2 "Device state" on page 4629

Chapter 6.9.1.5.3.3.2 "Method Ack / DiagAck: acknowledgement" on page 4623
In PLC display the diagnosis messages of "CM589-PNIO" are not shown.
²) Chapter 6.9.3.6.3 "CM589-PNIO(-4) errors" on page 4688

## CM589-PNIO - PROFINET IO slave

Configuration in Automation Builder is described in PROFINET IO slave configuration.
« Chapter 6.3.2.18 "PROFINET IO Configurator" on page 1640

### 6.3.2.11.3 EtherCAT

## CM579-ETHCAT - EtherCAT I/O master

« ${ }^{\wedge}$ Configuration in Automation Builder.

- Double-click on "CM579_ECAT (CM579-ECAT)" to open the CM579-ECAT configuration in the editor window.
The following parameters are available:

| Parameter | Default value | Value | Description |
| :--- | :--- | :--- | :--- |
| Run on config <br> fault | No | No | In case of a configuration error, <br> the user program is not started. |
|  |  | Yes | The user program is started inde- <br> pendent of a faulty configuration <br> of the EtherCAT communication <br> module. |
| Broken slave <br> behavior | Leave all broken <br> slaves down | Leave all broken <br> slaves down | Broken slaves will not be served. |
|  |  | Leave addressless <br> slaves down | Only slaves without address will <br> be left down. |
|  |  | Leave no slaves <br> down | Broken slaves will be ignored. |


| Parameter | Default value | Value | Description |
| :--- | :--- | :--- | :--- |
| Distributed clocks | Inactive | Inactive | Distributed clocks are inactive. |
|  |  | Active | Distributed clocks are active. |
| Bus Target State | Operational, OP | Operational, OP | Target state of the EtherCAT bus <br> at application start. |
|  |  | Safe-Operational, <br> SAFEOP |  |
|  |  | Pre-Operational, <br> PREo |  |
| Bus behavior | Asynchronous (IEC <br> bus cycle) | Asynchronous <br> (IEC bus cycle) | Type of bus behavior (asynchro- <br> nous/synchronous) |
|  |  | Synchronous <br> (Sync mode 1) | Minimum lag (1 bus cycle) <br> between input and output values. |
|  |  | Synchronous <br> (Sync mode 2) | Extended application time, higher <br> lag (2 bus cycles) between input <br> and output values. |
| Optimize I/O <br> update | Off | On | When activated, consecutive I/Os <br> are merged in one block to opti- <br> mize the performance. |
|  |  | Off |  |

## EtherCAT-Master - ABB functionality for sync units

With the EtherCAT sync units, several slaves are configured into groups and subdivided into smaller units. For each group, the working counter can be monitored for an granular input data validation. As soon as a slave is missing in a sync unit group, the input data of all other slaves in the same group becomes invalid.

Detection occurs immediately in the next bus cycle, as the working counter is continuously checked. Unaffected groups remain operable without any interference.

Right click on the "Application" node and press "Create configuration data".
Automation Builder creates a set of global variables defining the working counter state of a "SyncUnit" command.

The variables use the following naming scheme:
"SLOT_" + "CouplerSlot_ " + "SyncUnitName" + "_CMD_" + "LogicalAccess" + "_FRAME_" + "FrameID CouplerSlot".

## CouplerSlot

The communication module slot is the ID of the slot where the communication module is plugged in.
SyncUnitName
The sync unit name is as defined in the "Sync Unit Assignment" tab.

## LogicalAccess

The logical access defines the command List of logical access commands:

- $\quad$ Read $=10$;
- Write = 11
- Read/Write = 12


## FrameID

The frame ID starts with 1 and increments if the cyclic exchanged data is larger than the maximum Ethernet frame boundary.

Values

- FALSE : Working counter is as expected (data from slaves is valid)
- TRUE : Working counter is different to expected value (data from slaves is invalid)

The variables can be used by conditional consumption of slave data in the application:

```
IF NOT SLOT_1_default_CMD_10_FRAME_1 THEN
(* Consume SyncUnit default slave data *)
END_IF;
```


## EtherCAT diagnosis

Automation Builder 2.3 provides an enhanced diagnosis interface for the EtherCAT fieldbus. The user can get EtherCAT diagnosis information from different editor views. All these views are accessible within the EtherCAT master device editor and provide information about the master and all configured or connected slaves. The main diagnosis overview is given in the EtherCAT master view "Diagnostics main".
"Diagnostics main" shows EtherCAT state "Operate".

"Diagnostics main" shows EtherCAT state "Topology error"'.


If the EtherCAT bus state shows "Operate", the user does not need to check for any more information.

If the "Diagnostics main" shows any error, like "Topology mismatch detected", the user can continue to the next level of information by opening editor view "Master State Control".
"Master State Control" shows EtherCAT state "Operate".

"Master State Control" shows EtherCAT state "Toplogy error".


In editor view "Master State Control" the user can request a master state change or get information about configured parameters as well as events and latest communication errors. In case of any topology error (e.g. slaves are configured in a different order than they exist in hardware) the Automation Builder shows a hint to the user that it might be helpful to execute a bus scan in editor view "Diagnostics live list" to compare the scan result of the real hardware with the configures slaves in the Automation Builder project.

Bus scan result in editor view "Diagnostics live list" shows the connected hardware.


The bus scan result list shows the following information for each connected slave:

| ID | Position of the found slave device |  |
| :--- | :--- | :--- |
| Type | Slave identification (name or Vendor/Device ID number) |  |
| State | The connection/link state of all ports (0-3) of the given <br> slave |  |
|  | Connected | => Cable is plugged in |
|  | + Link | => Physically connected to another slave |
| Name | => Communication works fine |  |
| Address | Not used for EtherCAT |  |
| Details | Not used for EtherCAT |  |

The bus scan shows information about the real connected hardware.
Note that a bus scan will always restart the EtherCAT bus.
This should not be a problem during commissioning but it might not be applicable in a running system.

For runtime diagnosis the Automation Builder provides cyclic information of all configured slaves and their states in the editor view "Slave diagnosis".

Slave diagnosis information shows that configured slaves are ok.

"Slave diagnosis" view shows wrong slave at position 1.


The editor view "Slave diagnosis" shows information about the configured slaves. If these slaves are found in hardware, the states of all slaves are ok. If there is a mismatch between hardware and configuration the view shows at which position that mismatch was detected.

The "Slave diagnosis" shows the following information for each configured slave:

| Topology Position | Position of the configured slave device. |
| :---: | :---: |
| Configured Station Address | Address that is defined by configuration. <br> This address is not topology dependent. |
| Slave Name | Configured name of the slave device. |
| Slave State | The state of the slave. Possible slave states are: |
| NOT CONNECTED ${ }^{1}$ ) |  |
| INIT |  |
| PREOP |  |
| SAFEOP |  |
| OP |  |
| INIT ERR |  |
| PREOP ERR |  |
| SAFEOP ERR |  |
| Port State ${ }^{2}$ ) | The state of all ports (0-3) of the given slave. |
|  | Shows how many connections this slave has to other slaves and if the connections are working fine: |
| Connected | => Cable is plugged in |
| + Link | => Physically connected to another slave |
| + Communication | => Communication works fine |
| Last Error | The last error that occurred in this slave. As text, if available, and error number. If this is any topology error, the editor view will show a hint to perform a bus scan. |
| Emergency [CAN application protocol over EtherCAT (CoE)] | This column contains up to 5 CoE emergency entries. Each entry has |
| Error code |  |
| Address of the error register |  |
| Error data (1 byte) |  |
|  | If there are more than five emergencies reported by the slave, the columns show a hint that some emergency entries have been lost. <br> The column is empty, if no CoE emergencies exist. |
| Frame Error Counters ${ }^{2}$ ) | Counts transmission errors on frame layer, detected by CRC check of frames. Fast growing values show a serious problem. Possible root causes include damaged cables, high electromagnetic noise or misbehavior of EtherCAT slave devices. Four counter values are shown, one for each port 0-3. Column has red background in case of any value other than 0 . |
| Physical Layer Error Counters ${ }^{2}$ ) | Counts transmission errors on physical layer. Possible root causes include electromagnetic disturbance or faulty devices. Four counter values are shown, one for each port $0-3$. Column has red background in case of any value other than 0 . |


| Link Lost Counters ${ }^{2}$ ) | Optional feature of EtherCAT slave devices, not supported <br> by every device. |
| :--- | :--- |
| Counts loss of physical connection (no link, LED off). Even <br> short interruptions can be detected. Possible root causes <br> include power dips, device reset, poor cables or connec- <br> tors, loose contact. Four counter values are shown, one <br> for each port 0-3. Column has red background in case of <br> any value other than 0. |  |

[^30]
#### Abstract



\section*{General note on the counters}

Please note that this kind of errors will be detected by devices when power state changes, e.g. when the device itself or a neighboring device is powered on, caused by switching artifacts on the cable. This does not signal an issue, only counters increasing during normal operation should trigger deeper analysis. Counters can be reset by the PLC program using corresponding function blocks.


### 6.3.2.11.4 <br> PROFIBUS

## General

The fieldbus PROFIBUS is supported in AC500 PLC as master and slave. The communication modules "CM592-DP PROFIBUS DP V0/V1 master module" and "CM582-DP PROFIBUS DP slave module" are provided for these purposes.

## Parameterization of the CM592-DP/CM582-DP communication modules

Configuration is valid as of CPU FW 3.5.0.
To append a communication module, add the communication module to the "Extension_Bus" node.

- Right-click the desired slot and select "Add object".
- Select the communication module from the list and click [Replace object].
- Double-click the new node to open the CM592-DP/CM582-DP PROFIBUS DP configuration in the editor window. Click on tab "CM592-DP/CM582-DP Parameters" if not already opened.
The following parameters are available:

| Parameter | Default <br> value | Value | Description |
| :--- | :--- | :--- | :--- |
| Run on config fault | No | No | In case of a configuration error, <br> the user program is not started. |
|  | Yes | The user program is started inde- <br> pendent of a faulty configuration <br> of the PROFIBUS communica- <br> tion module. |  |

Click on tab "CM592-DP/CM582-DP I/O Mapping" to open the "Bus Cycle Options" in the editor window.
The following parameter is available:

| Parameter | Default value | Value | Description |
| :--- | :--- | :--- | :--- |
| Bus cycle task | Use parent bus cycle <br> settings | Use parent bus cycle <br> settings | Settings from "PLC <br> settings" tab are used. |
|  | Task | Name of the task that <br> triggers the bus cycle |  |

Click on tab "CM592-DP/CM582-DP IEC-Objects" to open the list of used IEC Objects. For information instantiated I/O driver function block class is shown.

## CM592-DP PROFIBUS DP master communication module

## Configuration of a PROFIBUS DP master

Double-click on "Profibus_Master_x (Profibus_Master)" to open the "Profibus_Master" configuration in the editor window:

Click on "General" tab if not already opend.


Most of the parameters are calculated automatically. Uncheck [Use defaults] to enable values to be edited individually. Checking [Use defaults] again will revert all parameters to default values.

All times for the PROFIBUS parameters are given in bit time [tBit]. The bit time is the result of the reciprocal of the transmission rate:
$t$ Bit $=1 /$ transmission rate in [bit/s]

The conversion from milliseconds into a bit time is shown in following formula: tBit $=$ Time in [ms] * transmission rate in [bit/s]

The following parameters are available:

| Parameter | Default | Value | Description | Parameter <br> (Remark 1) |
| :---: | :---: | :---: | :---: | :---: |
| Adresses |  |  |  |  |
| Station address | 1 | 0... 125 | The individual device address of the master device on the bus. | DpParameter -> Station address |
| Highest station address | 126 | $0 \ldots 126$ | The highest bus address up to which a master searches for another master at the bus in order to pass on the token. This station address must on no account be smaller than the master station address. | DpParameter -> Highest station address |
| Mode |  |  |  |  |
| Auto-Clear mode | Enabled | Disabled | The master operation mode will stay in the mode 'Operate' and the communication to all available slaves is kept up. | AutoClearSupported |
|  |  | Enabled | The masters operation mode will change from 'Operate' to 'Clear' and it shuts down the communication to all assigned slaves, if at least 1 slave is not responding within the data control time. |  |
| Automatic startup | Enable | Disable | Do not perform automatic startup | AutoStart |
|  |  | Enable | Perform automatic startup |  |
| Parameters |  |  |  |  |
| Baud rate | 1500 | 9.6 19.2 45.45 93.75 187.5 500 1500 3000 6000 12000 | Data transfer speed in [kBits/s]. <br> The baud rate must be set to the same value for all devices on the bus. The result of changing the baud rate is that all other parameters must be recalculated. | DpParameter -> Baudrate |
| T_SL <br> Slot time | 300 | $\begin{aligned} & 37 \ldots \\ & 65535 \end{aligned}$ | Monitoring time of the sender (requester) of a telegram for the acknowledgement of the recipient (responder). After expiration, a retry occurs in accordance with the value of maximum telegram retries. | DpParameter -> TSL |


| Parameter | Default | Value | Description | Parameter <br> (Remark 1) |
| :---: | :---: | :---: | :---: | :---: |
| min. T_SDR <br> Minimum station delay responder time | 11 | 1... 65535 | Shortest time period that must elapsed before a remote recipient (responder) may send an acknowledgement of a received query telegram. The shortest time period between reception of the last bit of a telegram to the sending of the first bit of a following telegram. | DpParameter -> min. TSDR |
| max. T_SDR <br> Maximum station delay responder time | 150 | 1 ... 65535 | Longest time period that must elapse before a sender (requestor) may send a further query telegram. Greatest time period between reception of the last bit of a telegram to the sending of the first bit of a following telegram. <br> The sender (requestor, master) must wait at least for this time period after the sending of an unacknowledged telegram (e.g. broadcast only) before a new telegram is sent. | DpParameter -> max. TSDR |
| T_QUI Quiet time | 0 | 0 ... 127 | Time delay that occurs for modulators (modulator-trip time) and repeaters (repeater-switch time) for the change over from sending to receiving. | $\begin{aligned} & \text { DpParameter } \\ & \rightarrow>\text { TQUI } \end{aligned}$ |
| T_SET <br> Setup time | 1 | 0 ... 255 | Minimum period reaction time between the receipt of an acknowledgement to the sending of a new query telegram (reaction) by the sender (requestor). | $\begin{aligned} & \text { DpParameter } \\ & \text {-> TSET } \end{aligned}$ |
| T_TR <br> Target rotation time | 11894 | $\begin{aligned} & \hline 1 \ldots 2 \\ & -1 \\ & (=1677721 \\ & 5) \end{aligned}$ | Pre-set nominal token cycle time within the sender authorization (token). The available time for the master to send data telegrams to the slaves depends on the difference between the nominal and the actual token cycle time. <br> The Target rotation time (TTR) is shown in Bit times [tBit] like the other bus parameters. Below the displayed bit time, the Target rotation time is also displayed in [ms]. <br> The default value depends on the number of slaves attached to the master and their module configuration. | $\begin{aligned} & \hline \text { DpParameter } \\ & ->\text { TTR } \end{aligned}$ |


| Parameter | Default | Value | Description | Parameter <br> (Remark 1) |
| :---: | :---: | :---: | :---: | :---: |
| Gap <br> Gap update factor | 10 | 0 ... 255 | Factor for determining after how many token cycles an added participant is accepted into the token ring. After expiry of the time period G*TTR, the station searches to see whether a further participant wishes to be accepted into the logical ring. | DpParameter -> Gap update factor |
| Retry limit Maximum retries | 1 | $1 . .15$ | Maximum number of repeats in order to reach a station. | DpParameter -> max. retry limit |
| Data control time | 120 | $1 . . .22^{24}-1$ | Defines the time in [ms] within the Data_Transfer_List is updated at least once. After the expiration of this period, the master (class 1) reports its operating condition automatically via the Global_Control command. <br> The default value depends on the transmission rate. | DpParameter -> Data control time |
| Slave interval <br> Minimum slave interval | 2000 | 1... 65535 | Defines the minimum time period between two slave list cycles in [ $\mu \mathrm{s}$ ]. The maximum value the active stations require is always given. <br> The default value depends on the slave types. | DpParameter -> min. slave interval |
| Poll timeout Minimum poll timeout | 10 |  | Sets the maximum period of time in [ms] during which the response has to be received. | DpParameter <br> -> Poll timeout |

Remark 1:
To display the parameters of this column, enable the option "Show generic device configuration views" under "Tools $\rightarrow$ Options $\rightarrow$ Device editor".

## Configuration of a PROFIBUS DP slave

A PROFIBUS DP slave can be added by right-clicking on "Profibus_Master_x (Profibus_Master)" and selecting "Add object".
If the desired device is not listed it can be installed via the "Device Repository" (menu item "Tools" -> "Device Repository").
The slave configuration parameters can be edited in slave related editor window. To open this editor window, double-click the corresponding slave in the device tree.

Click on tab "General" if not already opened.


All times for the PROFIBUS parameters are given in bit time [tBit]. The bit time is the result of the reciprocal of the transmission rate:
tBit $=1$ / transmission rate in [bit/s]

The conversion from milliseconds into a bit time is shown in following formula:
tBit $=$ Time in [ms] * transmission rate in [bit/s]

The following parameters are available:

| Parameter | Default | Value | Description | Parameter <br> (Remark 1) |
| :---: | :---: | :---: | :---: | :---: |
| Identification |  |  |  |  |
| Station address | 1 | $0 \ldots 126$ | Station address of the PROFIBUS DP slave device. | StationAddress |
| Ident number | GSD file specific | ---- | Ident number of the PROFIBUS DP slave device. | SlavePrmData -> identNumber |
| Parameter |  |  |  |  |
| T_SDR (tBit) | 11 | $11 . .255$ | The parameter T_SDR (tBit) represents the minimum station delay of a responder (time a responder waits before generating the reply frame). | SlavePrmData -> minTsdr |
| Lock/unlock | 2 (Lock) | $\begin{aligned} & 0 \text { (T_SDR } \\ & \text { unlock) } \end{aligned}$ | The TSDR and slave-specific parameter may be overwritten. | Bit $6=0$ and bit $7=0$ of bitmask SlavePrmData -> stationStatus |
|  |  | 1 (Will be unlocked) | The slave is released to other masters. | Bit $6=1$ and bit $7=0$ of bitmask SlavePrmData -> stationStatus |
|  |  | 2 (Lock) | The slave is locked to other masters, all parameters are accepted. | Bit $6=0$ and bit $7=1$ of bitmask SlavePrmData -> stationStatus |


| Parameter | Default | Value | Description | Parameter <br> (Remark 1) |
| :---: | :---: | :---: | :---: | :---: |
|  |  | 3 (Unlock) | The slave is released to other masters. | Bit $6=1$ and bit $7=1$ of bitmask SlavePrmData -> stationStatus |
| Watchdog |  |  |  |  |
| Watchdog control | Enabled | Disabled | The PROFIBUS Slave does not utilize the Watchdog Control Time setting. | SlavePrmData -> wdFact1 |
|  |  | Enabled | The PROFIBUS slave utilizes the Watchdog Control Time setting in order to detect communication errors to the assigned Master. When the Slave finds an interruption of an already operational communication, defined by a Watchdog time, then the Slave carries out an independent Reset and places the outputs into the secure condition. |  |
| Time (ms) | 400 | 0 ... 2540 | Watchdog time in [ms]. <br> The default value depends on the number of slaves attached to the master and their configuration. | SlavePrmData -> wdFact2 |
| User parameter |  |  |  |  |
| Symbolic values | Enabled | Disabled | No symbolic names for the user parameters. | - |
|  |  | Enabled | The values for the parameters are shown with symbolic names. | - |
| Length of user parameter (Byte) | 3 | Devicespecific | The length of the user parameters in [bytes]. By default this value is 3 due to the existing reserved values. | - |
| Defaults | - | - | The button restores the default values of the user parameters. | - |

Remark 1:
To display the parameters of this column, enable the option "Show generic device configuration views" under "Tools $\rightarrow$ Options $\rightarrow$ Device editor".

## Click on tab "Check configuration".

Following dialog shows values for input/output/parameter data occupied by your configuration. Here it can be checked how much data is left for further configuration.


## CM582-DP PROFIBUS DP slave communication module

## Configuration of PROFIBUS DP slave

Double-click on "PROFIBUS_DP_Slave" to open the PROFIBUS slave configuration in the editor window:


The following parameters can be modified:

| Parameter | Default value | Value | Description |
| :--- | :--- | :--- | :--- |
| Bus address | 1 | $0 \ldots 126$ | The bus address is the individual device <br> address of the slave device on the bus. |

## Configuration of I/O data objects

To append I/O data, add the desired input / output objects to the Communication Module node.
Right-click the Communication Module node and select "Add object".
Different types of data objects group I/O variables by size and direction. The I/O driver of the PLC firmware copies the amount of data bytes configured by these data objects cyclically.


Select the desired I/O objects from the list and click [Add object].


To keep basic load of PLC low, only configure as much I/O data objects as actually required. If further I/O variables need to be added later, additional data objects can be inserted.

Technical details on the device such as the maximum amount of bytes used for I/O data is described in the device specification for ${ }^{\leftrightarrows} \Rightarrow$ Chapter 5.2.2.8.1 "CM582-DP - PROFIBUS DP slave" on page 354.

Double-click an added I/O object node to open the preset configuration. As the I/O objects do not need user configuration all parameters in the "Parameters" tab are read-only.

Open the "I/O Mapping" tab to configure the mapping configuration for the I/O object. sistency

Possible incon- On using CM582-DP slave device configured with modules types combining input and output data the following situation may happen:

## Example CM582－DP communication module configuration uses module type 16 Byte In／Out．

The device representation assigned to CM592－DP master uses module types 16 Byte Output and 16 Byte Input at the same place instead．

臨 CM592＿DP（CM592－DP）
－Profibus＿Master（Profibus－Master）
－ －CM582＿DP＿Slave（CM582－DP－Slave）
［813］＿4＿Byte＿Input＿7（4 Byte Input）
10＿${ }^{16}$＿Byte＿Output（ 16 Byte Output）
＿16＿Byte＿Input（16 Byte Input）
的解 CM582＿DP（CM582－DP）
－臨 ${ }^{2}$ PROFIBUS＿DP＿Slave（PROFIBUS－DP－Slave）
5ala＿4＿Byte＿Input＿8（4 Byte Input（IEC output））
＿16＿Byte＿In＿Out（16 Byte In／Out）

This mismatch will not be detected；neither by Automation Builder nor by PROFIBUS master and slave．

However，the communication will run stable and I／O data exchange is executed successfully．

## Reason：

AC500 defines modules combining input and output directions to be split to two separated module configurations internally with output direction first．
Thus in AC500 the PROFIBUS configuration data for one module of type 16 Byte In／Out looks the same as for the combination of module types 16 Byte Output and 16 Byte Input．

## Mapping of the I／Os

Double－click on the desired I／O data object in the device tree to show current I／O mapping connected to this data object $\Longleftrightarrow$ Chapter 6．3．2．13．7＂Symbolic names for variables，inputs and outputs＂on page 1572.

## 6．3．2．11．5 Ethernet

CM5640－2ETH－Ethernet communication modul

Configuration is valid as of CPU FW 3．6．0．

## Configuration of the communica－ tion module

Append a CM5640－2ETH．

All interfaces of the onboard and CM5640－2ETH modules need to be in dif－ ferent subnets．Overlapping of subnets must be avoided．The selection of the used interface（ETH port）is done automatically according the match of the IP－address of the server and the subnet of the interfaces．

1. Right-click on your desired Slot below node "Extension_Bus" and click "Add object". $\Rightarrow$ Dialog "Replace object:" appears.
2. Click "CM5640-2ETH" in the list and click the button [Replace object].

Double-click "CM5640-2ETH (CM5640-2ETH)" to get the "CM5640-2ETH Parameters" in the editor window.
The communication module CM5640-2ETH is a port extender of the processor module used.
Each communication module adds 2 additional independent ethernet interfaces. The communication module in the configuration tree is displayed in the same way as the processor module onboard ethernet interfaces.

The configuration of both interfaces is done as for the onboard ethernet interfaces from a processor module.


Double-click "ETH1_1 (IP Setting)" will open the windows to select the possible server protocols and all non-available protocols for that interface are greyed.
The added feature appears below "NetConfig (NetConfig)" for example and the configuration of each feature is done by clicking on the selected feature and opening new windows.

Both interfaces can be independently configured and the available server protocols always displayed when available or greyed when disabled.
Selecting "Protocols_1 (Client Protocols)" and right click and use "Add object", a new window opens showing the available client protocols.


The configuration of each protocol either server or client is done in the same way as for onboard ethernet of the processor module.

### 6.3.2.12 Communication interface modules

### 6.3.2.12.1 Configuration of communication interface modules

Automation Builder can be used to configure the parameters of Cl 5 xx devices.

$$
\begin{aligned}
& \text { Configuration of S500 I/O modules can be performed without CI5xx devices } \\
& \text { connected. }
\end{aligned}
$$

## Adding Cl 5 xx device to the device tree

1. Right click in the device tree on the node "Slot1" or "Slot2" of the "Extension_Bus" and click "Add object".
$\Rightarrow$ The window "Replace object : Slot <...>" opens.
2. Select your "CM5xx master module" and click [Add object].
$\Rightarrow$ The "CM5xx master" appears in the Slot.
3. Right click on "CM5xx master module" and click "Add object".
$\Rightarrow$ The window "Add object below : <...>_Master" opens.
4. Select your "CI5xx" device and click [Add object].
$\Rightarrow$ The "Cl5xx" device appears in your device tree.

Adding S500 I/O modules

1. Right click on your "CI5xx" device and click [Add object].
$\Rightarrow$ The window "Add object below:" opens.
2. Select your I/O module and click [Add object].
$\Rightarrow$ The I/O module is added.

## Configure parameters

$\triangleright \quad$ Double-click the "CI5xx" device to open editors and select the "CI5xx_IO Parameters" tab.

This editor shows the parameters that can be set for each device , and .

### 6.3.2.12.2 Unbundled CI52x-MODTCP configuration

Automation Builder can be used to configure the parameters of CI52x-MODTCP devices.

## $\square$ <br> A direct Ethernet connection is required between the PC running Automation Builder and the CI52x-MODTCP module.

## Start a project from template

1. Select "New Project" in menue item "File".
$\Rightarrow$ The window "New Project" appears.
2. Select the "CI52x-MODTCP Configuration Project" and click "OK".
$\Rightarrow$ The window "Select PLC" opens.
3. Select a "CI52x-MODTCP" device and click "Add device".
$\Rightarrow$ A project is created. More modules can be added.

Add CI52x-
MODTCP to a project

1. Right click in the device tree on the root of the "Project" and click "Add object".
$\Rightarrow$ The window "Add object below" opens.
2. Select "Modbus devices" and click "Add object".
$\Rightarrow$ The node"Modbus_devices" appears in your device tree.
3. Right click on the node "Modbus_devices" and click "Add object".
$\Rightarrow$ The window "Select PLC" opens.
4. Select your "CI52x-MODTCP" device and click "Add device".
$\Rightarrow$ The "CI52x-MODTCP" device appears in your device tree.

Add S500 I/O modules

1. "Add object" to your "CI52x-MODTCP" device.
$\Rightarrow$ The window "Add object below: CI52x-MODTCP" opens.
2. Select your I/O module and click "Add object".
$\Rightarrow$ The I/O module is added.

## Configure

 parametersConnect to device
$\triangleright$ Double-click the device to open editors and select the "CI52x-MODTCP Parameters" tab.

This editor shows the parameters that can be set for each device. For more information see $\begin{gathered} \\ \end{gathered}$ 5.2.6.5.2.8.1 "Parameters of the module" on page 1120 CI 522 and.

| O CIS22_MODTCP $\times$ |  |  |  |  | - |
| :---: | :---: | :---: | :---: | :---: | :---: |
| CI522-MODTCP Parameters | Parameter | Type | Value | Default Value Unit | Description |
| Connection Settings Information | - Error LED / Falsafe function | Enumeration of BYTE | on | On | Error LED off by error class |
|  | - Master-P 0 | ARRAY[0.-3] OF BYTE | [ $0,0,0,0$ ] | [ $0,0,0,0$ ] | Master-IP 0 for write restriction |
|  | - Master-IP 1 | ARRAY[0.3] OF BYTE | [ $0,0,0,0$ ] | [0,0,0,0] | Maste-1P1 1 for write restricton |
|  | - Master-PP2 | ARRAYY[0.-3] OF BYTE | [ $0,0,0,0$ ] | [ $0,0,0,0$ ] | Master-1P 2 for write restriction |
|  | - Master-IP 3 | ARRAY[0..3] OF BYTE | [ $0,0,0,0]$ | [0,0,0,0] | Master-1P 3 for write restrition |
|  | - Master-IP 4 | ARRAY[0.3] OF BYTE | [ $0,0,0,0$ ] | [ $0,0,0,0$ ] | Master-Ip 4 for write restriction |
|  | - Master-IP 5 | ARRAY[0.-3] OF BYTE | [ $0,0,0,0$ ] | [0,0,0,0] | Master-1P 5 for write restriction |
|  | - Master-P 6 | ARRAY [0.3] OF BYTE | [ $0,0,0,0$ ] | [ $0,0,0,0$ ] | Master-IP 6 for write restriction |
|  | - Master-IP 7 | ARRAY[0.-3] OF BYTE | [ $0,0,0,0]$ | [ $0,0,0,0]$ | Master-19 7 for write restricton |
|  | - Tmeout for bus supervision | BYTE(0.255) | 0 | 0 | Teeout for bus supervson. Number of 1 c |
|  | - ע/O mapping structure | Enumeration of BYTE | Fxied mapping | Fxed mapping | Fixed Mapping means each module has is - |
|  | - check supply | Enumeration of BYTE | On | On | Check supply |
|  | - Input delay | Enumeration of BYTE | 8 ms | 8 ms | Input delay of digta inputs |
|  | - Fast counter | Enumeration of BYTE | O-No counter | 0-No counter | Operating mode fast counter |
|  | - Detect short crout at outputs | Enumeration of BYTE | On | On | Check for short crout |
|  | - Behavour digtal outputs at comm. error | Enumeration of BYTE | off | off | Output value by communication error of L/) |
|  | - Substrute valie digtal outputs | WORD(0..65535) | 0 | 0 | Set the substute value |
|  | - Preverive votage feedasck montorng | Enumeration of BYTE | off | off | Preventive voltage feedback montorng |
|  | - Overvotage behavicur on output | Enumerabion of BYTE | off | off | Overvotage behaviour on output |
|  | - $\square^{\text {I }}$ | $\square$ |  |  | , |

To read or write parameters, the $\mathrm{CI} 52 \mathrm{x}-\mathrm{MODTCP}$ module must be connected to the PC with an Ethernet connection.

See ${ }^{\star} \Rightarrow$ Chapter 5.2.6.5.1.6 "Addressing" on page 1089 CI 521 and $\Leftrightarrow$ Chapter 5.2.6.5.2.6 "Addressing" on page 1120 CI 522 of the CI52x-MODTCP hardware documentation for information on configuring the IP address of the device.

On the CI52x-MODTCP device editor, the "Connection Settings" tab allows the IP address of the device to be entered.


Read Reads the parameters from the CI52x-MODTCP and also for the attached S500 I/O modules.
Write Sends the parameters from the editors to the CI52x-MODTCP and also the S500 I/O modules.

Device checking The CI52x-MODTCP module knows which I/O modules are attached.
While reading and writing parameters, the project must match the physical hardware. Otherwise an error will be given.
Communication errors will also result in error messages.
When the parameters have been read or written correctly, a message is seen in the "All messages" window:


Attached S500 modules

[^31]It is possible to read and write parameters when the S500 I/O modules are not attached to the CI52x-MODTCP module.

To perform a read, the project structure must still match the configuration of CI52x-MODTCP.

A warning will be shown if an I/O module is not detected:


When writing parameters, the Cl52x-MODTCP configuration is overwritten so the current configuration of missing (unplugged) modules does not matter.
If the I/O modules are attached, then the project must match the hardware, otherwise an error will be given.

### 6.3.2.13 I/O bus and I/O modules

### 6.3.2.13.1 Hot swap configuration

Table 302: Parameters for hot swap configuration of I/O extension modules

| Parameter | Purpose | Value |
| :--- | :--- | :--- |
| Hot-swap terminal unit <br> required | To include diagnosis <br> for missing hot-swap ter- <br> minal unit | Yes: <br> Communication interface provides <br> extended diagnosis for missing hot- <br> swap terminal unit |
|  |  | No (default): <br> Extended diagnosis not available |
| Start-up with missing <br> module on hot-swap ter- <br> minal unit | Ignore missing module <br> during start-up on hot- <br> swap terminal unit. <br> Incomplete I/O configura- <br> tions must not prevent the <br> system from starting. | Yes: <br> Module is optional, start-up if there is <br> no module available on hot-swap ter- <br> minal unit |
|  | No (default): <br> Module is mandatory, start-up only if <br> correct module is available |  |

In the Automation Builder projects for V3 PLCs, hot-swap parameters can be configured from "Module Parameters" tab of respective I/O module.

| 畋 DC532 $\times$ CI501_IO |  |  |  |
| :---: | :---: | :---: | :---: |
| Diagnosis | Parameter | Type | Value |
| General | Hot-swap terminal unit required | Enumeration of BYTE | Yes |
|  | - Start-up with missing module on hot-swap terminal unit | Enumeration of BYTE | Yes |
| PNIO Module Parameters | - Config version | UDINT | 16\#03050900 |
|  | - Module parameters |  |  |

By default, the "Module Parameters" tab is not visible for parameter configuration.
Follow the below steps to enable this tab

1. In the Automation Builder menu select "Tools $\rightarrow$ Options"
2. Select "Device editor" option from "Options" dialog.
3. Enable the option "Show generic device configuration views" (if not already done)
4. To save the settings and close the dialog select [OK]

| Options |  | $\times$ |
| :---: | :---: | :---: |
| Automation Builder <br> 自 f) C Compiler <br> 聿 CFC Editor CoDeSys 2.3 converter Debugging Declaration Editor Device description download Device editor <br> Diagnosis External tools FBD, LD and IL editor Help Help IEC 60870-5-104 <br> International Settings <br> Libraries <br> Library download <br> Load and Save | Device editor <br> View Show generic device configuration views Create Cross References for IEC addresses (Clean necessary) <br> Communication page <br> Simple mode Show implicit files for application download on the editor of a PLC Show access rights page $\square$ Use horizontal tab pages |  |
|  | OK | Cancel |

### 6.3.2.13.2 Parameterization of the I/O bus

Double-click the "IO_Bus" node in the device tree to open the I/O bus configuration. The following parameters are available:

| Parameter | Default | Value | Description |
| :--- | :--- | :--- | :--- |
| Run on config fault | No | No | In case of configura- <br> tion fault the user <br> program will not be <br> launched. |
|  |  | Yes | The user program will <br> be also launched in <br> case of configuration <br> error on the I/O Bus. |
| Max wait run | 3000 | $0 \ldots 120000$ | Maximum waiting time <br> for valid inputs. |

In case of a digital I/O Module, the channels are provided as WORD, BYTE and BOOL. Because the analog inputs can also be configured as digital inputs, bit 0 of each channel is also available as BOOL.
The symbolic name of a channel can be entered in front of the string "AT" in the channel declaration.


All channels should have a symbolic name and only symbolic names should be used in the program code. If the hardware configuration has changed or if you want to download the project to a PLC with another hardware configuration and thus the PLC configuration has to be changed, the addresses of the inputs and outputs can change. In case of symbolic programming (i.e., symbolic names are used), the program code does not have to be changed.

### 6.3.2.13.3 Parameter 'Ignore module'

All I/O devices provide the parameter "Ignore module". This parameter can be used for simulation purposes and determines whether an I/O device is considered or ignored during a PLC configuration check.

This allows to use an existing Automation Builder project/PLC configuration though some hardware devices are not physically available in a hardware installation.

Example The Automation Builder project for machine A shall be used for machine B. However, the second DC523 device is missing in the hardware installation of machine B. Hence, for machine $B$ the value for "Ignore module" is set to "YES".

| ```-5 Project2 = III PLC_AC500_V3 (PM5675-2ETH) * [1] PLC Logic = IO_Bus(I/O-Bus)``` <br> ```AO523_for_machine_A_B (AO523) ``` <br> ```DC523_for_machine_A_B (DC523) \(\square\) DC523_for_machine_A_only (DC523) ``` <br> ```Interfaces (Interfaces) \\ = [**) Extension_Bus (Extension Bus) ``` | [] DCS23_for_machine_A_only | $\times$ |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  |  |
|  | I/O mapping list | Parameter | Type | Value | Default Value | Unit |
|  |  | - Ignore module | Enumeration of ... | Yes | No |  |
|  | DC523 Parameters | - Check supply | Enumeration of ... | On | On |  |
|  | DC523 I/O Mapping | - Input delay | Enumeration of ... | 8 ms | 8 ms |  |
|  |  | - Fast counter | Enumeration of... | 0-No counter | 0-No counter |  |
|  | Status | - Detect short circuit at outputs | Enumeration of... | On | On |  |
|  |  | - Behaviour outputs at comm, error | Enumeration of ... | Off | Off |  |
|  | Information | - Substute value | DWORD(0..167... | 0 | 0 |  |

### 6.3.2.13.4 I/O bus - Bus cycle task

General infor- The term "bus" includes all fieldbuses as well as the ${ }_{\xi}{ }^{\mu} / / O$ bus. Consider that there is no bus mation cycle task for Modbus as Modbus does not provide I/O mapping and is controlled by POUs.

It's recommended to define a dedicated bus cycle task for each fieldbus configured in the project. It's strongly recommended not to use "unspecified" in the ""PLC Settings" to avoid unexpected behavior. The task defined in "PLC Settings" determines the bus cycle task of I/O bus and, depending on the configuration, of the additional fieldbuses (the setting is by default inherited).
Especially in case of EtherCAT, a dedicated bus cycle task should be used which is not shared with other fieldbuses. If [unspecified] is set in the "PLC Settings", the EtherCAT task might be automatically used by other fieldbuses, potentially causing the EtherCAT task processing to fail. This should be avoided by specifying a task different to the EtherCAT task in the "PLC Settings".

As a rule, for each IEC task the used input data is read at the start of each task and the written output data is transferred to the I/O driver at the end of the task. The implementation in the I/O driver is decisive for further transfer of the I/O data. The implementation is therefore responsible for the timeframe and the specific time when the actual transmission occurs on the respective bus system.

Other tasks copy only the I/O data from an internal buffer that is exchanged only with the physical hardware in the bus cycle task.

(1) Read inputs from input buffer
(3) Write outputs to output buffer
(5) Input buffer
(7) Copy data to/from bus
(9) Bus cycle task, priority 1, 1 ms
(10) Bus cycle task, priority 5
(11) Bus cycle task, priority 10, interrupted by task 5

## Using tasks

The "Task Deployment" provides an overview of used I/O channels, the set bus cycle task, and the usage of channels.

## WARNING!

If an output is written in various tasks, then the status is undefined, as this can be overwritten in each case.

When the same inputs are used in various tasks, the input could change when a task is processed. This happens if the task is interrupted by a task with a higher priority and causes the process map to be read again. Solution: At the beginning of the IEC task, copy the input variables to variables and then work only with the local variables in the rest of the code.
Conclusion: Using the same inputs and outputs in several tasks does not make any sense and can lead to unexpected reactions in some cases.

### 6.3.2.13.5 Insertion of S500 I/O devices

1. Right-click "IO_Bus" in device tree and select [Add object].
-> The "Add Device" dialog window where all available S500 I/O devices are listed will open.
2. Append the S500 I/O devices in the same order as they are mounted on the hardware.

Input and output modules connected to the I/O bus occupy the I/O following area: \%IB0 ... \%IB999 or \%QB0 ... \%QB999.


### 6.3.2.13.6 Configuring the input and output modules and channels

The I/O channel configuration depends on the corresponding S500 I/O device.
The individual configuration parameters can be opened in the editor window via double-click on the corresponding module and are listed in tab "S500 I/O device name Configuration".

### 6.3.2.13.7 Symbolic names for variables, inputs and outputs

The IEC naming rules are not checked during input in Automation Builder.

Input and output Devices with I/Os provide an "//O Mapping" tab in their configuration editor where the available mapping $\quad \mathrm{I} / \mathrm{O}$ channels can directly be mapped to a global variable.

The corresponding variable declarations are automatically available in the project.
All available I/O channels can easily be assigned to a variable.

| Variable - b_Input_IB0 | Mapping | Channel <br> Digital inputs I0 - I7 | Address \%IBO | Type <br> BYTE | Default Value | Unit | Description |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| x_Input_IB0_IX0 | * | Digital input IO | \%IX0.0 | BOOL |  |  |  |
| * x_Input_IB0_IX1 | * | Digital input I1 | \%IX0.1 | BOOL |  |  |  |
| * x_Input_IB0_IX2 | - | Digital input I2 | \%IX0.2 | BOOL |  |  |  |
| x_Input_IB0_IX3 | * | Digital input I3 | \%IX0.3 | BOOL |  |  |  |
| * x_Input_IB0_IX4 | * | Digital input I4 | \%IX0.4 | BOOL |  |  |  |
| * x_Input_IB0_IX5 | - | Digital input 15 | \%IX0.5 | BOOL |  |  |  |
| * x_Input_IB0_IX6 | , | Digital input I6 | \%IX0.6 | BOOL |  |  |  |
| x_Input_IB0_IX7 | * | Digital input I7 | \%IX0.7 | BOOL |  |  |  |
| + |  | Digital inputs I8-I15 | \%IB1 | BYTE |  |  |  |
| $\pm 4$ |  | Digital inputs I16-I23 | \%IB2 | BYTE |  |  |  |
| $\dagger$ |  | Digital inputs I24-I31 | \%IB3 | BYTE |  |  |  |
| $\pm$ Fast counter |  |  |  |  |  |  |  |

AC500 uses Intel Byte Order (Little Endian).

Only entries with a data type set in column "Type" can be mapped. These entries can be expanded to show the available I/O channels.
If the project has been imported from a previous Automation Builder version, all variables should be checked to avoid inconsistencies concerning the I/O mapping.

### 6.3.2.13.8 I/O mapping list

Introduction
Automation Builder contains an I/O mapping list feature for creating mapping variables with better usability support compared to the tree structured view.
«) Details on the tree structured view.


Functionalities of the I/O mapping list:

- Displays I/O mappings for current node and all valid subsequent child nodes.
- Displays channel information with additional columns.
- Supports keyboard functions such as cut, copy, paste, delete, and select all within the editor and within Excel spreadsheet (for bulk editing).
- Contains a toolbar for various actions, e.g. filtering, undo/redo and clear mappings.
- Supports single click edit and easy navigation using arrow keys.
- Improvised error handling:
- Allows to enter invalid mapping variables. This provides flexibility in bulk editing. Only when saving the project, the errors - according to IEC 61131 standard - are displayed.
- In the message window, the error log is visible. The user can track the errors to their corresponding channel in the editor.
- Allows multi-selection of rows and columns. (Random selection is not allowed.)


## Editing I/O mapping list

1. In the device tree, double-click "IO_Bus" to configure entire I/O mapping list of different I/O devices.
2. Enter the variables and descriptions to map the I/O devices.

Do not start variable names with a number or a special character. When saving the project, this generates an error. Example: 12input3, @input4.
3. Click "Save Project" to save the I/O mapping changes.

## Configuring I/O mapping list

Automation Builder supports tree and list based editors for creating I/O mapping variables.

1. From the "Tools" menu, select "Options".
2. Under Automation Builder, select the "Editors" tab.
3. Choose your desired mapping dialog and click [OK].

- Choose "tree based" to display the I/O mapping in tree structure.
- Choose "list based" to display the I/O mapping as list with the functionalities of the ToolBar.
- Choose "both" to display both the tree structure ("//O Mapping" tab) and the list view ("/O mapping list" tab).


## Available The I/O mapping list displays the channel information in offline and online mode. In online channel information mode, all columns are read-only. In offline mode, some columns are editable.

The order of the devices in I/O mapping list is synchronized with the order in the device tree.
The channels of a device are ordered by the device description file. If channels have a section, the channel information is represented in a specific format.
Example: Fast counter: Actual value 1. These channels are listed at last position of a device.

Filtering $\quad \begin{aligned} & \text { Especially in case of long I/O mapping lists, it might be helpful to filter the I/O mappings. For } \\ & \text { this, click the "Filter" icon to display all available criteria for filter options. }\end{aligned}$ this, click the "Filter" icon to display all available criteria for filter options.


When reducing the width of the editor, some filters might be hidden.

Undo, redo and - "Undo": Cancels the last change.
clear - "Redo": Repeats the last change.

- "Clear mappings": Deletes all variables and descriptions.


### 6.3.2.13.9 Fast counter

## Configuration for $\mathbf{S} 500 \mathrm{I} / \mathrm{O}$ modules

## Configuring the fast counter

1. In the device tree, add a digital I/O module to the "IO-Bus" node.
2. Double-click the node for the I/O module, open the "Parameters" tab and set the operating mode of the "Fast counter" parameter.
\# Chapter 6.3.2.13.9.1.3 "Operating modes" on page 1577

3. In the "//O Mapping" tab channel configuration is displayed.

* Chapter 6.3.2.13.9.3 "Control of the fast counter" on page 1583


## Operands

Table 303: Input information

| Description of the <br> input information | Output information of <br> the user program | Description |  |
| :--- | :--- | :--- | :--- |
| Start value 1 | Output double word 0 | Double word | Set values for the counters 1 <br> and 2: <br> Each counter can be set to <br> a start value. Start values are <br> loaded into the counter by the <br> user program. Using the set <br> signal (depending on the oper- <br> ating mode either via a terminal <br> or the bit SET within the con- <br> trol byte 1 or 2), the values of <br> the double word variables are <br> loaded into the counter 1 or 2. |
| Start value 2 | Output double word 1 | Double word |  |
| End value 1 |  | Output double word 2 | Double word |
| End value 2 | Output double word 3 | Double word | End value for the counters 1 <br> and 2: <br> The end values for the <br> two counters are stored as <br> comparison values into the <br> module by the user program. <br> Both counters compare contin- <br> uously whether or not their pro- <br> grammed end value is equal <br> to their actual value. When the <br> counter (actual value) reaches <br> its programmed end value, the <br> binary output CF of the status <br> byte is set permanently. |


| Description of the input information | Output information of the user program | Description |  |
| :---: | :---: | :---: | :---: |
| Control byte 1 see 1) | Output byte 0 | Byte: <br> Bit $0=$ UP/DWN <br> Bit $1=E N$ <br> Bit $2=$ SET <br> Bit $3=$ CF_HW <br> Bit 4 to Bit 7 <br> free | Control bytes for the counter 1: <br> UP/DWN: In some operating modes, the counter can count downwards, too. If counting down is desired, set the bit UP/DWN to TRUE and the bit SET to 1 . When doing so, the counter starts counting downwards from the start value (set value) to the end value (max. from 4,294,967,295 to 0 or hexadecimal from FF FF FF FF to 00000000 ). After reaching 0 the counter jumps to $4,294,967,295$. <br> EN: Processing of the counter signals must be enabled. Depending on the operating mode, enabling is done via a terminal or by the bit $\mathrm{EN}=$ TRUE within the control byte. <br> SET: The counter can be set to a start value (see the description of the set values for the counters 1 and 2 at the beginning of this table. <br> CF_HW <br> $0=$ state of CF is set to hardware channel (only for mode 1 and 2) <br> 1 = normal output is set to hardware channel <br> Bit 3 is evaluated only in control byte of counter 1 . |
| Control byte 2 <br> see 1) | Output byte 0 | Byte: <br> Bit $0=$ <br> UP/DWN <br> Bit $1=\mathrm{EN}$ <br> Bit $2=$ SET <br> Bit 3 to Bit 7 free | Control bytes for the counter 2: <br> UP/DWN: In some operating modes, the counter can count downwards, too. If counting down is desired, set the bit UP/DWN to TRUE and the bit SET to 1 . When doing so, the counter starts counting downwards from the start value (set value) to the end value (max. from 4,294,967,295 to 0 or hexadecimal from FF FF FF FF to 000000 00). After reaching 0 the counter jumps to 4,294,967,295. <br> EN: Processing of the counter signals must be enabled. Depending on the operating mode, enabling is done via a terminal or by the bit EN = TRUE within the control byte. |


| Description of the <br> input information | Output information of <br> the user program | Description |  |
| :--- | :--- | :--- | :--- |
|  |  |  | SET: The counter can be set to <br> a start value (see the descrip- <br> tion of the set values for the <br> counters 1 and 2 at the begin- <br> ning of this table. |

1) Only for $\mathrm{Cl} 581-\mathrm{CN} / \mathrm{Cl} 582-\mathrm{CN}$ : Control bytes 1 and 2 are available twice on grounds of data consistency. Hence, a Start and End evaluation is only effected if the signals "Control Byte1_0" and "Control Byte1_1" or "Control Byte2_0" and "Control Byte2_1" (process image) are identical.

Table 304: Output information

| Output informa- <br> tion | Input information for <br> the user program | Description |  |
| :--- | :--- | :--- | :--- |
| Actual Value 0 | Input double word 0 | Double word | Actual value of the counter 0 |
| Actual Value 1 | Input double word 1 | Double word | Actual value of the counter 1 |
| Status Byte 0 | Input byte 0 | Byte: | CF: When the counter rea- <br> ches the programmed end <br> value, the counter output is <br> stored permanently as CF $=$ <br> TRUE (end value reached). <br> Only when the counter is set <br> again (set value), CF is reset <br> to FALSE. |
| Status Byte 1 | Input byte 1 | Bit $0=$ CF <br> Bit 1 to Bit 7 free |  |

## Operating modes

Inputs and outputs which are not used by the counters, are available for other tasks.
Legend:

- A refers to input channel A
- B refers to input channel B
- C refers to output channel C

| Operating mode | Function | Used inputs and out- <br> puts | Notes |
| :--- | :--- | :--- | :--- |
| 0 | No counter | none | This operating mode <br> is selected if the inte- <br> grated fast counter is <br> not necessary. |
| 1 | One count up counter | A = Counting input <br> C = End value <br> reached | The counting input <br> and the output "End <br> value reached) are <br> enabled by the bit EN <br> = TRUE within the <br> control byte. |


| Operating mode | Function | Used inputs and outputs | Notes |
| :---: | :---: | :---: | :---: |
| 2 | One count up counter with enable input via terminal | $\begin{aligned} & A=\text { Counting input } \\ & B=\text { Enable input } \\ & C=\text { End value } \\ & \text { reached } \end{aligned}$ | The enable input enables the counting input and the output "end value reached". The counter is only enabled if the enable input = TRUE (signal 1) AND the bit $E N=$ TRUE within the control byte. |
| 3 | Two up/down counters | $\begin{aligned} & A=\text { Counting input } 0 \\ & B=\text { Counting input } 1 \end{aligned}$ | With this operating mode, two counters exist, which are independent of each other. The state "End value reached" is only readable from the two status bytes. It is not readable from output terminals. <br> The counting direction is defined by the bit UP/DWN within the control byte. |
| 4 | Two up/down counters (1 counting input inverted) | $\begin{aligned} & A=\text { Counting input } 0 \\ & B=\text { Counting input } 1 \end{aligned}$ | This operating mode equals operating mode 3 with one exception: The counting input B (of counter 1) is inverted. It counts the TRUE/ FALSE edges at input B. |
| 5 | One bidirectional counter with a dynamic set input via terminal | $\begin{aligned} & A=\text { Counting input } \\ & B=\text { Dynamic set input } \end{aligned}$ | With this operating mode, one bidirectional counter is available which has a dynamic set input. Dynamic means that the set operation is performed at the FALSE/TRUE signal edge ( $0 / 1$ edge) of the set input and not while the signal is TRUE. <br> The state "End value reached" is only readable from the status byte, not from an output terminal. |
| 6 | One bidirectional counter with a dynamic set input via terminal | $\begin{aligned} & A=\text { Counting input } \\ & B=\text { Dynamic set input } \end{aligned}$ | This operating mode equals operating mode 5 with one exception: The dynamic set input operates at the TRUE/ FALSE edge (1-0 edge). |


| Operating mode | Function | Used inputs and outputs | Notes |
| :---: | :---: | :---: | :---: |
| 7 | One bidirectional counter for position sensors | A = Trace A of the position sensor $B=$ Trace $B$ of the position sensor | With this operating mode, incremental position sensors can be used which interchange their counting signals on tracks A and $B$ in a $90^{\circ}$ phase sequence. Depending on the sequence of the signals at $A$ and $B$, the counter counts up or down. There is no pulse-multiplier function (e.g. x2 or $x 4$ ). The position sensor must provide 24 V signals. Signals of 5 V sensors must be converted. Zero traces are not processed. The state "End value reached" is only readable from the state byte 0 , not from an output terminal. <br> The bit UP/DWN within the control byte must be FALSE. Otherwise, a parameter error occurrs. <br> In this operating mode, the maximum counting frequency is: <br> $\mathrm{I} / \mathrm{O}$ modules 35 kHz . <br> Communucation interface modules 50 kHz . |
| 8 | Reserved |  |  |


| Operating mode | Function | Used inputs and outputs | Notes |
| :---: | :---: | :---: | :---: |
| 9 | One bidirectional counter for position sensors (pulse multiplier x2) | A = Trace A of the position sensor B = Trace B of the position sensor | This operating mode equals operating mode 7 with one exception: There is a pulse multiplication $\times 2$ with the evaluation of the counting inputs. This means, that the counter counts both the positive edges and the negative edges of trace A. This results in the double number of counting pulses. The precision increases correspondingly. <br> In this operating mode, the maximum counting frequency is: <br> I/O modules 30 kHz . <br> Communucation interface modules 35 kHz . |
| 10 | One bidirectional counter for position sensors (pulse multiplier x4) | A = Trace A of the position sensor B = Trace B of the position sensor | This operating mode equals operating mode 7 with one exception: There is a pulse multiplication $x 4$ with the evaluation of the counting inputs. This means that the counter counts the positive and negative edges of the traces A and B. This results in the fourfold number of counting pulses. The precision increases correspondingly. <br> In this operating mode, the maximum counting frequency is: <br> I/O modules 15 kHz . <br> Communucation interface modules 20 kHz . |

## Configuration for onboard I/Os

## Configuring the fast counter

1. In the device tree, double-click the "Onboard I/O" node (OBIO).
2. In the "Parameters" tab set the counting mode for the fast counter.
\# Chapter 6.3.2.13.9.2.2 "Counting modes" on page 1582

| StartPage [] PLC_AC500_V3 [fis OnBoard_IO $\times$ |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 12DI/8D0-T/2DC Parameters | Parameter$\qquad$ | Type | Value | Default Value | Unit | Description |
| 12DI/8DO-T/2DC I/0 Mapping |  | Enumeration of BYTE | No | No |  | Start PLC pro |
|  | - Digital inputs 24 VDC |  |  |  |  |  |
| 12DI/8D0-T/2DC IEC Objects | - Input 0, input delay | Enumeration of BYTE | 8 ms | 8 ms |  | Fast input 0 - |
|  | - Input 0, channel configuration | Enumeration of BYTE | Input/Interrupt | Input/Interrupt |  | Fast input 0 - |
| I/O mapping list | - Input 1, input delay | Enumeration of BYTE | 8 ms | 8 ms |  | Fast input 1 - |
|  | - Input 1, channel configuration | Enumeration of BYTE | Input/Interrupt | Input/Interrupt |  | Fast input 1 - |
|  | - Input 2, input delay | Enumeration of BYTE | 8 ms | 8 ms |  | Fast input 2 - |
|  | - Input 2, channel configuration | Enumeration of BYTE | Input/Interrupt | Input/Interrupt |  | Fast input 2 - |
|  | - Input 3, input delay | Enumeration of BYTE | 8 ms | 8 ms |  | Fast input 3 - |
|  | - Input 3, channel configuration | Enumeration of BYTE | Input/Interrupt | Input/Interrupt |  | Fast input 3 - |
|  | - Input 4, input delay | Enumeration of BYTE | 8 ms | 8 ms |  | Fast input 4- |
|  | - Input 4, channel configuration | Enumeration of BYTE | Forward Counter | Input |  | Fast input 4- |
|  | - Input 5, input delay | Enumeration of BYTE |  | 8 ms |  | Fast input 5 - |
|  | - Input 5, channel configuration | Enumeration of BYTE | rack-A |  |  | Fast input 5 - |
|  | - Input 6, input delay | Enumeration of BYTE |  | vard Counter |  | Fast input 6 - |
|  | - Input 6, channel configuration | Enumeration of BYTE | Input | inpur |  | Fast input 6- |

3. In the "//O Mapping" tab channel configuration is displayed.

乡y Chapter 6.3.2.13.9.3 "Control of the fast counter" on page 1583

The parameter of the fast counter channels of the Onboard I/O must be configured before they can be used. User should take these steps to configure the fast counter:

| Channel | Direction | Width | Description |
| :--- | :--- | :--- | :--- |
| Actual <br> value X | Input | DWORD | Current value of the fast counter. |
| State byte <br> $X$ | Input | BYTE | Bit $0=$ CF <br> If the counter reaches the programmed end value, the <br> counter output is stored permanently as CF = TRUE <br> (end value reached). Only, if the counter is set again <br> (set value), CF is reset to FALSE. <br> Bit 1 to Bit 7 free |
| Start value <br> $X$ | Output | DWORD | Each counter can be set to a start value. Start values <br> are loaded into the counter by the user program. Using <br> the set signal (dependent on the operating mode either <br> via a terminal or the bit SET within the control byte X), <br> the values of the double word variables are loaded into <br> the counter X. |
| End value <br> $X$ | Output | DWORD | The end values for the two counters are stored as <br> comparison values into the module by the user pro- <br> gram. Both counters compare continuously, whether <br> or not their programmed end value is equal to their <br> actual value. If the counter (actual value) reaches its <br> programmed end value, the binary output CF of the <br> status byte is set permanently. |


| Channel | Direction | Width | Description |
| :---: | :---: | :---: | :---: |
| Control byte 1 | Output | BYTE | Bit $0=$ UP/DWN <br> In some operating modes, the counter can count downwards, too. If counting down is desired, the bit UP/DWN must be set to TRUE. When doing so, the counter starts counting downwards at the start value (set value) to the end value (max. from 4,294,967,295 to 0 or hexadecimal from FF FF FF FF to 000000 00). After reaching 0 , the counter jumps to 4,294,967,295. <br> Bit 1 = EN <br> The processing of the counter signals must be enabled. Depending on the operating mode, enabling is done via a terminal or by the bit EN = TRUE within the control byte. <br> Bit 2 = SET <br> The counter can be set to a start value (see the description of the set values for the counters 1 and 2 at the beginning of this table). <br> Bit 3 = CF_HW <br> $0=$ state of CF is set to hardware channel (only for mode 1 and 2) <br> 1 = normal output is set to hardware channel <br> Bit 3 is evaluated only in control byte of counter 1. <br> Bit 4 to Bit 7 free |
| Control byte 2 | Output | BYTE | Bit $0=$ UP/DWN <br> In some operating modes, the counter can count downwards, too. If counting down is desired, the bit UP/DWN must be set to TRUE. When doing so, the counter starts counting downwards at the start value (set value) to the end value (max. from 4,294,967,295 to 0 or hexadecimal from FF FF FF FF to 000000 00). After reaching 0 , the counter jumps to 4,294,967,295. <br> Bit $1=\mathrm{EN}$ <br> The processing of the counter signals must be enabled. Depending on the operating mode, enabling is done via a terminal or by the bit EN = TRUE within the control byte. <br> Bit 2 = SET <br> The counter can be set to a start value (see the description of the set values for the counters 1 and 2 at the beginning of this table). <br> Bit 3 to Bit 7 free |

## Counting modes

The fast counter can be configured as one mode out of 10 possible modes. The desired operating mode is selected in the PLC configuration using configuration parameters. Inputs and outputs which are not used by the counter are available for other tasks. In the following table, A means input channel $\mathrm{A}, \mathrm{B}$ means input channel B and C means output channel C .

| CPUs | Integrated <br> fast counter | Assigned inputs |  | Assigned <br> Outputs | Remarks |
| :--- | :--- | :--- | :--- | :--- | :--- |
|  |  | Channel A | Channel B | Channel C |  |

Only "Forward Counter" can be configured as fast counter mode for onboard IOs on eCo V3 PLCs.

| Parameter | PM5012-x- <br> ETH | PM5032-x- <br> ETH | PM5052-x- <br> ETH | PM5072- <br> T-2ETH(W) |
| :--- | :--- | :--- | :--- | :--- |
| Onboard digital inputs | PM5082- <br> T-2ETH |  |  |  |
| Channels | 6 <br> (incl. 2 fast <br> counter inputs <br> 5 kHz and 4 <br> interrupts) | 12 <br> (incl. 4 fast counter/encoder inputs (100 kHz/200 kHz), <br> counter inputs (5 kHz), 4 standard inputs) |  |  |
| Onboard digital outputs |  |  |  |  |
| Channels for <br> transistor version | 4 <br> 5 kHz <br> standard and <br> PWM) | (incl. 4 fast outputs or 4 PWM/2PTO (100 kHz/200 kHz), 4 <br> standard outputs (5 kHz)) |  |  |

* Chapter 5.2.1.1.2 "Onboard I/Os in processor module PM50x2" on page 196


## Control of the fast counter

To control the fast counter configuration open the "//O Mapping" tab.
The channels can be mapped as described in Symbolic Names for Variables, Inputs and Outputs and have the following meaning ${ }^{\text {nch }}$ Chapter 6.3.2.13.7 "Symbolic names for variables, inputs and outputs" on page 1572:

| $\square$ Fast counter |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: |
| ${ }^{4}$ | Actual value 1 | \%ID17 | DWORD | Actual value 1 |
| 4 | Actual value 2 | \%ID18 | DWORD | Actual value 2 |
| ${ }^{4}$ | State byte 1 | \%IB76 | BYTE | State byte 1 |
| ${ }^{4}$ | State byte 2 | \%IB77 | BYTE | State byte 2 |
| ${ }^{6}$ | Start value 1 | \%QD11 | DWORD | Start value 1 |
| ${ }^{6}$ | End value 1 | \%QD12 | DWORD | End value 1 |
| ${ }^{6}$ | Start value 2 | \%QD13 | DWORD | Start value 2 |
| ${ }^{6}$ | End value 2 | \%QD14 | DWORD | End value 2 |
| ${ }^{6}$ | Control byte 1 | \%QB60 | BYTE | Control byte 1 |
| * | Control byte 2 | \%QB61 | BYTE | Control byte 2 |


| Channel | Direction | Width | Description |
| :--- | :--- | :--- | :--- |
| Actual value X | Input | DWORD | Current value of the fast counter |
| State byte $X$ | Input | BYTE | Bit $0=$ CF <br> If the counter reaches the pro- <br> grammed end value, the counter <br> output is stored permanently as CF <br> = TRUE (end value reached. Only if <br> the counter is set again (set value), <br> CF is reset to FALSE. |
| Start value $X$ | Output | DWORD | Bach counter can be set to a start <br> value. Start values are loaded into <br> the counter by the user program. <br> Using the set signal (dependent on <br> the operating mode either via a ter- <br> minal or the bit SET within the contro <br> byte $X$ ), the values of the double word <br> variables are loaded into the counter <br> X. |
| End value $X$ | Output | DWORD | The end values for the 2 counters <br> are stored as comparison values into <br> the module by the user program. <br> Both counters compare continuously <br> whether or not their programmed end <br> value is equal to their actual value. <br> When the counter (actual value) rea- <br> ches its programmed end value, the <br> binary output CF of the status byte is <br> set permanently. |


| Channel | Direction | Width | Description |
| :---: | :---: | :---: | :---: |
| Control byte 1 | Output | BYTE | Bit $0=$ UP/DWN |
|  |  |  | In some operating modes, the counter can count downwards, too. If counting down is desired, the bit UP/DWN must be set to TRUE. If doing so, the counter starts counting downwards at the start value (set value) to the end value (max. from 4,294,967,295 to 0 or hexadecimal from FF FF FF FF to 000000 00). After reaching 0 the |
|  |  |  | Bit $1=\mathrm{EN}$ |
|  |  |  | The processing of the counter signals must be enabled. Depending on the operating mode, enabling is done via a terminal or by the bit EN = TRUE within the control byte. |
|  |  |  | Bit $2=$ SET |
|  |  |  | The counter can be set to a start value (see the description of the set values for the counters 1 and 2 at the beginning of this table. $\mathrm{CF}=0$ <br> Bit $3=$ CF HW |
|  |  |  | $0=$ state of CF is set to hardware channel (only for mode 1 and 2) |
|  |  |  | 1 = normal output is set to hardware channel |
|  |  |  | Bit 3 is evaluated only in control byte of counter 1 . |
|  |  |  | Bit 4 to Bit 7 free |
| Control byte 2 | Output | BYTE | Bit $0=$ UP/DWN |
|  |  |  | In some operating modes, the counter can count downwards, too. If counting |
|  |  |  | down is desired, the bit UP/DWN must be set to TRUE. If doing so, the |
|  |  |  | counter starts counting downwards at the start value (set value) to the end |
|  |  |  | value (max. from 4,294,967,295 to 0 or hexadecimal from FF FF FF FF to |
|  |  |  | 00000000 ). After reaching 0 the counter jumps to 4,294,967,295. |
|  |  |  |  |
|  |  |  | Bit $1=\mathrm{EN}$ |
|  |  |  | The processing of the counter signals must be enabled. Depending on the operating mode, enabling is done via a terminal or by the bit EN = TRUE within the control byte. |


| Channel | Direction | Width | Description |
| :--- | :--- | :--- | :--- |
|  |  |  | Bit $2=$ SET <br> The counter can be set to a start <br> value (see the description of the set <br> values for the counters 1 and 2 at the <br> beginning of this table. |
|  |  |  | Bit 3 to Bit 7 free |

### 6.3.2.14 Serial interface

### 6.3.2.14.1 Setting up a serial interface

## General



The configuration for serial interfaces and their protocols is done via two nodes:

- One node represents the common serial parameters related to the hardware port.
- The node below represents the parameterization for the particularly attached protocol.


## Supported protocols

- 3S CAA SerialCom (common serial communication, send/receive data)
* $\boldsymbol{y}$ Chapter 6.3.2.14.3 "Configuring CAA SerialCom on serial interface" on page 1590
- Modbus RTU (client \& server)
$\boldsymbol{y}$ Chapter 6.3.2.14.2 "Configuring Modbus RTU on serial interface" on page 1588
\& How to switch between the protocols.


## Configuration

The following parameters are available in the configuration view of the COM port node:

| Parameter | Value ranges | Description |
| :---: | :---: | :---: |
| Run on config fault | No | If this parameter is set to "Yes" the IEC application will not be prevented from switching to RUN state, independent from possibly existing configuration errors of the particular COM port. |
|  | Yes |  |
| Transmission rate | 9600 baud/sec | Sets up the transmission rate to use for the COM port. |
|  | 19200 |  |
|  | 38400 |  |
|  | 57600 |  |
|  | 115200 |  |
| Parity | None | Sets up the parity to use for the COM port. |
|  | Odd |  |
|  | Even |  |
| Data bits | 5 data bits | Sets up the number of data bits to use for the COM port |
|  | 6 data bits |  |
|  | 7 data bits |  |


| Parameter | Value ranges | Description |
| :--- | :--- | :--- |
|  | 8 data bits |  |
| Stop bits | 1 stop bits | Sets up the number of stop bits to use for the <br> COM port. |
|  | 2 stop bits | No flow control |
|  | Hardware RTS/CTS | Allows to switch between different flow control <br> modes (either RTS/CTS hardware or Xon/Xoff <br> software or none). This setting is only valid for <br> RS-232 serial interface mode. In case RS-485 <br> is used for parameter "Serial interface", flow <br> control must set to "No flow control". Otherwise <br> a configuration error is triggered. |
|  | Software Xon/Xoff | Allows to switch between RS-232 and RS-485. <br> Due to technical reasons, it's not possible to <br> dynamically switch between the modes. This <br> means, a reboot (or power cycle) of the PLC is <br> required to activate the particular setting once <br> changed. |
| Boot parameter <br> Serial interface | RS-232 | RS-485 |

## Comparison to AC500 V2 PLCs

The following table shows the differences between AC500 V2 and V3 PLCs regarding the parameter set for serial interfaces:

| V2 Parameter | Representation in V3 | Remark |
| :--- | :--- | :--- |
| Run on config fault | Run on config fault | Exactly the same |
| RTS control | Flow control (partially) | Special modes which allow to use PLCC as <br> modem and mode implicitly setting RS-485 will <br> not be taken over. Flow control settings will be <br> limited and only support hardware, software or <br> none. |
| Transmission rate | Transmission rate | For V3, the transmission rate range will be 9600 <br> to 115200. Low modes will not be supported <br> due to lack of support in Linux. High rates were <br> only realized in V2 to support field bus plug as <br> well as CS33 field bus. Both protocols are not <br> supported anymore in V3, so these transmission <br> rates won't be available in V3. Approach: Only <br> support most common transmission rates |
| Parity | Parity (subset) | AC500 V3 doesn't allow to configure parity <br> modes "mark" and "space". This means, only <br> "none", "odd" and "even" are configurable. |
| Data Bits | Data Bits | Exactly the same |
| Stop Bits | Stop Bits | Exactly the same |

### 6.3.2.14.2 Configuring Modbus RTU on serial interface

## Enable Modbus RTU

To enable Modbus RTU on a serial interface the protocol setup per default has to be replaced by either Modbus RTU client or server, depending on required operation mode.
A serial interface supports only one protocol/operation mode at once.


1. Right-click node "CAA_SerialCom" and click "Add object".
2. Select "Modbus RTU Client" or "Modbus RTU Server" and click "Add object".
$\Rightarrow$ "CAA_SerialCom" is replaced by your selection.

## Parameters

Serial Serial parameters to be set selecting the interfaces node "COM_1". They are common for both operating modes client and server.


Modbus RTU server

Server specific parameters to be set selecting the protocol's node "Modbus_RTU_Server".


Bus address of the PLC as Modbus RTU Server on that interface
"Byte Order"

Format/Endianess for the transmission of WORD values (register) within the request/response telegram (default: Big Endian)
"Disable"

| Parameter | Default | Value | Description |
| :--- | :--- | :--- | :--- |
| Disable write to \%MB from | 0 | $0 \ldots 65535$ | Disable write access starting at <br> $\% \mathrm{MBx}$ |
| Disable write to \%MB to | 0 | $0 \ldots 65535$ | Disable write access up to \%MBx |
| Disable read from \%MB <br> from | 0 | $0 \ldots 65535$ | Disable read access starting at <br> $\% M B x$ |
| Disable read from \%MBx <br> to | 0 | $0 \ldots 65535$ | Disable read access up at \%MBx |

It is possible to disable read and/or write access to individual segments. Reading/writing is disabled beginning at the set start address and is valid up to the set end address (inclusive).

Modbus RTU "Modbus RTU Client" does not have any protocol parameters.
client

### 6.3.2.14.3 Configuring CAA SerialCom on serial interface

## Enable CAA_SerialCom

The "CAA SerialCom" protocol is the serial standard protocol from CODESYS. It allows users to implement a custom protocol.

How to implement a serial communication for ASCII code via RS232 or RS485 within a AC500 V3 or AC500-eCo V3 PLC is described in an application example. The described implementation is based on the AC500_Com library which implements the CAA SerialCom library to establish a connection via the COM interfaces.

## Default setting



When creating a new project, the protocol "CAA SerialCom" is automatically attached to the "COM_1" port of a V3 PLC.

## Switch to default setting


$\triangleright$ Right-click on the node attached to "COM_1" node in the device tree and click "Delete". $\Rightarrow$ The node is switched back to the "CAA SerialCom" protocol.

## Parameters



Since "CAA SerialCom" doesn't represent a "real" protocol, there are no specific parameters required. All common settings can be found at the "COM_1" tab after double-click on the "COM_1" node.

* Chapter 6.3.2.14.1 "Setting up a serial interface" on page 1587


## Activate particular configuration parameters

The parameters set up in the Automation Builder device tree are NOT automatically taken over in the PLC.

It is still required to use the "3S IEC POUs" to activate the particular configuration parameters.


ABB provides the library AC500 Com ("ABB - AC500 / Use Cases / Serial Communication)" which contains a POU called "ComGetCaaSerialComConfig".
The function block can be used to obtain the configuration data which is set up in Automation Builder to directly pass it to "CAA SerialCom-POU Open". This avoids manual creation of a parameter list.
The following code snippet shows how the COM port is identified by its node name and how the parameter list for the function block is read from the configuration data of the currently loaded IEC application:

```
FUNCTION_BLOCK GET_CAA_COM_CFG
VAR_INPUT
END VAR
VAR_OUTPUT
END_VAR
VAR
```

ComGetCaaSerialComConfig:
bExecGetCfg:
bDoneGetCfg:
bErrorGetCfg:
bBusyGetCfg:
ErrorIdGetCfg:
AC500_Com.ERROR_ID.NO_ERROR;
asParamList: ARRAY[0..31] OF
AC500_Com. Serial Communication. COM. PARAMETER;
uiNumParams: UINT := 32;
uiStep:
szNodeName:
ComID:
bSuccess:
bError:
END_VAR

VAR CONSTANT

ComGetCaaSerialComConfig;
BOOL := FALSE;
BOOL := FALSE;
BOOL := FALSE;
BOOL := FALSE;
AC500_Com.ERROR_ID :=

UINT : = 0 ;
STRING (80) $:=$ 'COM1'; AC500_Com.COM_PORT_ID;
BOOL : = $\overline{\mathrm{F} A L S E ; ~}$
BOOL := FALSE;

```
    STEP_INIT:
        STEP_GET_ID:
        STEP FAI\overline{LED GET ID:}
        STEP_GET_CF\overline{G}_CA\overline{A}:
        STEP_FAILED_\overline{GET_CFG_CAA:}
        STEP_DONE_SUCCCESS:
END_VAR
IF uiStep = STEP GET ID THEN
    ComId := ComGetIdByName(szNodeName);
    IF ComId = AC500_Com.COM_PORT.COM_ID_INVALID THEN
                        uiStep }\overline{:}=\mp@subsup{S}{STEP_\overline{FAILED_GE\overline{T}}I\overline{D};}{
    ELSE
                bExecGetCfg := TRUE;
                    uiStep := STEP_GET_CFG_CAA;
    END_IF
END_IF
IF uiStep = STEP_GET_CFG_CAA THEN
    ComGetCa\overline{a}SerīalComConfig(
                            Execute:= bExecGetCfg,
                    Done=> bDoneGetCfg,
                    Busy=> bBusyGetCfg,
                    Error=> bErrorGetCfg,
                    ComID:= ComID,
                    pCaaParamList:= ADR(asParamList[0]),
                    NumParams:= uiNumParams,
                    ErrorID=> ErrorIdGetCfg);
        IF bDoneGetCfg THEN
                    uiStep := STEP DONE_SUCCESS;
    ELSIF bErrorGetCfg THEN
                            uiStep := STEP_FAILED_GET_CFG_CAA;
    END_IF
END_IF
IF uiStep = STEP_DONE_SUCCESS THEN
        bSucces的 := \overline{TRUE;}
END_IF
IF uiStep = STEP_FAILED_GET_ID THEN
    bError := TRUE;
END_IF
IF uiStep = STEP_FAILED_GET_CFG_CAA THEN
    bError := TRUE;
END_IF
```

Gateway configuration

1. In the Automation Builder project, right-click the topmost PLC tree node and select "Communication Settings".
$\Rightarrow$ The dialog window "Communication Settings" appears.
2. Click "Advanced Settings" to open the dialog ${ }^{\wedge}>$ Chapter 6.4.1.21.2.8.3 "Tab 'Communication Settings"' on page 2427.
$\Rightarrow$ This information will be stored in the project file.
3. Click "Gateway" and select the desired action from the "Gateway" menu either to change the local gateway ${ }^{〔}$ Chapter 6.4.1.21.3.19.2 "Command 'Configure the Local Gateway"' on page 2721 or to add a new gateway channel $\Leftrightarrow$ Chapter 6.4.1.21.3.19.1 "Command 'Add New Gateway'" on page 2720.

Confirm your settings with [OK].

### 6.3.2.15.1 Gateway settings on windows server

| Gateway as a $\quad$To allow multiple concurrent users from different user sessions on the server to connect to <br> service | PLCs, the user has to run CODESYS gateway as a system service. This is managed by a <br> service called "CoDeSys V2.3 Gateway Service Wrapper". The service starts on system start-up <br> and launch the gateway. |
| :--- | :--- | and launch the gateway.

If you want to restart the gateway, use "Services management console" to restart "CoDeSys V2.3 Gateway Service Wrapper".
This executable is installed only on a windows server system and a related service (CoDeSys V2.3 Gateway Service Wrapper) is created and started.
For V3 PLC the V3 gateway always runs as a service on windows servers and also on other Windows OS variants (e.g. Windows 10).

Gateway set- You can set the communication settings in the Automation Builder project for every PLC. Othertings wise, an error message is displayed while trying to open CODESYS.

See the description for Gateway configuration and select "TCP/IP" under "Connection" " Chapter 6.3.2.15 "Gateway configuration" on page 1593.

Windows server usage in Automation Builder requires modifications of the system environment and of the settings in Automation Builder, e.g. permissions, licensing, session handling etc. How to setup system and software for Windows server is described in the application note Windows Server 2016 - RDM access.
6.3.2.16 CAN onboard
6.3.2.16.1 CANopen

General
In Automation Builder, a CANopen network consists of one CANopen manager which acts as master device and optional CANopen remote devices which act as slave devices.

## CANopen manager (master)

## General

## Tab 'CANopen Manager - General'

Table 305: "General"

| "Node-ID" | The node number identifies the CANopen Manager as unique (range of values: <br> $1 \ldots .127)$. |
| :--- | :--- |
| "Check and Fix Configuration" | Opens the dialog of the same name. See below for details. |


| "Autostart CANopen Manager" | The CANopen Manager starts automatically (switches to OPERATIONAL mode) after all required slaves are ready. $\square$ The CANopen Manager has to be started from the application. The function block CiA405. NMT can be used to do this. <br> Hint: As long as the CANopen Manager is not in OPERATIONAL mode, no PDOs are sent (outputs refreshed). |
| :---: | :---: |
| "Polling of optional slaves" | When a slave does not respond during the boot sequence, the CANopen Manager interrogates it every second until it does respond. <br> Constantly polling the slave increases the bus cycle time, which can interfere with the application (especially motion applications). You can deactivate polling to avoid this behavior. If polling is deactivated, then a slave is detected again when it sends a bootup message. |
| "Start slaves" | The CANopen Manager is responsible for starting the slaves. <br> : You have to start the slaves from the application. Use the CiA405 NMT function block to do this. |
| "NMT start all (if possible)" | If the "Start slaves" option is activated, then the CANopen Manager starts all slaves with an "NMT Start All" command. The "NMT Start All" command is not executed as long as optional slaves are not yet ready to be started. In this case, the CANopen Manager starts each slave individually. The "NMT Start All" command can be guaranteed only in a project without optional slaves. |
| "NMT error behavior" | - "Restart slave". If an error occurs during slave monitoring (NMT Error Event), then the slave is restarted automatically by the stack (NMT Reset + SDO Configuration + NMT Start). <br> - "Stop slave". If an error occurs during slave monitoring (NMT Error Event), then the slave is stopped. Then you have to reset the slave from the application, using the CiA405 NMT function block. |

Table 306: "Guarding"
Working with heartbeat messages is an alternative method of monitoring. It can be executed from both master and slave nodes, as opposed to node guarding. Normally the master sends heartbeat messages to the slaves.

| "Enable heartbeat producing" | The master sends heartbeats. They define the time interval in the "Producer <br> time". When the slaves are provided with the heartbeat function, a heartbeat <br> consuming entry from the slave is created for the master. Then the Node-ID and <br> the 1.5x heartbeat interval of the master are applied. <br> $\square: ~ N o d e ~ g u a r d i n g ~ i s ~ e n a b l e d ~ f o r ~ t h e ~ s l a v e s . ~ T h e ~ s e t t i n g s ~ f r o m ~ t h e ~ E D S ~ f i l e ~ o f ~$ |
| :--- | :--- |
| the slaves are used for this. If the values there cannot be used, then default |  |
| values are used. Note that a CANopen Slave device can also be configured as a |  |
| heartbeat producer. |  |$|$| Unique identification (1-127) of the heartbeat producer on the bus |  |
| :--- | :--- |
| "Producer time (ms)" | Interval length between successive heartbeats (in milliseconds) |
| "Redundancy Node-ID" | Requirement: A "Redundancy Configuration" object is inserted below the appli- <br> cation. <br> Unique identification (1-127) of the redundant heartbeat producer on the bus |
| "Redundancy wait time ( $\mu \mathrm{s}) "$ | Requirement: A "Redundancy Configuration" object is inserted below the appli- <br> cation. <br> Duration of how long the passive controller waits for the heartbeat of the active <br> controller. If this time is exceeded, then the passive controller takes on the active <br> role. |

Table 307: "SYNC"

| "Enable SYNC producing" | $\boxed{\checkmark}$ : The CANopen manager sends SYNC telegrams (disabled by default) <br> The synchronous PDOs are sent directly after the SYNC telegram. |
| :--- | :--- |
| "COB-ID (Hex)" | CAN-ID of the SYNC telegram. Range of possible values: [1...2047]. |
| "Cycle period ( $\mu \mathrm{s}$ )" | Interval length (in microseconds) after which the SYNC telegram is sent |
| "Window length ( $\mu \mathrm{s}$ )" | Length of the time frame for synchronous PDOs (in microseconds) |
| "Enable SYNC consuming" | $\checkmark:($ disabled by default). Another device must produce the SYNC telegrams that <br> are received by the CANopen Manager. |

## - NOTICE!

```
If SYNC producing is enabled for the CANopen manager, then you are not permitted to select the "Enable SYNC producing" option for all other bus devices.
```

Table 308: "TIME"

| "Enable TIME producing" | $\boxed{\square}$ : (disabled by default). The CANopen Manager sends TIME messages. |
| :--- | :--- |
| "COB-ID (Hex)" | (Communication Object Identifier): identifies the time stamp of the message. <br> Default values: [0...2047], preset $16 \# 100$ |
| "Producer time (ms)": | Interval (in milliseconds) when the time stamp is sent. This value has to be a <br> multiple of the task cycle time. Possible values [0, 65535] |



See also

- \& Chapter 6.3.2.16 "CAN onboard" on page 1594
- \& Chapter 6.3.2.16.1.3.2 "Tab 'CANopen Remote Device - General"" on page 1597

Dialog 'Check If you insert several devices below the CANopen manager, then error messages may report and Fix Configu- multiple assigned Node-IDs or invalid COB-IDs. The "Check and Fix Configuration" button ration' opens a dialog for solving these conflicts.
For conflicts with Node-IDs or PDO COB-IDs, you can click "Edit Conflicts" to open a dialog with detailed information.

Table 309: "Node-ID and COB-ID conflicts"

| "Doubled node number" | List of all devices with identical IDs. In the field of the "Node-ID" column, you can <br> enter new node numbers for the affected devices. |
| :--- | :--- |
| "Incorrect and double | The COB-IDs that are generated automatically from the device description files <br> assignment of PDO COB-IDs" <br> may not be permitted. All incorrect entries are listed with the respective device <br> names, Node-IDs, and indexes. There are three options for correcting invalid <br> COB-IDs: <br> - Correct the displayed formula for calculating the COB-IDs so that a valid <br> COB-ID results. You can change the formula in the respective table element. <br> - Accept the automatic suggestion for the COB-ID by clicking the respective <br> button. <br> Accept all automatic suggestions by clicking the "Use Suggested COB-ID" <br> button. |

Corrected entries are removed from the displayed list automatically.

You can solve timing problems automatically by using the "Automatic Repair". The command modifies all timing values to compatible values. (The time should be a multiple of the task time.)

## CANopen remote device (slave)

## General

In CODESYS, a CANopen Remote Device is a slave device that you insert below a CANopen Manager in the device tree of a project. A distinction is made between modular and non-modular slaves:

- Modular slaves: You can insert CANopen modules (submodules) below a modular slave. These modules provide a "//O Mapping" tab to map their inputs and outputs. Modular slaves can also have fixed I/Os. Then these devices also provide the "I/O Mapping" tab. Modular devices provide the "Configure PDO mapping automatically" option, which we recommend for standard applications. You find this option in the "CANopen Remote Device" dialog, on the "General" tab.
- Non-modular slaves: You cannot insert additional modules below a non-modular device. The inputs and outputs of these devices are mapped in the "//O Mapping" dialog. Automatic mapping is not possible here.
See also
- ${ }^{2}$ Chapter 6.3.2.16.1.3.2 "Tab 'CANopen Remote Device - General"" on page 1597


## Tab 'CANopen Remote Device - General'

The general settings of the CANopen Slave are defined in this dialog of a CANopen Remote Device (slave).

Table 310: "General"

| "Node-ID" | The node number identifies the CANopen Remote Device uniquely. It corre- <br> sponds to the number (value between 1 and 127) set on the device (hardware). <br> You have to provide the Node-ID as a decimal. |
| :--- | :--- |
| "Expert settings" | V: All settings are displayed that are predefined by the device description (EDS <br> file) for the device. |
| "SDO channels (...)" | Click this button to open a dialog for activating the SDO channels that are prede- <br> fined in the EDS file. Service data objects (SDOs) allow access to all entries in <br> the CANopen object directory. An SDO creates a peer-to-peer communication <br> channel between two devices (SDO server and client channel). |
| "Optional device" | : The slave is optional and not required for starting the CAN network. |
| "Sync producing" | Available only when the "Enable sync producing" option is cleared in the <br> CANopen Manager. <br> 目 The I/O transmission is synchronized on the bus. The slave works as a sync <br> producer. The parameters of the sync interval are defined in the settings of the <br> CANopen Manager. |
| "No initialization" | This option is for non-configurable slave that already start with a valid configura- <br> tion. |
| V: The master does not send configuration SDOs or NMT start commands to <br> the slave. PDO communication and monitoring (heartbeat, node guarding) are <br> performed when this has been configured in the configurator. <br> If the slave does not start automatically, then the user can use the CiA405 NMT <br> function block to send an NMT start command to the slave. |  |


| "Default settings" | The availability of this option depends on the contents of the device description file. <br> Activated by default. The slave nodes are reset to the default parameters before the configuration is loaded to the device or always when the slave is configured. Which parameters can be set is device-specific. The concrete task is performed from the subindex of the list box. <br> - "Sub:001": All parameters are reset. <br> - "Sub:002": Communication parameters (index 1000h - 1FFFh manufacturerspecific communication parameters) are reset. <br> - "Sub:003": Application parameters (index 6000h - 9FFFh manufacturer-specific application parameters) are reset. <br> - "Sub:004" - "Sub:127": Manufacturer-specific, individual selection of parameters is reset. <br> - "Sub:128" - "Sub254": Reserved for future purposes |
| :---: | :---: |
| "Autoconfig PDO mapping". | This option is available for modular devices only. <br> $\checkmark$ PDO mapping is generated automatically from the definitions in the device description and then cannot be changed in the two mapping dialogs. If the automatically generated mapping does not match your application, then you can deactivate the option and configure the mapping manually. We recommend that this option is activated for standard applications. |

Table 311: "Node Guarding"
Node guarding is an outdated monitoring method and should not be used anymore because is uses RTR frames. You should always use heartbeats whenever possible. In some exceptions, such as for older slaves, you can use only node guarding.

| "Enable node guarding" | V: The CANopen Manager sends a message to the slave in the "Guard time <br> (ms)" interval. If the slave does not respond with the given "Guard COB-ID" <br> (Communication Object Identifier), then the CANopen Manager resends this <br> message as many times as defined in "Lifetime factor" or until the slave <br> responds. <br> If the slave does not respond, then it is marked as "unavailable". |
| :--- | :--- |
| "Guard time (ms)" | Interval for sending messages (default: 200 ms) |$|$| "Lifetime factor" | When the slave does not respond, a node-guarding error is established <br> according to the "Lifetime factor" time multiplied by the "Guard time". |
| :--- | :--- |
| "Enable heartbeat producing" | V: The module sends heartbeats in the time intervals as given in "Producer time <br> (ms)". |
| "Producer time (ms)": | The default setting is 200 as long as there is no special entry or the entry in the <br> device description file is 0. |
| "Heartbeat consuming (...)" | Opens a "Heartbeat Consuming Properties" dialog. There you activate the slaves <br> that you want to watch. <br> The number of possible slaves to be monitored is defined in the EDS file. To do <br> this, you must select the "Enable" check box and enter the Node-ID of the slave <br> and the required values in the "Heartbeat time" field (in milliseconds). Then the <br> slave monitors the heartbeats that are sent from the affected slaves (defined by <br> the Node-ID). When no more heartbeats are received, the slave switches off the <br> I/Os. <br> When a slave is monitoring, a green check mark is displayed on the "Heartbeat <br> Consuming" button. <br> Note: When you insert a device with the heartbeat function, its heartbeat settings <br> are harmonized automatically with the master (CANopen Manager). |

Table 312: "Emergency"

| "Activate Emergency" | $\checkmark:$ When internal errors occur, the slave sends emergency messages with a <br> unique COB-ID. You can read these messages by using the function blocks from <br> the library CAA Can Low Level Extern (RECV_EMCY_DEF, RECV_EMCY). |
| :--- | :--- |
| "COB-ID" | CAN ID of the EMCY message. Range of possible values: [1...2047]. |

Table 313: "TIME"

| The availability of this function depends on the device description. |  |
| :--- | :--- |
| "Enable TIME producing" | $\boxed{\square}:$ The device sends TIME messages. |
| "COB-ID (Hex)" | (Communication Object Identifier): identifies the time stamp of the message. |
| "Enable TIME consuming" | $\boxed{ } /$ : The device processes TIME messages. |

Table 314: "Checks at Startup"
The respective information is read from the firmware of the CANopen Slave ( $0 \times 1018$ identity object) and compared to the information from the EDS file. In case of disparities, the configuration is stopped and the slaves are not started.

| "Vendor ID" | $\boxed{\square}$ : Check of the vendor ID at startup |
| :--- | :--- |
| "Product number" | $\boxed{\square}:$ Check of the product number at startup |
| "Revision number" | $\boxed{\text { : }}$ : Check of the revision number at startup |

See also

- $\boldsymbol{y}^{*}$ Chapter 6.3.2.16.1.2.2 "Tab 'CANopen Manager - General"' on page 1594
- \#y Chapter 6.3.2.16 "CAN onboard" on page 1594


## Tab 'CANopen Device - PDOs'

Object: CANopen Remote Device, CANopen Local Device
This dialog is available only in the device editor of a CANopen Slave of version V3.5.6.0 or higher. It shows all PDOs and their default settings. In this dialog, you can add new objects and delete or edit existing objects.
On the left side, there are the PDOs that the slave receives from the master. On the right side, there are the PDOs that the slave sends to the master.

| "Add PDO" | Opens the "Select PDO" dialog where all available PDOs are displayed. In this <br> dialog, you select the PDOs to be added to "Receive PDOs" or "Transmit PDOs". |
| :--- | :--- |
| "Add Mapping" | Opens the "Select Item from Object Directory" dialog. Objects are listed there <br> that you can add to the PDO mapping. |
| "Edit" | When a PDO is selected, the "PDO Properties" dialog opens. <br> When a PDO mapping is selected, the "Select Item from Object Directory" dialog <br> opens. |
| "Delete" | Deletes the selected objects from the list |
| "Move Up" | Moves the selected object upwards by one line. |
| "Move Down" | Moves the selected object downwards by one line. |

See also

- \& Chapter 6.3.2.16 "CAN onboard" on page 1594

Dialog 'PDO
Properties'

| "COB-ID" | Every PDO message must have a COB-ID (Communication Object Identi- <br> fier). You can input explicit values (example: 16\#201) or formulas (example: <br> \$NODEID+16\#200). |
| :--- | :--- |
| "RTR" | Remote Transmission Request. This option is available for transmit PDOs only. <br> V You can use an RTR frame for interrogating the PDO externally. |
| "Inhibit time $(x 100 \mu s) "$ | You can edit this field only if the device supports this functionality. The inhibit <br> time is the minimum time between two messages of a specific PDO. You can <br> use this setting for preventing PDOs from being sent too often when their values <br> are edited. Default: "0". Possible values: 0-65535. |
| "Transmission type""Acyclic - synchronous": When a change is made, the PDO is transmitted <br> synchronously, but not periodically. (default) |  |
| "Cyclic - synchronous": The PDO is transmitted every nth sync. |  |
| - "Synchronous - only RTR": Available for transmit PDOs only. After a syn- |  |
| chronization message, the PDO is updated, but not transmitted. Transmis- |  |
| sion is by explicit request only (Remote Transmission Request). |  |
| "Asynchronous - only RTR": Available for transmit PDOs only. The PDO |  |
| is updated and transmitted by explicit request only (Remote Transmission |  |
| Request). |  |
| "Asynchronous - manufacturer specific": The PDO is transmitted only after |  |
| specific events. |  |

Dialog 'Select Item from Object Directory'

For modular slave, you have to clear the "Autoconfig PDO mapping" option to be able to configure the mapping manually.

The table shows all object directory entries from the EDS file of the device. For receive PDOs, CODESYS provides only the objects here with write permission (flag $=\mathrm{w}$ ); for transmit PDOs, read permission.

| "Name" | COB-ID of the PDO or the name of the mapped object as it is used in the device <br> description and in the object directory. |
| :--- | :--- |
| "Index" | Index of the object |
| "Subindex" | Subindex of the object |


| "Access type" | - "RW": Read/Write |
| :--- | :--- |
|  | - "RO": Read Only |
|  | - "WO": Write Only |
|  | - "RWW": Read/Write per SDO; write permission per PDO (==> RxPDO, |
| output from the master viewpoint, input from point of view of the slave). |  |
|  | "RWR": Read/Write per SDO; read permission per PDO (== TxPDO, input <br> from the master viewpoint, output from the point of view of the slave). <br> CONST=constant |
| "Type" | Data type of the object |
| "Default value" " | Default value of the object |
| "Bit length" | Length of the object |

## Tab 'CANopen Remote Device - SDOs'

During the initialization of the CAN bus, CODESYS transmits the current configuration settings by using SDOs (service data objects).
On this tab, you configure the necessary SDOs. You configure the necessary SDOs and determine the transmission order of the objects and the actions taken in case of a transmission error.

The object order in this list corresponds to the transmission order of SDOs to the module.

## NOTICE!

If the "Expert settings" option is not activated for the current device, then only the user-defined SDOs are shown here.

| "Add SDO" | Opens the "Select Item from Object Directory" dialog where all available SDOs <br> are displayed. The selected object is inserted after the selected object. |
| :--- | :--- |
| "Edit" | Opens the "Select Item from Object Directory" dialog and marks the corre- <br> sponding object. You can modify the object parameters or replace the object <br> with another one. |
| "Delete" | Deletes the selected objects from the list |
| "Move Up" | Moves the selected object upwards by one line. |
| "Move Down" | Moves the selected object downwards by one line. |
| "Abort on Error" | V: If an error is detected for this SDO, then the stack stops the configuration <br> phase of the current slave. The slave remains in PREOPERATIONAL mode. |
| "Jump to Line on Error" | V: The transmission is continued with the SDO that you indicated in the "Next <br> Line" column. |
| "Next Line" | Line number where processing continues if there an error is detected |
|  | Timeout for the SDO transmission. If the slave does not respond to the SDO <br> request within this time, then the transmission is canceled with a timeout. |


| "Create all SDOs" | Creates an SDO for all writable objects starting at index 16\#2000 for which a default value is given in the EDS. Only experts should use this option. It should be deactivated for standard use. |
| :---: | :---: |
| "Write complete PDO configuration" | This option forces the writing of all PDO configuration objects. In this way, you make sure that the settings in the project correspond to those of the slave. $\square$ PDOs are not deactivated explicitly. The requirement is that the "Default settings" option is activated in the common settings of the slave and the PDOs are also deactivated in the EDS. If the default values in the EDS do not match the default settings of the slave firmware, then this procedure may cause problems. In this case, you should activate this option. |

- ⓨ Chapter 6.3.2.16 "CAN onboard" on page 1594

Dialog 'Select Item from Object Directory'

The table shows all object directory entries from the device EDS file for each SDO that are writable and not larger than 4 bytes. Before you add an SDO for selection in the SDO dialog, you can modify its parameters in the fields below the table. In this way, you can also created an SDO that is not writable in the EDS file by entering a new index/subindex value.

| "Name" | COB-ID of the PDO or the name of the mapped object as it is used in the device <br> description and in the object directory. |
| :--- | :--- |
| "Index" | Index of the object |
| "Subindex" | Subindex of the object |
| "Access type" | $\bullet$ "RW": Read/Write <br> $\bullet$ <br> - "WO": Write Only <br> "RWW": Read/Write per SDO; write permission per PDO (==> RPDO, output <br> from the master viewpoint, input from point of view of the slave). <br> "RWR": Read/Write per SDO; read permission per PDO (==> TPDO, input <br> from the master viewpoint, output from the point of view of the slave). |
| "Type" | Data type of the object |$|$| "Default value" | Default value of the object |
| :--- | :--- |
| "Bit length" | Length of the object |
| "Value" |  |

## CANopen module

CANopen modules are components that you insert below a CANopen remote device.

### 6.3.2.16.2 <br> J1939

## General

J1939 is a CAN-based protocol (CAN stands for "Controller Area Network"). It was developed for serial data transmission between electronic control units (ECU) in heavy goods vehicles. The CODESYS plug-in 'DeviceEditorJ1939' provides dialogs to configure J1939 devices according to SAE J1939 standards.

See also

- $y^{4}$ Chapter 6.3.2.16 "CAN onboard" on page 1594

Bus Cycle Task
Behavior of the bus cycle for J1939

(1) Receive single package PG
(4) Receive multi-package PGs, send PGs

See also

- Tab '<device name> I/O Mapping'

J1939 manager

## General

The J1939 Manager is inserted in the device tree below the CAN bus node. It provides the J1939 parameter groups and signal database. The ECUs are inserted below the J1939 Mananger.

The "Scan Devices" command is not available for J1939.

```
- - jol J1939
---0) Device
```



Application
ImagePool
fill Bibliotheksverwalter
[夆 POU (PRG)
$\pm$ 閣 Taskkonfiguration

1)     - CANbus (CANbus)


- 



J1939_Manager (J1939_Manager)
OffBoardDiag (J1939_ECU)
KeyPad_ECU (J1939_ECU)
Controller (J1939_ECU)
(1) CANopen Manager
(2) : J1939 Manager
(3) J1939 ECU

See also

- \& Chapter 6.3.2.16 "CAN onboard" on page 1594
- \& Chapter 6.3.2.16.2.3.2 "Tab 'J1939 Manager - General"' on page 1604


## Tab 'J1939 Manager - General'

In CODESYS version 3.5 SP17 and CODESYS J1939 Manager SL version 4.O.0.0, the J1939 configurator is no longer supplied with a parameter group / signal database. The old database is no longer supported.
However, you can post-install a DBC database in the J1939 Manager. A database can be purchased, for example from CSS Electronics: https://www.csselectronics.com/screen/product/j1939-dbc-file-pgn-spn
If you do not install a database, then you can also configure parameter groups and signals manually on the "User-Defined" tab.

| "Databases" | List with names of installed databases (DBC or DB format) |
| :--- | :--- |
| "Install" | Opens the file manager to select a J1939 file (DBC or DB format). The J1939 <br> files are usually stored in "C $: \backslash$ ProgramData $\backslash$ CODESYS J J1939 Databases". |
| "Uninstall" | Uninstalls the selected database |
| "Set as default" | Sets a database as the default database. This database is then set as default in <br> the "Add Parameter Group" and "Add Signal" dialogs. |

See also

- \& Chapter 6.3.2.16 "CAN onboard" on page 1594

J1939 ECU
6.3.2.16.2.4.1 Tab 'J1939 ECU - General'................................................... 1604
6.3.2.16.2.4.2 Tab 'J1939 ECU - TX Signals'.............................................. 1605
6.3.2.16.2.4.3 Tab 'J1939 ECU - P2P RX Signals'...................................... 1607

Tab 'J1939 ECU - General'
In this dialog of the J1939 ECU editor, the general parameters of a J1939 ECU can be displayed and modified.

Table 315: "General"

| "Preferred address" | Address of the ECU. If more than one ECU with the same address exists in the <br> network, then all affected ECUs get a new address. The requirement is that the <br> ECUs allow an address change ("Arbitrary Address Capable"). |
| :--- | :--- |
| "Local Device" | (You can configure any number of local ECUs. Then every local ECU is its <br> Own ECU instance in the J1939 network. For local devices, an additional " $R X$ <br> Signals (P2P)" dialog is provided to configure received signals. |

Table 316: "ECU NAME"

| "NAME (64 bit): 16\#" | Hexadecimal 64-bit code that contains complete information about the subse- <br> quent parameters. Each time this code is modified, the respective parameter is <br> also modified. The same is true for the other direction. |
| :--- | :--- |
| "Arbitrary address capable" | VIf the ECU detects an address conflict, then it tries independently to set <br> another address. |
| "Industry group" | List of industry groups according to the definition from SAE J1939. |
| "Vehicle system instance" | The parameter depends on the "Vehicle system". The 4-bit value assigns a <br> number to each instance of the "Vehicle system". |
| "Vehicle system" | The value is defined in the SAE J1939 standard. |
| "Reserved" | Always deactivated and reserved for future SAE definitions. |
| "Function" | The parameter is defined and assigned by SAE. The range of values is 0...255, <br> but not all values are assigned. <br> The interpretation of values, which are greater than or equal to 127, depends on <br> the "IIdustr" selection. For example, the value e"33" means "Product Flow" in <br> the "Agricultural and Forestry Equipment" industry. If "Construction Equipment" <br> is selected for "Industry", then the same value means "Land Leveling System <br> Display". <br> If the value is less than 128 (0 - 127), then there is no dependency to other <br> parameters. |
| "Function instance" | The parameter is related to the "Function" field. A J1939 network can consist of <br> multiple ECUs with the same "Function". The 5-bit "Function instance" assigns <br> a number to each instance of the "Function", where 0 is assigned to the first <br> instance. |
| "ECU instance" | A J1939 network can include multiple ECUS that have the same task. For <br> example, a vehicle can have two identical ECUs, where one measures vehicle <br> speed and the other measures the trailer speed. |
| "Manufacturer code" | The 11-bit manufacturer code is assigned by SAE and indicates the company <br> that manufactured the ECU. This code is defined in the SAE J1939 document. |
| "Identity number" | The 21-bit identity number is assigned by the manufacturer and should be used <br> for assuring unique names in a product line. The manufacturer can also add <br> more information to the identity number, such as serial number and date of <br> manufacture. |

Table 317: "Communication Watchdog"

| "Enable communication <br> watchdog" | The stack checks whether the ECU transmits data within the given "Watchdog <br> time". If this does not happen, then the device is classified as "not available" and <br> highlighted in red in the device tree. |
| :--- | :--- |

See also

-     * Chapter 6.3.2.16 "CAN onboard" on page 1594


## Tab 'J1939 ECU - TX Signals'

This dialog shows the parameter groups that are transmitted to all other ECUs (broadcast) or to a specific ECU (P2P). In this dialog, you can activate and deactivate individual groups and modify their parameters. You can also add new groups or signals to the list.

| "Enable" | ■: The parameter group is transmitted. <br> $\square$ <br> The parameter group is not transmitted for a local device. For a remote <br> device, the PLC does not process this group. |
| :--- | :--- |
| "Type" | $\bullet$ "Broadcast": The parameter group is transmitted to all ECUs. <br> $\bullet$ <br> "Peer-to-Peer P2P": The parameter group is transmitted to a specific ECU. |
| "Add PG" | Opens the "Add Parameter Groups" dialog. |
| "Add Signal" | Opens the "Add Signal" dialog. The button is enabled only if you have selected a <br> parameter group. |

## Parameter

## group proper-

ties
Table 318: "General"

| "PGN" | The "parameter group number" is a unique number for addressing a parameter <br> group. |
| :--- | :--- |
| "Name" | Name of the parameter group |
| "Description" | Description of the parameter group |
| "Length" | The length of the message data (0...1785 bytes). Due to the maximum array <br> length of 8 bytes, messages with over 8 bytes are transmitted as multipackages. |

Table 319: "Transmission settings"

| These settings are provided only for the TX parameter groups of local devices. |  |
| :--- | :--- |
| "Priority" | Priority of the parameter group (0..7). Priority 0 is the highest and 7 is the lowest. |
| "Target address" | The target address is needed for P2P parameter groups only. |
| "Transmission mode" | Determines the time when a parameter group is transmitted (for local devices). <br> $\bullet$ "Mode change": The PG is transmitted when the value of the signal changes. <br> - "Cyclic": The PG is transmitted after a specified number of PLC cycles (see <br> cycle time factor). <br> - "On request": The PG is transmitted on request of another device. <br> - "Application-controlled": The PG is transmitted when triggered by the appli- <br> cation. |
| "Cycle time factor" | Number of PLC cycles after which the parameter group is transmitted. <br> Only applies for cyclic transmission. |

## Signal parame-

ters
Table 320: "General"

| "SPN" | Suspect Parameter Number. One of the numbers assigned by the SAE for a <br> specific parameter in a parameter group. |
| :--- | :--- |
| "Name" | Name of the parameter |
| "Description" | Description of the parameters |
| "Length (bits)" | Length of the signal (in bits: $1 \ldots 14280)$. |
| "Byte position (0..1784)" | Start byte in the parameter group (0...1784). |
| "Bit position (0..7)" | Bit position of the start byte (0..7). |

Table 321: "Conversion"

| "Conversion" | TRUE: The value is calculated with scaling and offset. |
| :--- | :--- |
| "RAW data type" | Format of the raw data: Unsigned / Signed / Float / Double |
| "Byte order" | Little endian or big endian of the raw signal |
| "Scaling" | Factor (for "Conversion" =TRUE) |
| "Offset" | Offset (for "Conversion" =TRUE) |
| "Minimum value" | Expected minimum value of the converted signal (for informational purposes <br> only) |
| "Maximum value" | Expected maximum value of the converted signal (for informational purposes <br> only) |
| "Unit" | Unit of the converted signal |
| "ICE data type" | Resulting data type of the I/O channels |

See also

- \& Chapter 6.3.2.16 "CAN onboard" on page 1594


## Tab 'J1939 ECU - P2P RX Signals'

This dialog is available for local ECUs only. It shows all PGs (parameter groups) that should be received by other ECUs. In this dialog, individual groups can be activated and deactivated as well as their parameters modified. New groups or signals can also be added to the list.
The commands and parameters of this dialog are the same as those on the " $T X$ Signals" tab.
See also

- © Chapter 6.3.2.16.2.4.2 "Tab 'J1939 ECU - TX Signals"' on page 1605


### 6.3.2.16.3 Command 'Scan for Devices'

Function: The command establishes a brief connection to the hardware and determines the devices in the network. Then you can apply the devices found into the device tree of your project.
Call: Menu bar: "Project"; context menu of a device object in the device tree
Requirement: The communication settings to the controller are correct. The gateway and the PLC are started. The device supports the scan function.
The following devices provide the scan function: EtherCAT master, EtherNet/IP Scanner (IEC), Sercos master, CANopen Manager, CANopen Manager SIL2, PROFINET controller and PROFIBUS DP master.

You can perform the device scan immediately if the scan function is permanently implemented in the PLC. When scan function is implemented in a library, you have to log in only one time to download the library to the controller.

The command refers to the master controller selected in the device tree. For example, an already inserted PROFINET IO controller can be selected and the command used to determine the I/O devices and I/O modules assigned to it.
After performing the scan operation, the "Scan Devices" dialog opens and displays the found devices.

Dialog 'Scan
Devices'
Table 322: "Scanned Devices"
$\left.\begin{array}{|l|l|}\hline \text { "Device name, Device type, } \\ \text { Address, Station name, etc." } & \begin{array}{l}\text { Data about the scanned device depending on network type. } \\ \text { When you change a value in the list of scanned devices, the value is shown } \\ \text { in italics. This indicates that the new value has been changed in the editor in } \\ \text { CODESYS, but not in the device. When you download the value to the device, it } \\ \text { is shown normally. } \\ \text { Value that indicate differences between the project and the scanned device are } \\ \text { shown in orange. } \\ \text { If multiple device descriptions are available for the scanned device, then the } \\ \text { name is displayed in bold. The selection of the matching device description is } \\ \text { resolved differently for different fieldbuses. For more detailed information, see } \\ \text { the corresponding fieldbus chapters. } \\ \text { If a device description cannot be found, then the following message is shown: } \\ \text { "Attention! The device was not found in the repository." Depending on the bus } \\ \text { system, additional information is displayed, such as manufacturer number and } \\ \text { product number. The device cannot be inserted into the project without the } \\ \text { installed device description. }\end{array} \\ \hline \text { "Show differences to project" } & \begin{array}{l}\text { V: The table in the dialog also shows additional configured devices (in the } \\ \text { device tree of the project). }\end{array} \\ \square: \text { The table shows all scanned devices. The configured devices are not shown. }\end{array}\right\}$

## 0

## NOTICE!

If you insert devices, which are available in the device tree, to the device tree with "Copy All Devices to Project", then the following should be noted. The data of the "Process Data" and "<...> I/O Mapping" tabs of the existing devices can be overwritten with the data of the recently inserted devices.

Table 323: "Configured Devices"
This part of the dialog is visible only when you select the "Show differences to project" option.
Differences between the scanned and configured devices are color-coded. Devices displayed in green are identical on both sides. Devices displayed in red are available only in the view of the scanned or configured devices.

|  | If you have selected a device in both views, then the scanned devices are <br> inserted above the selected configured device. |
| :--- | :--- |
| If you have selected a device in both views, then the scanned devices are |  |
| inserted below the selected configured device. |  |

See also

- $\&$ Chapter 6.3.2.16 "CAN onboard" on page 1594


### 6.3.2.16.4 Tab 'CANbus - General'

Table 324: "General"

| "Network" | Number of the CAN network that is linked via the CAN bus interface. Permitted <br> values: 0 to 100. |
| :--- | :--- |
| "Baud rate" | Baud rate (in bits per second) for transmitting data on the bus. The default value <br> is used from the device description file ( <br>  <br> can .devdesc) of the CAN bus device. You <br> can selt the baud rate from the list box or type it directly into the input field. |

See also

- $\sum^{*}$ Chapter 6.3.2.16 "CAN onboard" on page 1594


### 6.3.2.17 EtherCAT configurator

### 6.3.2.17.1 General

Onboard EtherCAT support is currently limited to AC500-eCo PM5072-ETH and PM5082-ETH. It is not supported by any other AC500 PLC.

## Two alternatives are available for using EtherCAT:

- Onboard EtherCAT for the AC500-eCo
- EtherCAT for the AC500 with the CM579 communication module

Further information can be found here:
Onboard EtherCAT
« EtherCAT configurator
\& Communication schema onboard Ethernet/EtherCAT
«) Motion Control
EtherCAT
② CM579-ETHCAT
4) Communication schema onboard Ethernet/EtherCAT
«) Motion Control

```
Refer to the general description for information about the following tabs of the device editor.
- \({ }^{〔}\) Chapter 6.4.1.21.2.8.12 "Tab '<device name> I/O Mapping'" on page 2444
- \(\boldsymbol{y y}^{4}\) Chapter 6.4.1.21.2.8.13 "Tab '<device name> IEC Objects'" on page 2449
- \(\quad\) ² Chapter 6.4.1.21.2.8.4 "Tab 'Parameters'" on page 2433
- \(\quad\) Chapter 6.4.1.21.2.8.19 "Tab 'Status'" on page 2460
- \(\quad\) \& Chapter 6.4.1.21.2.8.20 "Tab 'Information"' on page 2460
Only in the case of special features is there an additional help page for the specific device editor.
If the "<device name> Parameters" tab is not shown, then select the "Show generic device configuration editors" option in the CODESYS options ("Device Editor" category).
```

The configuration of EtherCAT modules is based on the device description files for the used master and slave devices employed and can be adapted in the project in configuration dialogs. In order to make sure that the simplest and most error-free use possible, we recommend for standard applications that you enable the option for the "Automatic Configuration" of the master, so that the majority of the configuration settings are dopne automatically.


#### Abstract

Optional devices

The "Optional devices" function can be used for variable hardware configurations. At the start of the stack, the system checks whether optional devices are available. If the devices are not found, then they are deactivated automatically. However, the devices can also be activated at a later date if, for example, it is a device at the end of the network line. On the other hand, if the "Automatically restart slaves" option is selected, then the activated devices are automatically switched to "operational", thereby activating the process data.


The bus cycle task is set in the general PLC settings.

Access to the EtherCAT configuration by the application takes place via instances of the EtherCAT master and EtherCAT slave. If the EtherCAT master or EtherCAT slaves are inserted as objects into a project, instances are automatically created for master and slaves which can be addressed in the application program. For example a restart, a stop or a status check of the EtherCAT device can be performed from the application.
Furthermore, the EtherCAT library offers function blocks for the reading and writing of individual parameters, even during bus operation.

See also

- ${ }^{4}$ Chapter 6.3.2.17.2.1 "Tab 'EtherCAT Master - General' " on page 1610
- ${ }^{4}$ Chapter 6.4.1.21.4.14.7 "Dialog 'Options' - 'Device Editor'" on page 2786
- ⓨ Chapter 6.4.1.21.2.8.10 "Tab 'PLC Settings'" on page 2439


### 6.3.2.17.2 EtherCAT master

## Tab 'EtherCAT Master - General'

Object: EtherCAT Master
The tab is used for the configuration of the basic settings for the EtherCAT Master. The basic settings are preset from the device description file.

Settings of the configuration parameters

## NOTICE!

The auto-configuration mode ("Autoconfig master/slaves" option) is selected by default and is adequate for standard applications. If the mode is not selected, then all configuration settings for the master and the slave(s) have to be done manually. Expert knowledge is required to do this. The auto-configuration mode option has to be switched off to configure slave-to-slave communication.

| "Autoconfig master/slaves" | The main part of the master and slave configuration is done automatically, <br> based on the device description file and implicit calculations. The dialog for the <br> FMMU/Sync settings is not available. |
| :--- | :--- |
|  | Even if this option of the master is selected, an expert mode can be enabled <br> explicitly for each individual slave, which allows for manual editing of the auto- <br> matically generated process data configuration. |

Table 325: "EtherCAT NIC Settings"

| "Target address (MAC)" | MAC address of the device in the EtherCAT network that is to receive the <br> telegrams. <br> Options |
| :--- | :--- |
| "Broadcast": A "Target address (MAC)" does not have to be specified. |  |
| $\bullet$ "Redundancy": Enabled when the bus is constructed in a ring topology and |  |
| redundancy is to be supported. With this function, the EtherCAT network |  |
| remains functional even in the case of a broken cable. When this function is |  |
| enabled, the parameters have to be defined in the "Redundancy EtherCAT |  |
| NIC Settings" area. |  |

Table 326: "Redundancy EtherCAT NIC Settings"
These settings are displayed only when the "Redundancy" option is selected. Here the parameters of the additional device are defined according to the description for "EtherCAT NIC Settings".

Table 327: "Distributed Clock"
\(\left.$$
\begin{array}{|l|l|}\hline \text { "Cycle time ( } \mu s \text { )" } & \begin{array}{l}\text { Time span after which a new data telegram is dispatched on the bus. When } \\
\text { the "Distributed Clock" function is enabled in the slave, the master cycle time } \\
\text { specified here is transferred to the slave clocks. As a result, a precise synchro- } \\
\text { nization of the data exchange can be achieved. This is particularly important } \\
\text { when spatially distributed processes require simultaneous actions. An example } \\
\text { of a simultaneous action is applications in which multiple axes have to execute } \\
\text { coordinated movements at the same time. A very precise, network-wide time } \\
\text { base with a jitter of considerably less than } 1 \text { microsecond can be achieved in this } \\
\text { way. }\end{array} \\
\hline \text { "Sync offset" } & \begin{array}{l}\text { Parameter for setting the delay time between the DC time base of the EtherCAT } \\
\text { Slave and the cycle start of the PLC. The default value is 20\%. This time is } \\
\text { active at the same time for all slaves with DC. An offset of 20\% means that the } \\
\text { timer interrupt in the EtherCAT Slave takes place 20\% before the next IEC cycle. } \\
\text { This means in the case of }\end{array}
$$ <br>
- FrameAtTaskStart = FALSE when the EtherCAT data is sent at the end of <br>
the PLC cycle: <br>
The PLC cycle may require 80\% of the bus cycle time minus the delay time <br>
in the runtime, and this without the master no longer placing the current <br>
process data on the bus in time (assuming that the EtherCAT Slave expects <br>

the new data exactly with the sync interrupt).\end{array}\right\}\)| FrameAtTaskStart = TRUE (default value when using CODESYS SoftMo- |
| :--- |
| tion): |
| For the controller program, nearly 100\% of the cycle is always available. |
| Here the "Sync offset" determines only when the EtherCAT data of the |
| master is exchanged to and from the slaves relative to the time base of |
| the EtherCAT Slave. |

Table 328: "Options"

| "Use LRW instead of LWR/ | Direct communication from slave to slave is possible. Combined read/write <br> LRD" |
| :--- | :--- |
| commands (LRW) are used instead of separate read commands (LRD) and write |  |
| commands (LWR). |  |

Table 329: "Master Settings"
These settings can be edited only when the "Autoconfig master/slaves" option is deactivated. Otherwise this is done automatically and they are not visible here.

| "Image In Address" | First logical address of the first slave for input data |
| :--- | :--- |
| "Image Out Address" | First logical address of the first slave for output data |

See also

- «y Chapter 6.3.2.17.3.1 "Tab 'EtherCAT Slave - General' " on page 1613
- $\Leftrightarrow$ Chapter 6.3.2.17 "EtherCAT configurator" on page 1609

Tab 'EtherCAT Master - Sync Unit Assignment'

Sync unit assignment is not supported in Onboard EtherCAT.

## Tab 'EtherCAT Master - Parameters'

Object: EtherCAT Master
The tab contains the master parameters which are defined in the device description file.
When the auto-configuration mode is selected in the "Master" dialog, the parameters are set here automatically according to the specifications from the device description file and the network topology. Nothing should be changed in the generic editor because an invalid configuration can be set here.

| "Value" | Editable: A change is effective only when the auto-configuration mode is disa- <br> bled. <br> Whether or not the change becomes effective depends on the respective param- <br> eter. |
| :--- | :--- |

See also

- ${ }^{*}$ Chapter 6.3.2.17.3.9 "Tab 'EtherCAT Slave - Parameters' " on page 1623
- $\Leftrightarrow$ Chapter 6.3.2.17 "EtherCAT configurator" on page 1609
- $\Leftrightarrow$ Chapter 6.4.1.21.2.8.4 "Tab 'Parameters'" on page 2433


### 6.3.2.17.3 EtherCAT slave

Tab 'EtherCAT Slave - General'
Object: EtherCAT Slave
The basic settings for the EtherCAT Slave are configured on this tab. The basic settings are preset from the device description file.

Table 330: "Address "
Fields can be edited only when the auto-configuration mode of the EtherCAT Master is disabled.

| "Autolnc address" | Self-incrementing address (16-bit) that results from the position of the slave <br> in the network. The address is used only during the system boot when the <br> master assigns the EtherCAT addresses to its slaves. When the first message <br> runs through all the slaves for this purpose, each slave increments its "Autolnc <br> address"by 1. The slave with address 0 then gets the data. A possible input <br> here is "-8". |
| :--- | :--- |
| "EtherCAT address" | Final address of the slaves, assigned by the master during bootup. The address <br> is independent of the position of the slave in the network. |

Table 331: "Additional"

| "Expert settings" | $\boxed{\checkmark}:$ Additional settings are possible for the startup checking and time monitoring <br> (see below). The "Expert Process Data" tab is also available in the device editor. <br> However, expert settings are not required for standard applications. The auto- <br> configuration mode is recommended and sufficient for standard applications. |
| :--- | :--- |
| "Optional" | At the start of the stack, the system checks whether optional devices are avail- <br> able. <br> V. The slave is defined as optional and no error message is generated if the <br> device is missing from the bus system. If a device is not found, then it is disabled <br> automatically and displayed in gray in the device tree. A corresponding message <br> is displayed in the logger. <br> Note: If you define a slave as "optional", then it has to have a unique identi- <br> fication. You can change this by means of the three possible settings in the <br> "Identification" section. |
| Available only when the "Autoconfig master/slaves" option is selected in the |  |
| settings of the EtherCAT Master and the EtherCAT Slave supports this function. |  |,

Table 332: "Distributed Clock"

| "Select DC" | List box with all settings for the distributed clocks of the device description file |
| :--- | :--- |
| "Enable" | $\checkmark:$ Cycle time for the data exchange. It is displayed in the "Sync unit cycle <br> $(\mu s) "$ input field and determined by the cycle time of the master. As a result, the <br> master clock can synchronize the data exchange in the network. |

The "Sync0" and "Sync1" settings described below are slave-dependent:

Table 333: "Sync0"

| "Enable Sync 0" | 冋: Synchronization unit "Sync0" is used. A synchronization unit describes a set <br> of process data that is exchanged synchronously. |
| :--- | :--- |
| "Sync unit cycle" | V: The master cycle time (multiplied by the factor selected from the list box) is <br> used as the synchronization cycle time for the slave. "Cycle time ( $\mu \mathrm{s}$ )" displays <br> the cycle time currently set. |
| "User-defined" | $\square:$ A custom cycle time (in microseconds) can be specified in the "Cycle time <br> $(\mu \mathrm{s})$ " field. |

Table 334: "Sync1"

| "Enable Sync 1" | Synchronization unit "Sync1" is used. A synchronization unit describes a set of process data that is exchanged synchronously. |
| :---: | :---: |
| "Sync unit cycle" | The master cycle time (multiplied by the factor selected from the list box) is used as the synchronization cycle time for the slave. The "Cycle time ( $\mu \mathrm{s}$ )" field displays the cycle time currently set. |
| "User-defined" | A custom cycle time (in microseconds) can be specified in the "Cycle time $(\mu s)$ " field. |

Table 335: "Diagnosis"
This area is only displayed in online mode.

| "Current State" | State of the slave |
| :--- | :--- |
|  | Possible states: "Init", "Preoperational", "Safe Operational", and "Operational" |
|  | The state Operational indicates that the slave configuration has been correctly <br> completed and that process data (inputs and outputs) are being accepted. |

Table 336: "Startup Checking"

| "Check vendor ID" | By default the vendor ID and product ID of the device are checked against the <br> current configuration settings when the system boots up. If they do not agree, <br> then the bus is stopped and no further actions are executed. This is done to <br> prevent an incorrect configuration from being loaded onto the bus system. <br> Options for deactivating the corresponding check. |
| :--- | :--- |
| "Check product ID" | Q: The revision number is checked during the system bootup according to your <br> selection in the list box. |
| "Check revision number" | "Download expected slot <br> configuration" |
| configurations verification of the configured and actual module configuration. If the the the device still switches to "Run". In this case, <br> an entry is made in the device logbook. |  |

Table 337: "Timeouts"
By default, watchdog is not defined for the following actions. If necessary, an appropriate timeout can be specified here (in milliseconds):

| "SDO access" | Transmits the SDO list at system start. Specified in milliseconds. |
| :--- | :--- |
| " $->P "$ | Switch from "Init" mode to "Preoperational" mode. Specified in milliseconds. |
| " $P$-> S / S -> O" | Switch from "Preoperational" mode to "Safe Operational" mode, or from "Safe <br> Operational" mode to "Operational" mode. Specified in milliseconds. |

Table 338: "DC Cyclic Unit Control: Assign to Local $\mu$ C"
One or more options for the "Distributed Clock" function can be activated here that should be used on the local microprocessor. The check is performed in the registry at 0x980 in the EtherCAT Slave. Possible settings:

| "Cycle unit" |  |
| :--- | :--- |
| "Latch unit 0" |  |
| "Latch unit 1" |  |

Table 339: "Watchdog"

| "Set multiplier" | The PDI watchdog and SM watchdog receive their impulses from the local ter- <br> minal clock divided by the watchdog multiplier. |
| :--- | :--- |
| "Set PDI watchdog" | This watchdog triggers when there is no PDI communication with the EtherCAT <br> Slave controller for longer than the PDI (Process Data Interface) watchdog time <br> which has been set and activated. |
| "Set SM watchdog" | This watchdog triggers when there is no EtherCAT process data communication <br> with the terminal for longer than the SM (SyncManager) watchdog time that has <br> been set and activated. |

Table 340: "Identification"
In this section, you set the device identification of the slave. As a result, you can make the address of the slave independent of its position in the bus.
The following options are visible only when the "Activate expert settings" option or "Optional" option is selected. If you have identified the slave as "Optional", then you have to assign a unique ID to it.

| "Disabled" | The identification of the slave is not checked. |
| :--- | :--- |
| "Configured station alias (ADO <br> ox0012)" | Address that is stored in the EEPROM of the device. <br> You can change the value in the "Scan Devices" dialog or in online mode. For <br> stock devices, you need to assign this number one time. This means that you <br> have to connect the device one time to an EtherCAT Master and save the <br> number. |
| "Write to EEPROM" | Visible in online mode only for "Configured station alias". Writes the defined <br> address for "Value" to the EEPROM of the slave. |
| "Explicit device identification <br> (ADO Ox0134)" | The device identification is hard set on the hardware (for example, by DIP <br> switches). It is displayed in "Actual address". |
| "Data Word (2 Bytes)" | A 2-byte value for the identification is saved in the slave. |
| "Value" | Expected value for the check. If the actual value does not correspond to this <br> setting, then an error is issued. |
| "ADO (hex)" | Initial value from the device description. You can change this value in the "Data <br> word" option. |
| "Actual address" | Visible in online mode only. Displays the address of the slave. You can use this <br> display for checking the success of the "Write to EEPROM" command. |

## Tab 'EtherCAT Slave - FMMU/Sync'

## Object: EtherCAT Slave

The tab shows the FMMUs and Sync Manager of the EtherCAT Slave as they are defined in the device description file. There is an option to edit the FMMUs and Sync Manager (for example, for the configuration of slave-to-slave communication).

Requirement: The auto-configuration mode in the EtherCAT Master is disabled.


Table 341: "FMMU"
The table shows the Fieldbus Memory Management Units of the slave, which are used for handling the process data. In each case the allocation of the logical address ("Global Start Address") to a physical address ("Phys. start address") is defined. Bit-by-bit mapping is possible.

| "Edit" |  |
| :--- | :--- |
| "Add" |  |
| "Delete" |  |

Table 342: "Edit FMMU"

| "Global Start Address" |  |
| :--- | :--- |
| "Length" |  |
| "Start bit" |  |
| "End bit" |  |
| "Phys. start address" |  |
| "Phys. start bit" |  |
| "Access" | "Read" |
|  | "Write" |
| "Flags" | "Enable" |

Table 343: "Sync Manager"
Display and editing of the synchronization manager of the slave. The physical start address, the type of access, the buffer, and the physical address to which the interrupts are to be sent (as well as others) are defined for each available Sync Manager type (mailbox in, mailbox out, inputs, outputs).

Table 344: "Edit Syncman"

| "Phys. start address" |  |
| :--- | :--- |
| "Length" |  |
| "Buffer" | "1" |
| "3" |  | \left\lvert\, | "Read" |
| :--- | :--- |
| "Write" |$\quad$| "Interrupts" | "to EtherCAT" <br> "to PDI" |
| :--- | :--- |
| "Flag control" | "Activate" |
| "Watchdog" | "Trigger" |
| "SyncMan type" | "" |\right.

## Tab 'EtherCAT Slave - Expert Mode Process Data'

Object: EtherCAT Slave
The tab provides another more detailed view of the process data, which is also displayed in the "Process Data" dialog. Moreover, the download of the PDO assignment and the PDO configuration is enabled here.
Requirement: The expert settings for the slave are selected.

Table 345: "Sync Manager"
List of the Sync Managers with data size and PDO type

Table 346: "PDO assignment (16\#1C12)"
List of the PDOs assigned to the selected "Sync-Manager".
When a check box is selected, the PDOs are enabled and I/O channels are created. This is similar to the simple PDO configuration view.

Table 347: "PDO list"
List of the PDOs assigned to the selected "Sync-Manager".
You can add new entries or edit or delete existing entries by executing the respective commands ("Add", "Delete", "Edit") in the command bar or context menu.

Table 348: "Edit PDO list"

| "Name" |  |
| :--- | :--- |
| "Index" | - "TxPDO (input)": 圆: The PDO is transmitted from the master to the slave. <br> - "RxPDO (output)": <br> "Direction" The PDO is transmitted from the slave to the master. |
| "Flags" | - "Required": The PDO is required and cannot be disabled in the "PDO <br> assignment". <br> "Windows contents": The contents of the PDO are fixed and cannot be modi- <br> fied. It is then not possible to add entries in "PDO contents". <br> - "Virtual PDO": Reserved for future use |
| "Exclude PDOs" | It is possible to define an exclusion list. When a PDO is enabled in the "PDO <br> assignment", others are disabled and cannot be enabled. |
| "Sync unit" | ID of the Sync Manager to which the PDO is to be assigned |

Table 349: "PDO Contents"
Displays the contents of the PDOs selected in the "PDO list". You can add new entries or edit or delete existing entries by executing the respective commands ("Add", "Delete", "Edit") in the command bar or context menu. You can change the PDO order by clicking "Move Up" and "Move Down".

Table 350: "Download"

| "PDO assignment" | : Specific CoE commands for initializing the $0 \times 1 \mathrm{cxx}$ objects are generated <br> and written to the slave. |
| :--- | :--- |
| "PDO configuration" | : The CoE commands for $0 \times 16 \mathrm{xx}$ or $0 \times 1 \mathrm{axx}$ are generated, and then the <br> PDO mapping is downloaded to the slave. Normally, the default values originate <br> from the ESI file and the device has to support this functionality. For example, <br> if a device has a fixed configuration, then these commands are regarded as <br> flawed. |
| "Load PDO info from the <br> device" | The current PDO configuration is read from the slave and entered into the <br> configuration. The lists in the upper and lower right are then deleted and filled <br> with the read data. This is especially useful when the ESI file is incomplete and <br> the configuration is available only on the slave. |

## Tab 'EtherCAT Slave - Process Data'

Object: EtherCAT Slave
The tab of the EtherCAT configurator displays the process data for the inputs and outputs of the slave. The data is preset from the device description file.

Table 351: "Select the Outputs"
The table shows the outputs of the slave defined by "Start address", "Type", and "Index".
If outputs of the device are enabled here (for writing), then these outputs can be assigned to project variables in the "EtherCAT I/O Mapping" dialog.

Table 352: "Select the Inputs"
The table shows the inputs of the slave defined by "Name", "Type", and "Index".
If inputs of the device are enabled here (for reading), then these inputs can be assigned to project variables in the "EtherCAT I/O Mapping" dialog.

See also

- «\& Chapter 6.3.2.17.3.1 "Tab 'EtherCAT Slave - General' " on page 1613
- $\Leftrightarrow$ Chapter 6.3.2.17 "EtherCAT configurator" on page 1609


## Tab 'EtherCAT Slave - Startup Parameters'

Object: EtherCAT Slave
On the tab, the SDOs (service data objects) for 'CAN over EtherCAT' (CoE) or the IDNs (identification numbers) for 'Servodrive over EtherCAT' (SoE) are defined for the current slave. These parameters are determined for the device when the system is started.
The object directory with the required data objects is described in the EtherCAT XML description file or in an EDS file that is referenced in the XML file.
Requirement: The device supports 'CAN over EtherCAT' or 'Servodrive over EtherCAT'.

Some modules that are inserted below a slave have their own startup parameters. These parameters are also displayed in this list but cannot be edited here. The parameters are modified in the editor of the corresponding module.

List of SDOs or IDNs
The order (from top to bottom) specifies the order in which the objects are transferred to the module.

| "Line" | Line number |
| :--- | :--- |
| "Index:Subindex" | For CoE only |
| "IDN" | For SoE only <br> Identification number |
| Name |  |
| "Bit length" | Bit length of the SDO or IDN |
| "Abort on Error" | $\boxed{\checkmark}$ : In case of error, the transfer is aborted with an error status. |
| "Jump to Line on Error" | Ø: In case of error, the transfer is resumed with the SDO or IDN at <br> the specified "Line". |


| "Next Line" | $\boxed{ } /$ : The transfer is resumed with the SDO or IDN at the next line. |
| :--- | :--- |
| "Comment" | Input field for comments |
| "Move Up" | Moves the selected line upwards by one line |
| "Move Down" | Moves the selected line downwards by one line |
| "Add" | Opens the "Select Item from Object Directory" dialog adding SDOs <br> or IDNs. |
| "Delete" | Removes the selected entry. |
| "Modify" | Opens the "Select Item from Object Directory" dialog for changing <br> the parameters of the selected SDO or IDN |

Dialog 'Select
Item from
Object Directory'

The dialog lists all available object directory entries as defined in the XML file. The parameters of the objects can be modified in this dialog. New objects can also be created. This is useful when none or only an incomplete object directory exists.

Table 353: "CAN over EtherCAT"
List of available object directory entries as defined in the EDS file.

| Column "Index:Subindex" | Identifies the entry in the object directory |
| :--- | :--- |
| Column "Name" |  |
| Column "Flags" | Display of access flags: RW (read/write), RO (read only), WO (write only) |
| Column "Base Value" | Editable (double-click to open) |
| Input fields | Input field for displaying and changing the name |
| "Name" | By specifying new index/subindex entries, a new object can be added to the <br> SDO that is not yet described in the EDS file. |
| "Index: 16\#" | Range of values of the object |
| "Subindex: 16\#" | Each value may be max. one byte (0-255). It can also be a hexadecimal in IEC <br> syntax (for example, 16\#ad). <br> If the "Byte array" option is enabled, then the values have to be specified as a <br> comma-separated list (for example, 1,2,3,4). |
| "Bit length" | "Value" |
| "Full access" | The complete object is written with one access and all subindexes are set at <br> the same time. The time needed for the transfer is reduced because not every <br> subindex has to be transferred individually. |
| "Byte array" | $\boxed{V}$ Values can be specified as a comma-separated byte array. |

Table 354: Servodrive over EtherCAT

| List of available object directory entries as defined in the XML file. |  |
| :---: | :---: |
| Column "IDN" | Identification number |
| Column "Base Value" | Base value of the IDN. <br> Double-click to modify. |
| Input fields |  |
| "IDN" | Identification number: Composed from the subsequent parameters <br> - "S": Standard data <br> - "P": Product-specific data <br> - "PSet": Parameter set <br> - "Offset" |


| "Bit length" | List box for selecting the bit length |
| :--- | :--- |
| "Value" | List box for selecting the value |
| "Channel" | If the object has multiple subobjects, then this list box is displayed automatically. |
| "As list" | Parameters are loaded as a list. The first four bytes indicate the length. <br>  <br>  |

See also

- «\& Chapter 6.3.2.17.3.1 "Tab 'EtherCAT Slave - General' " on page 1613
- $\Leftrightarrow$ Chapter 6.3.2.17 "EtherCAT configurator" on page 1609


## Tab 'EtherCAT Slave - Diagnostics History'

Object: EtherCAT Slave
The tab reads the diagnosis data ( $0 \times 10 f 3$ object) of the slave and displays it in a tree structure. Moreover, the messages are issued in plain language.
Requirement: The application is in online mode. The attribute DiagHistory in the ESI file is set.

| "Update history" | The data is read again and the view is refreshed. <br> New messages are displayed in the "Flags" column with " $N$ " (New). Acknowl- <br> edged messages with " $Q$ ". |
| :--- | :--- |
| "Auto update" | The data is updated automatically. |
| "New messages only" | Acknowledged messages are not displayed. |
| "Acknowledge messages" | All messages with the " $N$ " flag are acknowledged. Then the flag changes to " $Q$ ". |

See also

- $\quad>$ Chapter 6.3.2.17.3.1 "Tab 'EtherCAT Slave - General' " on page 1613


## Tab 'EtherCAT Slave - Online'

Object: EtherCAT Slave
The tab provides slave status information and functions for the file transfer to the slave via EtherCAT (FoE).
Requirement: The expert settings for the slave are selected and the application is logged in to the controller.

Table 355: "State machine"

| "Init" | Use for debugging purposes |
| :--- | :--- |
| "Boot Strap" | The slave is switched to Bootstrap Mode. <br> Required if firmware files should be transferred to or from the slave |
| "Pre-Op" | Use for debugging purposes |
| "Safe-Op" | Use for debugging purposes |
| "Op" | Use for debugging purposes |


| "Current status" |  |
| :--- | :--- |
| "Requested status" |  |

Table 356: "DLL status"

| "Port A" | - No Carrier / Open: No carrier signal exists at the port, but the port is <br> open. <br> "Port B" |
| :--- | :--- |
| "Port C" | No Carrier / Closed: No carrier signal exists at the port and the port is <br> closed. |
| "Port D" | - Carrier / Open: A carrier signal exists at the port and the port is open. <br> Carrier / Closed: A carrier signal exists at the port, but the port is <br> closed. <br> Note: If the function is not supported, then the status can be hidden. |

Table 357: "File Access over EtherCAT"
$\left.\begin{array}{|l|l|}\hline \text { "Download" } & \text { Download of firmware files } \\ \text { A dialog opens to save the firmware file. In this dialog, the input of a string } \\ \text { as well as a password is necessary in order to execute the file transfer. The } \\ \text { information should be taken from the data sheet of the slave. }\end{array}\right\}$

Table 358: "E²PROM access"

| "Write E2PROM" | Writes the configuration of the slave to the E ${ }^{2}$ PROM. <br> A dialog for storing the configuration opens in the same way as for "File access over EtherCAT". |
| :---: | :---: |
| "Read E2PROM" | Reads the configuration of the slave from the $E^{2} P R O M$. <br> Upload of firmware files <br> A dialog for opening the configuration opens in the same way as for "File access over EtherCAT". |
| "Write $E^{2} P R O M X M L "$ | Writes the slave configuration directly from the XML file into the device. <br> Can be executed only when configuration data exists in the XML file (<ConfigData> section). |

- $\boldsymbol{y y}^{2}$ Chapter 6.3.2.17.2.1 "Tab 'EtherCAT Master - General' " on page 1610
- $\boldsymbol{y y}^{2}$ Chapter 6.3.2.17.3.1 "Tab 'EtherCAT Slave - General' " on page 1613


## Tab 'EtherCAT Slave - CoE Online'

Object: EtherCAT Slave
The tab shows the object directory of the ESI file or the slave.
Requirement: The application is in online mode. The expert settings in the "Slave" dialog have to be enabled and the slave has to support CoE.

| "Read objects" | The object directory is read one time. |
| :--- | :--- |
| "Auto update" | The objects are read in cycles. |
| "Offline from ESI file" | The dialog shows the contents of the object directory from the device descrip- <br> tion. |
| "Online from device" | The dialog shows the contents of the object directory of the device. SDOInfo <br> has to be enabled in the ESI file. |
|  |  |
| "Flags" | $\bullet$ "RO": The value is write-protected. <br> $\bullet$ <br> "RW": The value can be modified. |
| "Type" | Data type of the parameter |
| "Value" | You can double-click in the text field to edit this value. |

- $\Rightarrow$ Chapter 6.3.2.17.2.1 "Tab 'EtherCAT Master - General' " on page 1610
- ※y Chapter 6.3.2.17.3.1 "Tab 'EtherCAT Slave - General' " on page 1613


## Tab 'EtherCAT Slave - Parameters'

Object: EtherCAT Slave
The tab contains the slave parameters which are defined in the device description file.
When the auto-configuration mode of the master is selected, the parameters are set here automatically according to the specifications from the description file and the network topology. For standard applications, it is also not normally required to edit them.

| "Value" | Only a few parameters are editable. A change is effective only when the auto- <br> configuration mode is disabled. <br> Basically, the user should not modify anything here because doing so could <br> create an invalid configuration, which would prevent the slave from entering into <br> the operational state. |
| :--- | :--- |

- ${ }^{*}$, Chapter 6.3.2.17.3.1 "Tab 'EtherCAT Slave - General' " on page 1613
- ※y Chapter 6.3.2.17.2.3 "Tab 'EtherCAT Master - Parameters' " on page 1613
- $\boldsymbol{y y}^{4}$ Chapter 6.3.2.17 "EtherCAT configurator" on page 1609


## Tab 'EtherCAT Slave - EoE Settings'

## Object: EtherCAT Slave

This tab is used to configure the communication settings for the individual slaves that support Ethernet over EtherCAT (EoE).

Table 359: "Settings"

| "Virtual Ethernet Port" | ■: Enables the EOE functionality of the slave. A unique "Virtual MAC ID" has to <br> be defined. |
| :--- | :--- |
| "Virtual MAC ID" | Input field for the "Virtual MAC ID" |
| "Switch port" | $\bullet:$ The device acts as a switch. No additional network settings are required. |
| "IP port" | ๑: The device acts as an IP port. The "IP Settings" have to be configured. |

Table 360: "IP Settings"
The Ethernet communication parameters have to be set according to the parameters of the virtual Ethernet adapter.

| "IP address" | IP address of the slave in the network (length: 4 bytes) |
| :--- | :--- |
| The IP port has to be in the same range as the virtual Ethernet adapter. For |  |
| example, if the address of the network adapter is 192.168.1.1 and the subnet |  |
| mask is 255.255.255.0, then the IP port has to be in the range from 192.168.1.2 |  |
| to 192.168.1.254. |  |

### 6.3.2.17.4 EtherCAT module

## Tab 'EtherCAT Module - Startup Parameters'

## Object: EtherCAT Module

The SDOs (Service Data Objects) or IDNs that transmit specified parameters to the device at the system start are defined on this tab for the current module.
The object directory with the required data objects is described in the EtherCAT XML description file or in an EDS file that is referenced in the XML file.
Requirement: The device supports CAN over EtherCAT or Servodrive over EtherCAT

Some modules have their own start parameters which are displayed on the tab. The parameters can be modified there. Likewise, the parameters are also displayed in the slave, but they are blocked there.

Table 361: SDO table
List of SDOs or IDNs
The order (from top to bottom) in the SDO table specifies the order in which the SDOs are transferred to the module.

| "Line" | Line number |
| :--- | :--- |
| "Idn" |  |
| "Bit length" | Bit length of the SDO |
| "Abort on Error" | $\boxed{ }$ : In case of error, the transfer is aborted with an error status. |
| "Jump to Line on Error" | 目: The transfer is resumed with the SDO at the specified "Line" in <br> case of error. |
| "Next Line" | : The transfer is resumed with the SDO at the next line. |
| "Comment" | Input field for comments |
| "Move Up" | Moves the selected line upwards by one line |
| "Move Down" | Moves the selected line downwards by one line |


| "Add" | Opens the "Select Item from Object Directory" dialog. In this dialog <br> you can change the parameters of the SDO before the SDO is <br> added to the configuration. <br> By specifying new index/subindex entries, a new object can be <br> added to the SDO that is not yet described in the EDS file. This is <br> useful if only an incomplete object directory or none at all exists. |
| :--- | :--- |
| "Delete" | Removes the selected entry. |
| "Modify" | Opens the "Select Item from Object Directory" dialog for changing <br> the parameters of the selected SDO or IDN in the table |

## Servodrive over

EtherCAT
Table 362: "Select Item from Object Directory"

| List of available object directory entries as defined in the XML file. |  |
| :--- | :--- |
| Column "Idn" |  |
| Column "Base Value" | Base value of the IDN. <br> Editable (double-click to open) |
| Input fields | $\bullet$ <br> $\bullet$ <br> "S" "P" |
| "IDN" | By specifying new "PSeT"/"Offset" entries, a new object can be added to the <br> IDN that is not yet described in the XML file. This is useful if only an incomplete <br> object directory or none at all exists. |
| "PSet" | By specifying new PSet/Offset-entries, a new object can be added to the IDN <br> that is not yet described in the XML file. This is useful if only an incomplete <br> object directory or none at all exists. |
| "Offset" | List box for selecting the bit length |
| "Bit length" | List box for selecting the value |
| "Value" | If the object has multiple subobjects, then this list box is displayed automatically. |
| "Channel" | Parameters are loaded as a list. The first four bytes indicate the length. <br> "As list" |

## "CAN over

EtherCAT"
Table 363: "Select Item from Object Directory"

| List of available object directory entries as defined in the EDS file. |  |
| :--- | :--- |
| Column "Flags" | Display of access flags: RW (read/write), RO (read only), WO (write only) |
| Column "Base Value" | Editable (double-click to open) |
| Input fields | Input field for displaying and changing the name |
| "Name" | By specifying new index/subindex entries, a new object can be added to the <br> SDO that is not yet described in the EDS file. |
| "Index: 16\#" | Range of values of the object |
| "Subindex: 16\#" | "Bit length" |


| "Value" | Each value may be max. one byte (0-255). It can also be a hexadecimal in IEC <br> syntax (for example, 16\#ad). <br> If the "Byte array" option is enabled, then the values have to be specified as a <br> comma-separated list (for example, 1,2,3,4). |
| :--- | :--- |
| "Full access" | The complete object is written with one access and all subindexes are set at <br> the same time. The time needed for the transfer is reduced because not every <br> subindex has to be transferred individually. |
| "Byte array" | $\boxed{V}$ Values can be specified as a comma-separated byte array. |

See also

- $\boldsymbol{y}^{2}$ Chapter 6.3.2.17.3.1 "Tab 'EtherCAT Slave - General' " on page 1613
- \# Chapter 6.3.2.17 "EtherCAT configurator" on page 1609


### 6.3.2.17.5 Bus Cycle Task - EtherCAT

General infor- The term "bus" includes all fieldbuses as well as the ${ }^{4} / / / O$ bus. Consider that there is no bus mation cycle task for Modbus as Modbus does not provide I/O mapping and is controlled by POUs.
It's recommended to define a dedicated bus cycle task for each fieldbus configured in the project. It's strongly recommended not to use "unspecified" in the ""PLC Settings"" to avoid unexpected behavior. The task defined in "PLC Settings" determines the bus cycle task of I/O bus and, depending on the configuration, of the additional fieldbuses (the setting is by default inherited).

Especially in case of EtherCAT, a dedicated bus cycle task should be used which is not shared with other fieldbuses. If [unspecified] is set in the "PLC Settings", the EtherCAT task might be automatically used by other fieldbuses, potentially causing the EtherCAT task processing to fail. This should be avoided by specifying a task different to the EtherCAT task in the "PLC Settings".
As a rule, for each IEC task the used input data is read at the start of each task and the written output data is transferred to the I/O driver at the end of the task. The implementation in the I/O driver is decisive for further transfer of the I/O data. The implementation is therefore responsible for the timeframe and the specific time when the actual transmission occurs on the respective bus system.

Other tasks copy only the I/O data from an internal buffer that is exchanged only with the physical hardware in the bus cycle task.


```
(1) Read inputs from input buffer (2) IEC task
(3) Write outputs to output buffer
(4) Bus cycle
(5) Input buffer
(6) Output buffer
(7) Copy data to/from bus
(9) Bus cycle task, priority 1, 1 ms
(10) Bus cycle task, priority 5
(11) Bus cycle task, priority 10, interrupted by task 5
```


## Using tasks

The "Task Deployment" provides an overview of used I/O channels, the set bus cycle task, and the usage of channels.

## WARNING!

If an output is written in various tasks, then the status is undefined, as this can be overwritten in each case.

When the same inputs are used in various tasks, the input could change when a task is processed. This happens if the task is interrupted by a task with a higher priority and causes the process map to be read again. Solution: At the beginning of the IEC task, copy the input variables to variables and then work only with the local variables in the rest of the code.

Conclusion: Using the same inputs and outputs in several tasks does not make any sense and can lead to unexpected reactions in some cases.

Behavior of the bus cycle for EtherCAT

Before the IEC inputs are copied, the pending network messages of the last cycle are read.


When the "Messages per task" option is enabled in the settings of the EtherCAT Master, additional telegrams are transmitted to the devices employed per task and input or output employed. Channels that are used in a slow task are also transmitted less frequently. As a result, the bus load can be reduced.


See also

- *s Chapter 6.3.2.17.2.1 "Tab 'EtherCAT Master - General' " on page 1610


### 6.3.2.17.6 Command 'Scan for Devices'

Function: The command establishes a brief connection to the hardware and determines the devices in the network. Then you can apply the devices found into the device tree of your project.
Call: Menu bar: "Project"; context menu of a device object in the device tree
Requirement: The communication settings to the controller are correct. The gateway and the PLC are started. The device supports the scan function.
The following devices provide the scan function: EtherCAT master, EtherNet/IP Scanner (IEC), Sercos master, CANopen Manager, CANopen Manager SIL2, PROFINET controller and PROFIBUS DP master.

You can perform the device scan immediately if the scan function is permanently implemented in the PLC. When scan function is implemented in a library, you have to log in only one time to download the library to the controller.

The command refers to the master controller selected in the device tree. For example, an already inserted PROFINET IO controller can be selected and the command used to determine the I/O devices and I/O modules assigned to it.
After performing the scan operation, the "Scan Devices" dialog opens and displays the found devices.

Dialog 'Scan
Devices'

Table 364: "Scanned Devices"

| "Device name, Device type, | Data about the scanned device depending on network type. |
| :--- | :--- |
| Address, Station name, etc." | When you change a value in the list of scanned devices, the value is shown <br> in italics. This indicates that the new value has been changed in the editor in <br> CODESYS, but not in the device. When you download the value to the device, it <br> is shown normally. <br> Value that indicate differences between the project and the scanned device are <br> shown in orange. <br> If multiple device descriptions are available for the scanned device, then the <br> name is displayed in bold. The selection of the matching device description is <br> resolved differently for different fieldbuses. For more detailed information, see <br> the corresponding fieldbus chapters. <br> If a device description cannot be found, then the following message is shown: |
|  | "Attention! The device was not found in the repository." Depending on the bus <br> system, additional information is displayed, such as manufacturer number and <br> product number. The device cannot be inserted into the project without the <br> installed device description. |
| "Show differences to project" | V: The table in the dialog also shows additional configured devices (in the <br> device tree of the project). <br> $\square:$ The table shows all scanned devices. The configured devices are not shown. |
| "Scan for Devices" | Starts a new search. |

## NO If you with of be

Table 365: "Configured Devices"
This part of the dialog is visible only when you select the "Show differences to project" option.
Differences between the scanned and configured devices are color-coded. Devices displayed in green are identical on both sides. Devices displayed in red are available only in the view of the scanned or configured devices.

|  | If you have selected a device in both views, then the scanned devices are <br> inserted above the selected configured device. |
| :--- | :--- |
| in you have selected a device in both views, then the scanned devices are |  |
| inselow the selected configured device. |  |

### 6.3.2.17.7 Diagnosis

## Diagnosis in the Application

## EtherCAT Status

An instance of data type IoDrvEtherCAT is created for each EtherCAT Master that is inserted into the device tree. The name of the instance corresponds to the name of the master in the device tree. The availability of the instance is displayed on the "IEC Objects" tab of the device editor.

An instance of data type ETCS lave is created for each EtherCAT Slave that is inserted into the device tree. The name of the instance corresponds to the name of the slave in the device tree. The availability of the instance is displayed on the "IEC Objects" tab of the device editor. The slave instance is used in the application to query or change the state of the slave at runtime.

If the device diagnosis is enabled in the PLC settings, then the data type is IoDrvEtherCAT_Diag or ETCSlave_Diag.

See also

- FB IoDrvEtherCAT (FB)
- ETCSlave (FB)
- ② Chapter 6.3.2.17.7.2.9 "Status Page (Master and Slave)" on page 1637


## Emergency

When an EtherCAT device supports CoE (CANopen over EtherCAT), emergency messages at the slave instance can be determined with the following properties:

- LastEmergency: Evaluation of the last received message in the application
- ClearEmergency: Deletion of the last received message and resetting of the property LastEmergency


See also

- ETCSlaveStack.LastEmergency (PROP)
- ETCSlaveStack.ClearEmergency (METH)

Error codes

| Error Code | Description |
| :--- | :--- |
| $16 \# 00 x x$ | Error Reset or No Error |
| $16 \# 10 x x$ | Generic error |
| $16 \# 20 x x$ | Current |
| $16 \# 21 x x$ | Current, device input side |
| $16 \# 22 x x$ | Current, inside the device |
| $16 \# 23 x x$ | Current, device output side |
| $16 \# 30 x x$ | Voltage |
| $16 \# 31 x x$ | Main voltage |
| $16 \# 32 x x$ | Voltage, inside the device |


| Error Code | Description |
| :--- | :--- |
| $16 \# 33 x x$ | Output voltage |
| $16 \# 40 x x$ | Temperature |
| $16 \# 41 x x$ | Ambient temperature |
| $16 \# 42 x x$ | Device temperature |
| $16 \# 50 x x$ | Device hardware |
| $16 \# 60 x x$ | Device software |
| $16 \# 61 x x$ | Internal software |
| $16 \# 62 x x$ | User software |
| $16 \# 63 x x$ | Data set |
| $16 \# 70 x x$ | Additional modules |
| $16 \# 80 x x$ | Monitoring |
| $16 \# 81 x x$ | Communication |
| $16 \# 82 x x$ | Protocol error |
| $16 \# 8210$ | PDO not processed due to length error |
| $16 \# 8220$ | PDO length exceeded |
| $16 \# 90 x x$ | External error |
| $16 \# A 0 x x$ | ESM transition error |
| $16 \# F 0 x x$ | Additional functions |
| $16 \# F F x x$ | Device specific |

Manufacturer-specific error codes need to be checked in the respective documentation from the vendor.

## AL Status

The property ALStatus returns any errors at the start of the stack. For example, configuration errors at the slave instance are displayed.

The AL status messages are also recorded in the logger.

See also

- ETCSlaveStack.ALStatus (PROP)

| Error codes | Value | Error Text |
| :--- | :--- | :--- |
| Comment |  |  |
| $16 \# 0$ | No error |  |
| $16 \# 1$ | Unspecified error |  |
| $16 \# 2$ | No memory |  |
| $16 \# 11$ | Invalid requested state change |  |
| $16 \# 12$ | Unknown requested state |  |


| Value | Error Text | Comment |
| :---: | :---: | :---: |
| 16\#13 | Bootstrap not supported |  |
| 16\#14 | No valid firmware |  |
| 16\#15 | Invalid mailbox configuration | The configuration is not accepted by the device because, for example, data from the ESI file does not match the device. |
| 16\#16 | Invalid mailbox configuration |  |
| 16\#17 | Invalid sync manager configuration |  |
| 16\#18 | No valid inputs available |  |
| 16\#19 | No valid outputs |  |
| 16\#1A | Synchronization error | The DC settings may be incorrect. |
| 16\#1B | Sync manager watchdog | Interruption in the connection, for example due to a brief disconnection of the cable, or the PLC has been stopped. |
| 16\#1C | Invalid sync manager types |  |
| 16\#1D | Invalid output configuration | The configuration is not accepted by the device because, for example, data from the ESI file does not match the device. |
| 16\#1E | Invalid input configuration | The configuration is not accepted by the device because, for example, data from the ESI file does not match the device. |
| 16\#1F | Invalid watchdog configuration |  |
| 16\#20 | Slave needs cold start | Switch device off and back on |
| 16\#21 | Slave needs INIT |  |
| 16\#22 | Slave needs PREOP |  |
| 16\#23 | Slave needs SAFEOP |  |
| 16\#24 | Invalid input mapping |  |
| 16\#25 | Invalid output mapping |  |
| 16\#26 | Inconsistent settings |  |
| 16\#27 | Free-Run not supported | The device has to be configured with "Distributed Clock". |
| 16\#28 | Synchronization not supported | The device does not support "Distributed Clock". |
| 16\#29 | Free-Run needs 3 buffer mode |  |
| 16\#2A | Background watchdog |  |
| 16\#2B | No valid inputs and outputs |  |
| 16\#2C | Fatal Sync error |  |
| 16\#2D | No sync error | The synchronization with DC is not successful, possibly because the jitter of the runtime is too large. |
| 16\#30 | Invalid DC SYNCH configuration |  |
| 16\#31 | Invalid DC latch configuration |  |


| Value | Error Text | Comment |
| :--- | :--- | :--- |
| $16 \# 32$ | PLL error | The slave synchronization is not suc- <br> cessful. |
| $16 \# 33$ | Invalid DC IO error |  |
| $16 \# 34$ | Invalid DC timeout error |  |
| $16 \# 35$ | DC invalid sync cycle time | The configuration of "Distributed Clock" <br> does not match the device. |
| $16 \# 36$ | DC Sync0 cycle time | The configuration of "Distributed Clock" <br> does not match the device. |
| $16 \# 37$ | DC Sync1 cycle time | The configuration of "Distributed Clock" <br> does not match the device. |
| $16 \# 41$ | MBX_AOE |  |
| $16 \# 42$ | MBX_EOE |  |
| $16 \# 43$ | MBX_COE |  |
| $16 \# 44$ | MBX_FOE |  |
| $16 \# 45$ | MBX_SOE |  |
| $16 \# 4 F$ | MBX_VOE |  |
| $16 \# 50$ | EEPROM no access |  |
| $16 \# 51$ | EEPROM error |  |
| $16 \# 60$ | Slave restarted locally |  |

## Diagnosis in the User Interface

## Device Tree

In online mode, the device tree allows for a pinpointing of a pending diagnosis.

- Green symbol: The device is in the OPERATIONAL state.
- Orange triangle: The device is in the PREOPERATIONAL state and is configured, and it is still not in the OPERATIONAL state.
- Error flag (red triangle): Hard error, such as an incorrect/missing device or connection interruption.
- Diagnosis flag (red exclamation mark): Indicates that a diagnosis entry is currently available for exactly this device. The details are then displayed on the status page of the respective device.
- Error-cleared flag (gray exclamation mark): Indicates that a previously pending error has been corrected.
Devices


## EtherCAT - General (Master and Slave)

In online mode, the state of the device is displayed on the "General" tab for the EtherCAT Master and Slave.

In the case of the master, any errors or messages are displayed as text as well as the bus load:

```
Diagnostics message Startup finished: All slaves in operational !
Busload 0 % 
```

In the case of the slave, the current state of the device is displayed.

## Diagnostics

$$
\text { Current State } \quad \text { Operational }
$$

See also

- *s Chapter 6.3.2.17.2.1 "Tab 'EtherCAT Master - General' " on page 1610
- \& Chapter 6.3.2.17.3.1 "Tab 'EtherCAT Slave - General' " on page 1613

Table 366: State of the slave

| Value | Description |
| :--- | :--- |
| Bootstrap | State for firmware downloads |
| Init | Communication is established. |
| Pre-Operational | Mailbox communication is possible, but process data is not <br> transferred |


| Value | Description |
| :--- | :--- |
| Safe Operational | Process data has already been transferred. Inputs are read, <br> but outputs are not given. |
| Operational | Communication has been established, and all process data is <br> transferred. |

## Overview Page in the Master

All slaves and their statuses are displayed together on the "Overview" tab.
Each EtherCAT Slave is listed with its physical address, status, and any error counters. The error counters (CRC) are read from each device for the active connection and also show problems (EMC) with the wiring between the devices.

| Name $\qquad$ AXL F BK EC | Address 1001 | State OP | $\begin{aligned} & \text { CRC } \\ & 0,0 \end{aligned}$ |
| :---: | :---: | :---: | :---: |
| - EK1100 | 1002 | OP | 0,0 |
| [ EL2008 | 1003 | OP | 0,0 |
| EL1008 | 1004 | OP | 0, 0 |
| EL3142 | 1005 | OP | 0,0 |
| [if EL4132 | 1006 | OP | 0 |

## IEC Objects - Master

The driver instance and its inputs and outputs are displayed on the "IEC Objects" tab of the master. You can use the outputs to check whether or not the configuration has completed. In the case of "Distributed Clock", you can see whether or not the controller is synchronized.


## IEC Objects - Slave

The respective slave instance is displayed on the "IEC Objects" tab of the slave. The "wState" output returns the current state of the slave.


## Online Page for the Slave

If the controller is in online mode and the "Expert settings" option is selected, then the additional "Online" tab is displayed. The current status of the device is displayed on this tab.
In addition, the status of the EtherCAT connections are displayed.


File access over EtherCAT
Download... Upload...
$E^{2}$ PROM Access
Write E2PROM... Read E2PROM... Write E2PROM XML

## See also

- ※ Chapter 6.3.2.17.3.7 "Tab 'EtherCAT Slave - Online' " on page 1621


## CoE Online Display for the Slave

If the "Expert settings" option is selected, then the additional "CoE Online" tab is displayed with all CoE objects and their current values.
Here you can find and directly correct an incorrect parameters.


See also

- \& Chapter 6.3.2.17.3.8 "Tab 'EtherCAT Slave - CoE Online' " on page 1622


## Diagnosis History for the Slave

All messages in the respective slave are displayed on the diagnosis history tab. The visibility of this tab depends on the devices used.

| Update History |  |  |  |
| :--- | :--- | :--- | :--- |
| Auto Update |  | Only New Messages | Ack. Messages |
| Type | Flags | Timestamp | Message |
| i Info | N | $16.11 .202011: 38: 25365 \mathrm{~ms}$ | (0) Busfailure gone |

See also

- ${ }^{4}$ ) Chapter 6.3.2.17.3.6 "Tab 'EtherCAT Slave - Diagnostics History' " on page 1621


## Status Page (Master and Slave)

Different error counters are displayed on the "Status" tab of the master. Normally all counters should be 0 .
EtherCAT : Running

Last diagnostic message

| Statistics |  |  |
| :---: | :---: | :---: |
| -. SendFrameCount | 69360 |  |
| - FramesPerSecond | 240 |  |
| - LostFrameCount | 2 |  |
| - TxErrorCount | 0 |  |
| - RxErrorCount | 9 |  |
| - Recv Time (Avg) | LTIME\#22us409ns | Average Time for receiving Ethernet frames per paket |
| - Recv Time (Max) | LTIME\#111us | Max Time for receiving Ethernet frames per paket |
| - Send Time (Avg) | LTIME\#123us 449 ns | Average Time for sending Ethernet frames per paket |
| - Send Time (Max) | LTIME\#202us | Max Time for sending Ethernet frames per paket |

The different times values indicate the time requirement for sending and receiving the messages. As a result, they display the properties of the respective operating system and network driver.

| Value | Description |
| :--- | :--- |
| LostFrameCount | Number of lost packages. This usually indicates connection errors or <br> EMC problems. |
| TxErrorCount | Shows the packages that cannot be sent if for example the internal <br> buffer of the operating system is insufficient |
| RxErrorCount | Lost or late messages are displayed here if for example the operating <br> system is overloaded and the incoming messages are not forwarded <br> to the EtherCAT stack in time. |

A diagnosis message is displayed on the "Status" tab of the slave if errors have occurred. For example, an emergency message is displayed.

| EtherCAT | Running IThe error has been deared. |  |
| :--- | :--- | :--- |
| Last diagnostic message |  | Acknowledge |

Diag String *

See also

- ${ }^{4}$ Chapter 6.3.2.17.7.1.1 "EtherCAT Status" on page 1630

Logger Page (Master and Slave)
Master All messages of the master and all slaves below it are displayed on the "Log" tab of the master.

| ! 0 warning | s) 11 error(s) | (E) 0 | exception(s) | (i) 115 information(s) | D 0 debug message(s) | 亚 |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $\square$ UTC time |  |  |  |  |  |  |
| Severity (3) | Time Stamp 16.11.2020 11:45: | 42.... | Description |  |  |  |
| (3) | 16.11.2020 11:45:42.... |  | All slaves operational |  |  |  |
| (3) | 16.11.2020 11:45:42.... |  | Set operational mode |  |  |  |
| (3) | 16.11.2020 11:45:42.... |  | All slaves safe-operational |  |  |  |
| (3) | 焉,11.2020 11:45:42.... |  | SDO write ok: Address: 1005 Index: $16 \# 1 \mathrm{C} 13$ SubIndex: 0 Data: $16 \# 02$ Re... |  |  |  |
| (3) | 16.11.2020 11:45:42.... |  | SDO write ok: Address: 1005 Index: 16 \#1C13 SubIndex: 2 Data: $16 \# 1$ A01 ... |  |  |  |
| (1) | 16.11.2020 11:45:42.... |  | SDO write ok: Address: 1005 Index: 16 \#1C13 SubIndex: 1 Data: $16 \# 1$ A00 ... |  |  |  |
| (3) | 16.11.2020 11:45:42.... |  | SDO write ok: Address: 1006 Index: 16 \#1C13 SubIndex: 0 Data: 16 \#00 Re... |  |  |  |
| (1) | 16.11.2020 11:45:42.... |  | SDO write ok: Address: 1006 Index: 16 \#1C12 SubIndex: 0 Data: 16 \#02 Re... |  |  |  |

Slave Only the messages from this one slave are displayed on the "Log" tab of the slave.
$\square$ UTC time

| Severity (3) | Time Stamp 16.11.2020 11:45:42.... |
| :---: | :---: |
| (3) | 16.11.2020 11:45:42.... |
| (3) | 16.11. 2020 11:45:42.... |
| (3) | 16.11.2020 11:45:42.... |
| 6 | 16.11.2020 11:45:42.... |
| ${ }^{3}$ | 16.11.2020 11:40:35.... |
| (3) | 16.11.2020 11:40:35.... |
| (1) | 16.11.2020 11:40:35.... |
| (3) | 16.11.2020 11:40:35... |
| (3) | 16.11.2020 11:40:35.... |
| $\bigcirc$ | 16.11.2020 11:40:35.... |
| (3) | 16.11.2020 11:40:14.... |
| (1) | 16.11.2020 11:40:14.... |
| 6 | 16.11.2020 11:40:13.... |
| (3) | 16.11.2020 11:40:13.... |
| (1) | 16.11.2020 11:40:13.... |



### 6.3.2.17.8 <br> Libraries

Only the "EtherCATStack" library is to be used for onboard EtherCAT applications. ② Chapter 6.5.14 "Reference, function blocks" on page 4086

Reference documentation
"CODESYS - Intern $\rightarrow$ IoDrivers $\rightarrow$ EtherCATStack (LIB)"
ⓨ Chapter 6.5.3.3 "View embedded documentation of all libraries" on page 3775

### 6.3.2.18 PROFINET IO Configurator

### 6.3.2.18.1 General

PROFINET IO (Process Field Network) is an industrial Ethernet standard widely used in the field of manufacturing and process automation. It is managed by the user organization PI (PROFIBUS\&PROFINET International) and is considered the successor of PROFIBUS (see https://www.profibus.com/).

### 6.3.2.18.2 PROFINET IO controller

## Controller - General

## Object: PROFINET IO Controller

The PROFINET IO controller, like the slaves, is identified by the station name. For AC500 communication modules, you can also configure the IP settings here. Otherwise the settings apply from the superordinate Ethernet node.

| "Station name" | The station name of the device. It is used for unique identification of the device in <br> the network. |
| :--- | :--- |

Table 367: IP Parameters

| "IP address" | Note: Available for AC500 communication module only. |
| :--- | :--- |
| "Subnet mask" | If you insert the controller below an Ethernet adapter, then you have to define the |
| IP parameters in the dialog of the Ethernet adapter. |  |

Table 368: Default Slave IP Parameter

| "First IP address" | Range of IP addresses that CODESYS uses by default when inserting |
| :--- | :--- |
| "Last IP address" | PROFINET IO devices into the device tree. If you use the "Auto-IP" function |
| in the scan dialog, then IP addresses are also used from this range. The next |  |
| "Subnet mask" | free IP address is selected here. |

Table 369: IO Provider / Consumer Status

| "Application stop --> Substitute <br> values" | When the user stops the application, the provider state is set to "BAD". Then <br> the slaves set the inputs and outputs to predefined substitute values. For more <br> information, see "CODESYS default values - PROFINET IO substitute values" at <br> the end of this chapter. |
| :--- | :--- |
| "Add to I/O mapping" | V: The incoming status information is added to the I/O mapping for all modules; <br> provider state for the input data and consumer state for the output data. |
| "Substitute input data" | - "Zero" <br> - "Last valid value" |

Table 370: Port Data

| "Peer station/port" | Neighboring device with port that is connected to this port. <br> You can accept this setting in the "PROFINET IO Controller Topology" tab. |
| :--- | :--- |
| "Check cable length" | Length of the network cable (in meters)  <br>  $\bullet<10$ <br>  $\bullet<25$ <br>  $\bullet<50$ <br>  $<100$ <br> When the cable length is specified, it is checked when the controller is powered  <br> up. An incorrect cable length causes an error message.  |
| "Check MAU type" | Type of network cable |

Table 371: Watchdog

| "Activate" | Note: Available for AC500 communication module only. <br> 目: If the AC500 communication module firmware is not set within the given time <br> (for example, in the case of an exception error in the application), then it is reset. <br> The connection is terminated and the slaves switch to their substitute values. <br> The defaults for the watchdog originate from the device description. |
| :--- | :--- |

See also

- ⓨ Chapter 6.3.2.18 "PROFINET IO Configurator" on page 1640

CODESYS PROFINET IO devices set their inputs and outputs to predefined substitute values when there
default values - is an interruption. These values are defined in the field device in contrast with default values.
PROFINET IO substitute values These values are usually zero, but specific substitute values can also be configured depending on the device.
The substitute values are set in the following cases:

- The connection is interrupted.
- The controller sets the provider state for the incoming data to "BAD".
- Other interruptions occur (for example, exception in host application, incorrect parameterization)
If the "Application stop --> Substitute values" option is enabled, then the controller sets the provider states to "BAD" at application stop. In this case, the slaves set their substitute values. All incoming data from the controller is ignored (including default values).
If the default values defined in the application should be set for an application stop, then you have to disabled this option. Moreover, you should select the "Update IO while stop" option (in the "PLC Settings"). Otherwise, the CODESYS PROFINET IO controller is stopped.


## PROFINET IO Controller - Bus Cycle Task

General infor- The term "bus" includes all fieldbuses as well as the ${ }^{\leftrightarrows} / / O$ bus. Consider that there is no bus mation cycle task for Modbus as Modbus does not provide I/O mapping and is controlled by POUs.
It's recommended to define a dedicated bus cycle task for each fieldbus configured in the project. It's strongly recommended not to use "unspecified" in the ""PLC Settings"" to avoid unexpected behavior. The task defined in "PLC Settings" determines the bus cycle task of I/O bus and, depending on the configuration, of the additional fieldbuses (the setting is by default inherited).

Especially in case of EtherCAT, a dedicated bus cycle task should be used which is not shared with other fieldbuses. If [unspecified] is set in the "PLC Settings", the EtherCAT task might be automatically used by other fieldbuses, potentially causing the EtherCAT task processing to fail. This should be avoided by specifying a task different to the EtherCAT task in the "PLC Settings".

As a rule, for each IEC task the used input data is read at the start of each task and the written output data is transferred to the I/O driver at the end of the task. The implementation in the I/O driver is decisive for further transfer of the I/O data. The implementation is therefore responsible for the timeframe and the specific time when the actual transmission occurs on the respective bus system.
Other tasks copy only the I/O data from an internal buffer that is exchanged only with the physical hardware in the bus cycle task.

(1) Read inputs from input buffer
(2) IEC task
(3) Write outputs to output buffer
(5) Input buffer
(4) Bus cycle
(7) Copy data to/from bus
(9) Bus cycle task, priority $1,1 \mathrm{~ms}$
(10) Bus cycle task, priority 5
(11) Bus cycle task, priority 10, interrupted by task 5

## Using tasks

The "Task Deployment" provides an overview of used I/O channels, the set bus cycle task, and the usage of channels.

## WARNING!

If an output is written in various tasks, then the status is undefined, as this can be overwritten in each case.
When the same inputs are used in various tasks, the input could change when a task is processed. This happens if the task is interrupted by a task with a higher priority and causes the process map to be read again. Solution: At the beginning of the IEC task, copy the input variables to variables and then work only with the local variables in the rest of the code.

Conclusion: Using the same inputs and outputs in several tasks does not make any sense and can lead to unexpected reactions in some cases.

PROFINET IO bus cycle behavior

PROFINET IO does not provide any additional settings. Its functionality corresponds to the general description.

## PROFINET IO device

## Device - General

Object: PROFINET IO Device

In this dialog, you configure a communication link (PROFINET IO: application relation) to a PROFINET IO Field Device.

For all settings in the present dialog, the device description determines if the values here are editable and the values that are predefined or possible.

| "Station name" | The station name of the device. It is used for unique identification of the device in the network. |
| :---: | :---: |
| "Station status" | 32-bit error code compliant with the PROFINET IO specification. In case of error, the status is provided here, for example, when establishing a connection fails or a link is interrupted. A description is also displayed. |
| "IP Parameters" |  |
| "IP address" | The IP settings of the device. Set when establishing the connection to the controller. |
| "Subnet mask" |  |
| "Default gateway" |  |
|  |  |
| "Communication Settings" |  |
| "Send clock (ms)" | Send clock (in milliseconds). |
| "Reduction ratio" | Scaling factor <br> The send cycle is defined by "Send clock" * "Reduction ratio". Therefore, a "Send clock" of 1 ms and a "Reduction ratio" of 4 means that I/O data is sent every 4 ms . |
| "Phase" | With a "Reduction ratio" of $n$, the send cycle is divided into phases 1 to $n$ (where data is sent in one phase only). You can determine in which phase the data is sent for the purpose of load distribution. <br> If "Send clock" = 1 and "Reduction ratio" = 4 (as in the example above), then you could configure phases 1-4. For four slaves with this send clock and reduction ratio settings, you could assign one of the four phases to each of the four slaves. In this way, only one data packet is sent in each of the four phases of the send cycle and the load is distributed equally. |
| "Watchdog (ms)" | Monitoring time. A multiple of the send cycle (send cycle = "Send Clock" * "Reduction Ratio"). Possible values: $3 \mathrm{~ms}-1920 \mathrm{~ms}$. <br> A connection is terminated when the controller or the PROFINET IO Device does not receive I/O data from the communication peer within this time period. The device enters failure mode and switches the outputs to substitute values. |
| "VLAN ID" | VLAN identifier: Number between 0 and 4095 for VLAN type 802.1Q. <br> Note: For newer devices compliant with PROFINET IOspecification V2.3, only " 0 " is still permitted. |
| "RT class" | If available, you can select the required RT class from the list (real-time communication). |
|  |  |
| "User Parameters" |  |
| "Set All Default Values" | CODESYS resets all settings to default values (see default value column) from the GSDML file. |
| "Read All Values" | CODESYS reads the current values from the device and updates them in the editor. |
| "Write All Values" | CODESYS writes the current values from the editor to the IO device. Not all IO devices support parameter updates in run mode. If not, then an error message is displayed. |

See also

- $\boldsymbol{y y}^{\text {® }}$ Chapter 6.3.2.18 "PROFINET IO Configurator" on page 1640


## PROFINET IO－Module

Module－General
Object：PROFINET IO Module

Table 372：＂Module Information＂

| ＂ID number＂ | Identification of the module（from the device description）． |
| :--- | :--- |
| ＂Slot number＂ | Position of the I／O module below the I／O device，starting at＂1＂for the first <br> module and incremented for each additional module．This results automatically <br> from the current structure in the device tree． |

Table 373：＂User Parameters＂

| ＂Set All Default Values＂ | CODESYS resets all settings to default values（see default value column）from <br> the GSDML file． |
| :--- | :--- |
| ＂Read All Values＂ | CODESYS reads the current values from the device and updates them in the <br> editor． |
| ＂Write All Values＂ | CODESYS writes the current values from the editor to the I／O module．Not all I／O <br> modules support parameter updates in run mode．If not，then an error message <br> is displayed． |

See also
－乡 4 Chapter 6．3．2．18＂PROFINET IO Configurator＂on page 1640

## 6．3．2．18．3 PROFINET IO－Field Device

## General

The configuration of the PROFINET IO field device consists of the device itself as well as the modules inserted below．

CODESYS provides two different PROFINET IO field devices：
－A variant especially for Communication Module CM579－PNIO
－A variant which is hardware－independent，the CODESYS PROFINET IO field device This variant runs on any number of Ethernet adapters and is also available in a purely program－ matically configurable variant．

When inserting the Ethernet－based CODESYS field device，two tasks are created implicitly that are required by the PROFINET IO communication stack．
－＂Profinet＿CommunicationTask＂：This task includes the acyclic communication services，such as establishing connections and diagnostics．These services are not time－critical due to very weak real－time demands．Therefore the task is low priority．
－＂Profinet＿IOTask＂：This is where the actual PROFINET IO real－time data exchange takes place．Pending I／O data packets are processed in each cycle（see Slave Configurator： ＂Send clock＂）．Therefore，a cycle time of 1 ms is required（for 1 ms send clock）．

|  | （6）出 Chapter 6．3．2．18．3．3＂Field Device NetX－ General＂on page 1646 <br> （3）${ }^{\wedge}$ y Chapter 6．3．2．18．2．4．1＂Module－General＂ |
| :---: | :---: |
| $\begin{aligned} & \text { (4) Ethernet (Ethernet) } \\ & \text { (7) } 13 \text { cooesvs, Profnet_Device (Cooesvs Profnet Device) } \end{aligned}$ $\text { (3) Inout Float } 322_{-}^{8} \text { (nnout float } 32^{88} \text { ) }$ | on page 1644 <br> （4）Ethernet adapter <br> （7）出 Chapter 6．3．2．18．3．2＂Field Device－Gen－ eral＂on page 1645 |

For maximum IO performance with minimum delay when reading/writing, I/O data can be updated in this task ( $\rightarrow$ insert own POU that updates IOs in this task). No blocking or persisting operations should be executed in the IO task, such as visualization or file access. If the task is blocked too long, then the watchdog cancels the connection for communicating with the slave (see Slave Configurator: "Watchdog").

## NOTICE!

We recommend that you activate the "Refresh I/Os in Stop" option in the PLC settings. Otherwise the communication is canceled when the application stops at a breakpoint.

See also

- Device Editor Options
- $\Leftrightarrow$ Chapter 6.3.2.18.3.2 "Field Device - General" on page 1645
- $\Leftrightarrow$ Chapter 6.3.2.18.3.3 "Field Device NetX - General" on page 1646


## Field Device - General

Object: PROFINET IO Field Device
The tab displays the basic communication settings.
According to the PROFINET IO standard, the PROFINET IO device is responsible for the IP settings of the used adapter. It has to save remanent IP settings and be able to reset or modify at the request of the controller (IP=0.0.0.0). Initial state (Reset to factory) is with deactivated IP suite ( $\mathrm{IP}=0.0 .0 .0$ ).
To allow this full reset of the IP configuration, some settings have to be done on most systems (see chapter ${ }^{4}$ ) Chapter 6.3.2.18 "PROFINET IO Configurator" on page 1640 ).

However, if the PROFINET IO device is a programmable logic controller and connected with the (CODESYS) programming environment via one and the same Ethernet adapter, then modifying and resetting the IP address is interruptive (connection termination between IDE and PLC). Therefore, one of the modes is provided that deviates from the standard ("Use project parameters').

Table 374: "IP and Name Assignment"

| "Use remanent data" | The IP settings and the station name of the file are used. The file is stored in the <br> file system. <br> The data is set by the controller and saved to a file by the device. |
| :--- | :--- |
| "Use project parameters" | The IP settings and the station name of the project are used (settings of the <br> Ethernet adapter). <br> This option must be selected for Windows, VxWorks, and WinCE, because <br> changing the IP address is not possible for these systems. |
| "Station name" | Station name of the PROFINET IO Device |

Table 375: "IO Provider / Consumer Status"

| "Use incoming" | V: The I/O data for the provider and consumer states is generated which is <br> received by the controller. |
| :--- | :--- |
| "Use outgoing" | V: The I/O data for the provider and consumer states is generated which is sent <br> to the controller. |
| "Substitute values" | The substitute values for the output data become active when the corresponding <br> provider status (Output Data PS) is set to Bad. The output data is sent by the <br> controller and copied to the \%l area of the Profinet modules. <br> The following options are available for the substitute values: |
|  | - "Inactive": The outputs are set to "inactive" (example: 0 ). <br> - "Last value": The output data retains the last valid value (provider status = <br> GOOD). The value is retained even if the connection to the controller has <br> been interrupted. |

In online mode, the station name and the IP settings are displayed "Status" tab.

See also

- ② Chapter 6.3.2.18 "PROFINET IO Configurator" on page 1640

Field Device NetX - General

## Object: PROFINET IO Field Device

The tab displays the basic communication settings.

| "Use remanent data" | The IP settings and the station name of the file are used. Initially, the IP address <br> is 0.0 .0 .0 according to the standard and the station name is blank. When <br> a controller sets these values with the "store remanent" option, then they are <br> stored here. |
| :--- | :--- |
| "Use project parameters" | When starting the device, the values defined in the project for IP configuration <br> and station name are always used initially. |
| "Station name" | The name of the device in the network. <br> Note: The station name and the IP settings can deviate from the (default) set- <br> tings configured in the project. They can be set by the controller in runtime mode <br> and in some cases stored persistently (this means when this is specified for the <br> controller in the Set IP or Set station name commands). After a restart, <br> the device is configured with these values as long as the "Use remanent data" <br> option is set. |
| "IP address" | Initial IP settings <br> Caution: This data can be modified by the PROFINET IO Controller. The rema- <br> nent data is stored on the file system of the controller. After the controller is <br> restarted, this stored data goes into effect. The settings here are then ignored. |
| "Subnet mask" | "Default gateway" |

In online mode, the station name and the IP settings are displayed "Status" tab.

## See also

- $\boldsymbol{y y}^{4}$ Chapter 6.3.2.18 "PROFINET IO Configurator" on page 1640


### 6.3.3 Windows server

Windows server usage in Automation Builder requires modifications of the system environment and of the settings in Automation Builder, e.g. permissions, licensing, session handling etc. How to setup system and software for Windows server is described in the application note Windows Server 2016-RDM access.

### 6.3.4 Protocols and special servers

### 6.3.4.1 IEC60870-5-104 (Telecontrol)

### 6.3.4.1.1 General information IEC 60870

## Introduction

The implemented IEC60870-5-104 protocol allows link-ups between AC500 CPUs with onboard Ethernet and external systems. The link-up takes place via the onboard Ethernet interface of the CPU. The telecontrol protocol according to IEC60870-5 is used.

The CPU can work as both control station and substation. In control direction, setpoints and commands can be set; in monitoring direction the substation sends status values, real values and discrete values to the control station. Via general inquiry, the control station requests the substation to send all status values, real values and discrete values. Otherwise, these values are sent by the substation on a change-driven basis, cyclically or when triggered by an application. Status values, real values and discrete values may contain timestamps. These are filled in with the time of the process station when sent. The CPU can time-synchronize the telecontrol link.

A module accepts the configuration of the physical interface (link layer) and the general protocol parts (application layer).
Send and receive blocks are available for data exchange. These blocks exist for the IEC60870-5 data types setpoint value, command value, double command value, status value, double status value, real value and discrete value. The inputs/outputs of the send and receive blocks are combined with the signals to be communicated. See documentation of IEC60870 library for more information.


The S500 I/O modules do not support the functionality to create and transfer a timestamp with an event directly at the I/O channel. A workaround is using SoE logging (Sequence of Event) for a root-cause analysis. For a better understanding on how events are processed on a AC500 V3 PLC, refer to the application example $\mathbf{A C 5 0 0}$ sequence of events with V3.

Limits of supported devices

AC500 V3 (Standard):

- PM5630: Support of 5 control stations and/or substations with 1.000 information objects overall on ETH1 and ETH2.
- PM5650: Support of 10 control stations and/or substations with 5.000 information objects overall on ETH1 and ETH2.
- PM5670: Support of 20 control stations and/or substations with 10.000 information objects overall on ETH1 and ETH2.

Limits of sup-
AC500-eCo V3:
ported devices

- PM5012/PM5032/PM5052: no support of IEC60870-5-104 protocol.
- PM5072/PM5082: Support of 5 substations with up to 1.000 information objects overall on ETH1 and ETH2. IEC60870 control stations are not supported.


## Data flow control

Each send or receive block can only process one data message. Ideally, new data are available at each user task run-through or new data can be sent.
If the output OV (send block only) indicates TRUE, the function block computes more quickly than the data can be sent. This can happen if the receive block is not computed quickly enough and has thus not collected all the data.

Alternatively, this block sends either cyclically or if the input value is changed. Ideally, the topical data can be sent via the telecontrol link in connection with every user task run-through.

## Data integrity

With IEC60870-5 protocol, a distinction is made between data transmission in the monitoring direction (status values, real values, discrete values) and in the control direction (commands and setpoints).
All data transmissions are acknowledged from the link communication level by the receiver. This acknowledgement is not sent to the sender of the data in every telecontrol link.

For data transmission in control direction, additional acknowledgement (e.g. ACTTERM) is possible. These acknowledgements are not sent by every telecontrol link either. For safe data transmission, it is necessary, in such cases, to configure data readback. The receiver then sends the data received back to the sender via the corresponding send blocks.
Information in the monitoring direction is acknowledged by the receiver on the lowest communication level (link level) when received. This acknowledgement is generated by the telecontrol head itself with some telecontrol heads. In the event of overload/overrun, a data message may be lost. For data in the control direction, so-called ACTTERM acknowledgement can be used. This additional acknowledgement is sent back to the sender when the data have been executed in the process. If data are to be sent in the monitoring direction with guaranteed transmission, it is necessary to read back the sent value via another variable and, after observing a monitoring time, resend in the event of an error.

## Data transmission

## Send blocks

On the basis of the communication protocol, it is sensible to restrict the data types at one send block to one type. Therefore, there are 5 types of send blocks: send of status values, commands, real values, setpoints and discrete values. These types are mapped to the IEC1131 data types BOOL, REAL and DINT. See documentation of IEC60870 Library for more information.

Operating modes of the send blocks
The send blocks know three operating modes to send their data:

- Caused by request pin (SEND)
- $\quad$ Send in connection with a change of data (AUTO)
- Cyclic send of data (CYCLE)


## Send via request pin

The SEND signal is evaluated on the rising edge, the RDY signal remains applied for one computation cycle. If a rising edge is generated again at the SEND signal although no acknowledgement has yet been received from the receiver, the OV pin is set in order to indicate that an overrun has happened. The evaluation of the receive acknowledgement is carried out before the evaluation of whether transmission is to take place. This means, assuming that there is an appropriately fast telecontrol link, that in connection with change-driven and cyclic transmission, a transmission job can be sent in connection with every computation of the block. In connection with send via the request pin it is possible to send only in connection with every second computation (send takes place only with a rising edge).

## Change-driven send of data

Data are always sent when the value of the input variables changes. When changes take place, there is an internal simulation that the SEND pin changed from 0 to 1.
In order to prevent unnecessarily frequent send in the event of mild fluctuations in the input value, a threshold value can be configured for real values and setpoints. The input value is not sent until it differs positively or negatively from the value last sent by more than the threshold value.

If the input value changes again although no acknowledgement has yet been received from the receiver, the OV pin is set in exactly the same way as in connection with send via the request pin. If an error occurs during send, the job is automatically retried until the value has been sent without error.

## Cyclic send

The data are automatically sent after expiration of a configurable cycle time (SCANDOWN). This cycle time is indicated in multiples of the task cycle time in which the block is computed. In this operating mode, an overrun error can occur if the transmission is faster than the response time of the receiver. For setpoints, it is necessary to ensure that an acknowledgement is generated by the receiver which is not sent until the setpoint is accepted. The send block is not ready for transmission again until after this acknowledgement has been received.

## Receive blocks

In receive direction, the jobs enter the device module via the interface. The device module selects the correct receive block using the telecontrol address. To this end, during installation the receive blocks pass their parameterized telecontrol addresses to the device module. The device module stores the data received and the receive blocks make the data available at their output pins in connection with the next computation of the user task.

### 6.3.4.1.2 Configuration

## Configuration changes >= Automation Builder 1.1/CBP 2.4

The IEC 60870 protocol allows link-ups between AC500 CPUs with onboard Ethernet (e.g. PM595-4ETH and PM591-2ETH) and external systems.

The link-up takes place via the onboard Ethernet interface of the CPU. As of Automation Builder Version 1.1 telecontrol is also supported for CPUs that provide more than one Ethernet interface (e.g. PM595-4ETH and PM591-2ETH). This allows to use different Ethernet interfaces for IEC 60870 connections, hence, telecontrol configuration is changed. Further, as of this version terminology is aligned with IEC 60870 standard and provides additional features that are described in this chapter. For a description on principle telecontrol configuration.


For further information on configuration changes, see the following chapters:
Control and Substations $\geq$ CBP 2.4 \& Chapter 6.3.4.1.2.2 "Control station and substation configuration" on page 1650

Import Export $\geq$ CBP $2.4 \Leftrightarrow$ Chapter 6.3.4.1.2.1 "Configuration changes $>=$ Automation Builder 1.1/CBP 2.4 " on page 1649

Validity Check of Configuration $\geq$ CBP 2.4 * $\stackrel{\text { Chapter 6.3.4.1.2.4 "Validity check of configura- }}{ }$ tion" on page 1670

## Control station and substation configuration

## General

The CPU can work as both, control station and substation.

| Control station | Client, master, controlling station: Synonyms for a higher-level sta- <br> tion (central station, monitors other stations) |
| :--- | :--- |
| Control direction | Data transfer direction from the control station to the substation |
| Substation | Server, slave, controlled station: synonyms for a subordinate IEC <br> $60870-5-104$ <br> telecontrol station (which is monitored) |
| Monitoring direction | Data transfer direction from the substation to the controlling station |

Configure a control station in the device tree "PLC"--> "Interfaces" --> "Ethernet" --> "Protocols":

1. Right-click "ETHx $\rightarrow$ Add objects".
2. Select the control station from the list and click "Add object". Configure substations and further control stations in the same way. As of Automation Builder 1.1 any combination of control stations and substations can be configured, in due consideration of a total number of 10 stations.
3. Double-click the "new control station" node to open parameter configuration. In the "Link Layer" tab access to the Ethernet interface is configured.

As of Automation Builder 2.2.1 the V3 PLC telecontrol station objects from GVL IEC60870_5_104_Connection_GVL can only be used in Structured Text by adding the namespace of the GVL as prefix.
Example:
byteValue := IEC60870_5_104_Connection_GVL.IEC_60870_5_104_Controlstation.Con;

## "Link layer" tab



In order to provide flexible usage of control stations and substations as of Automation Builder 1.1 configuration of substations has been changed. As several substations can be operated with several Ethernet interfaces, select the Ethernet interface to be used from the pull-down menu. Enter the IP address to the control station and if required to another control station (redundant connection). If no IP address is defined, the substation accepts connection to any control station.

This field is not available in the "Link Layer" tab of control stations. Selection of ETH interface is only possible for substations. The control station is always configured on both interfaces by default.

Timeout set- $\quad$ T1, T2, T3: The values for the connection control and message replication; timeout $1 / 2 / 3$.

## tings

Buffer settings This parameter gives the maximum number of outstanding messages and acknowledgement behavior.

Send buffer (k): Maximum difference receive sequence number to send state variable.
Rec buffer (w): Latest acknowledge after receiving w I format APDUs.

## Network settings

Network settings are available for control stations and for substations. The IP address of the control station and if available the IP address of another control station (redundant IP address) can be selected by the user.

For an overview on the configured Ethernet interfaces for the control stations and substations, double-click the "Protocols" node.

## "Application layer" tab

## Settings

The application layer is the communication layer with which the send and receive blocks work.

```
Link Layer Application Layer Information objects
    General
        V Use ACTTERM
        \square \text { Foreignacknowledge}
        Applicationtimeout: }\quad3000\textrm{ms
        Station address: 255 - 255
        \squareimesync
        \ Send 'Init end' after reconnection
        General inquiry
        \square \mp@code { A c t i v a t e d }
        With parameters
        | Without integrated totals
        Counter interrogation
        Group General *
        With Reset
        With Store
```

Use ACTTERM This parameter concerns only setpoints and commands. If this parameter is checked, an acknowledgement with set 'actterm' is generated as reason for transmission at the time at which the receive block is computed and outputs its telecontrol data at its output pins. On transmission side, the data block awaits the reception of this ACTTERM acknowledgement and reacts with its corresponding output to the reception of this acknowledgement. For commands with execution time, the acknowledgement is generated when the command is terminated, for commands with continuous execution time and for setpoints, the acknowledgement is generated when the data are output to the output pins.

## ForeignAcknowledge

If this option is not enabled (default), a message that was sent is considered as ok as soon as transmission was successful. If you enable this option, a message that was sent is not considered as ok until a success message (foreign acknowledge) is returned from the receiver.

Application This time indicates how long an acknowledgement will be awaited on the application level. An timeout acknowledgement is generated only for commands and setpoints on the application level.

Station address The station address defines which station will be subject to a count query. The values define the 2 bytes for the common telecontrol address (Common addr.). The values concerned are as follows:

0 : The station address is not used.
1 ... 254: The count is queried on the station defined by the station address.
255: The count is queried on all accessible stations.

| Timesync | After each new establishment of a link and once per hour, a 'coarse time synchronisation' |
| :--- | :--- |
| message is generated. This time synchronisation is only supported from AC500 to external sys- |  |
| tems. Time synchronisation from an external system to AC500 is not provided! Incoming time |  |
| synchronisation messages are confirmed by the process station but not executed. Greenwich |  |
| Mean Time (GMT) is used as the time for the synchronisation. |  |

Send 'Init end' After each establishment of a link or only in connection with the first establishment of a link and after reconnection after reconfiguration, an init end message is generated. After the init end message, there is a general inquiry, if configured.

## General inquiry

Activated
This parameter concerns only real values, discrete values and status values. The device module generates a general inquiry message after each new establishment of a link. The other side then generates a message with the reason for transmission 'general inquiry' for every data point and subsequently an init end message. This procedure ensures that, in the event of a new establishment of a link, all data are available on the reception side in topical form.

With parameters If general inquiry is activated the parameter values are sent.

Without inte- With a general inquiry no integrated total values are sent.
grated totals

## Counter interrogation

Group General, $1 \ldots 4$ : The count inquiry is executed for a specific group of counters (1 ... 4). The count inquiry is executed for all groups of counters.

With reset The reset quality bit is sent along with the count inquiry.

With relocate The relocate quality bit is sent along with the count inquiry.

## "Information objects" tab

General
Open the "Information object" tab to configure so called information objects and a common address (known as 'data points' and 'Global address' in former Automation Builder versions). In this tab different information objects and their services for transmission are defined. A data point or information object is identified via a system-wide unambiguous address containing a maximum 5 bytes.

1. Right-click in the empty view and select "Add Information Object with ASDU" to add a data group. Select the desired object from the list (e.g. M_SP_NA_1).
$\Rightarrow$ An information object with a corresponding ASDU (Application Service Data Unit) is created.
2. Configure the settings in the "Information Object" tab to your convenience.

| Link Layer | Application Layer Infor | Information objects Control station Configuration |  |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Information Objects |  |  |  |  |  |  |  |  |  |  |
| ASDU na |  | Data type |  | ASDU type | Common addr | Info obj addr | Norm start | Norm end | Threshold | Description |
| Single | C_point_information_12 | IEC60870_S | Inglepointinformation | (1) M_SP_Na_1 | 1.1 | 0.2.200 |  |  |  | Infoobl |
| Doub | e_point_jinformation_15 | IEC60870_D | OublePointInformation | (3) M_DP_NA_1 | 1.1 | 0.2 .201 |  |  |  | Infoobj |
| Meas | ured_value_30 | IEC60870_M | easuredValue | (36) M_ME_TF-1 | 1.1 | 0.2.202 | -1000.0 | 1000.0 | 0.0 | Infoobs |
| Set_p | oint_command_25 | IEC60870_S | etPoint | (50) C_SE_NC_1 | 1.1 | 0.3.0 | -1000.0 | 1000.0 | 0.2 | InfoObj |
| Meas | ured_value_20 | IEC60870_M | easuredValue | (13) M_ME_NC_1 | 1.1 | 0.3.1 | -1000.0 | 1000.0 | 0.1 | Infoobj |
| Single | _command_35 | IEC60870_S | ingleCommand | (45) C_SC_NA_1 | 1.0 | 0.3.02 |  |  |  | Infoobs |
| Set_p | oint_command_40 | IEC60870_S | etPoint | (63) C_SE_TC_1 | 1.0 | 0.3 .01 | -1000.0 | 1000.0 | 0.0 | Infoobs |

3. Double-click a table cell to modify pre-set values. For some ASDUs additional sub information objects can be configured. For this, right-click the already existing ASDU and select "Add Information Object" to selected ASDU option. This allows configuration of 16 data points at the most (depending on the ASDU type). With "Remove Information Object" the selected ASDU is deleted.

## Description of

- ASDU name: node name of the information object (name of the ASDU) the columns
- Data type: Data type of the ASDU.
- ASDU type: Type of ASDU.
- Common addr: Common address of the ASDU (known as 'Global Address' in former AB Versions). Byte $1 / 2$ of the common telecontrol address of the block (range: 0 ... 255).
- Info obj addr: Together with common address Info obj addr defines the endpoint (range: 0 ... 255).
- Norm start: Low limit (0\%) of the normalized range for real values and setpoints.
- Norm end: High limit ( $100 \%$ ) of the normalized range for real values and setpoints.
- Threshold: Threshold limit beyond which a change of the input value referred to.
- Description: Table cell for free text. Use this field to describe your configuration settings e.g. differences between configuration variants.


## Format of common addr and info obj addr

The following address formats of your entries in the columns Common addr and Info obj addr of the "Information Objects" tab are possible:

- 1.2 and 3.4.5 (Default format)
- 1-2 and 3-4-5
- 258 or hex $0 \times 102$ and 197637 or hex $0 \times 30405$
- 513 or hex $0 \times 201$ and 328707 or hex $0 \times 50403$

Previously you have to choose your preferred address format:

| Iools | Window Help |
| :---: | :---: |
| 줄 | Create Device list CSSV |
|  | Instal additonal license |
|  | Installation Manager |
|  | IP-Configuration |
|  | MultionlineChange |
|  | Drive Overview |
|  | Install application parameters |
| (1itil | Library Repository... |
| 11 | Device Repository... |
|  | Open Device Type Editor |
|  | Device ECAD data |
| - | License Manager... |
| 2 2 | Visualization Styles Repository... |
|  | Scripting |
|  | Options... |

1. Click "Tools"and then "Options..."
$\Rightarrow$ The window "Options" appears

2. Select "IEC 60870-5-104", make your choice and click [OK].

## Import options of information objects

The User can accept the imported IEC60870 information objects as single change or change as block.










## IEC60870-5-104 Multiple connections

## General

An AC500 with more than one substation connection must be able to identify the corresponding control station clearly. This identification takes place exclusively via the control station's IP address. In order to make it possible for a non-redundant control station to have redundant access to a substation with 2 Ethernet connections. The local substation address is ignored during connection establishment.

In the following descriptions, the term station must not be confused with the individual connection. One station can have several connections. An IEC60870-5-104 communication always takes place between a control station and a substation. A control station can manage several substations and also simultaneously be a substation for one or several control stations. However, these must then be realized using different stations.

A PLC may not be configured for another PLC repeatedly as a substation or a control station unless a disjunctive Ethernet infrastructure is used for this.
Redundant connections must be specified as such in the configuration.
An AC500 can be used only once as control station for another AC500, it makes no sense to use the same AC500 repeatedly as a control station for the same substation. Such a structure is configured as a redundant control station as long as only one AC500 exists as a control station per substation. However, this control station may have 2 IP addresses. Therefore, this configuration must either have the IP address 0.0.0.0 entered on the substation for the control station, meaning that all IP addresses are accepted and no other control station can access this AC500 or alternatively the possible control station addresses must be specified (ETH1 and ETH2).

Tree constella- PM57x-/ PM58x-/ PM59x-ETH, PM5650-2ETH:
If you plan to control several substations with the AC500, they can be cascaded. This results in a tree structure.

(1) 2 control stations
(2) Substation and 3 control stations
(3) Substation and 3 control stations
(4) Substation
(5) Substation

In the following, the notation 192.168.1.0/24 is used for TCP/IP networks. Here, the figure /24 specifies the network mask with 255.255.255.0 and 192.168.1.0 describes the network. The valid addresses for this Class C network are 192.168.1.1 to 192.168.1.254! Only the last byte of the address is provided on the respective devices, with e.g. .10. This means that the respective device has the address 192.168.1.10.

## Minimal structure

A control station with an Ethernet interface is connected to a substation with an Ethernet interface.

(1) Control station
(2) Substation

Configuration at PM57x-/ PM58x-/ PM59x-ETH, PM5650-2ETH:
control station
The respective substation IP address must be specified at the control station. For this, in the network settings of the control station (1) enter the IP address of the substation (in the example: 192.168.1.25). Option "Enable redundant connection" must be disabled.

Configuration at PM57x-/ PM58x-/ PM59x-ETH, PM5650-2ETH:

## substation

Either the control station IP address or the general address 0.0 .0 .0 must be specified at the substation (2). For this, in the network settings of the substation enter the IP address of the control station (in the example: 192.168.1.10). Option "Enable redundant connection" must be disabled.

If the general address 0.0.0.0 is used at the substation, no further control station can be configured on this controller for a further substation.

## Minimal redundancy structure

The most simple redundant structure with an AC500 consists of a redundant control station (not AC500) which is connected to the AC500 substation with 2 different IP addresses. These redundant control stations must synchronize which control station is active.

Only one control station can be active at any given time.

(1) Control station $1 \mathrm{~A}($ Not AC 500$)$
(2) Control station 1B (Not AC500)
(3) Substation
(4) Redundancy link

Configuration at The respective substation IP address must be specified at the control stations 1 and 2 (not control stations AC500). For this, in the network settings of both control stations enter the IP address of the substation (in the example: 192.168.1.25).

## Configuration at PM57x-/ PM58x-/ PM59x-ETH, PM5650-2ETH:

Either the control station IP addresses or the general address 0.0 .0 .0 must be specified at the substation (3). For this, in the network settings of the substation enter the IP addresses of the control station (in the example: 192.168.1.10 and 192.168.1.11). Option "Enable redundant connection" must be enabled.


If the general address 0.0 .0 .0 is used at the substation, no further control station can be configured on this controller for a further substation.

## Network redundancy

## General

For network redundancy a control station can reach a substation via 2 paths.
Both the control station and the substation can have 2 different IP addresses. Without special network routing, 2 separate networks should exist, within which both the substation and the control station each have 2 interfaces.

Possible variants of network redundancy are described in the following.

## Network redundancy with 2 separate networks


(1) Control station with 2 redundant paths
(2) 1 Substation with 2 Ethernet interfaces

Configuration at PM591-2ETH, PM595-4ETH, PM5650-2ETH:
control stations
The substation's IP addresses must be specified at the control stations (1). For this, in the network settings of the control station enter the IP addresses of the substation (in the example: 192.168.1.25 and 192.168.2.26). Option "Enable redundant connection" must be enabled.

Configuration at PM591-2ETH, PM595-4ETH, PM5650-2ETH:

## substation

Either the control station's IP addresses or the general address 0.0 .0 . 0 must be specified at the substation (2). For this, in the network settings of the substation enter the IP addresses of the control station (in the example: 192.168.1.10 and 192.168.2.11). Option "Enable redundant connection" must be enabled.

If the general address 0.0 .0 .0 is used at the substation, no further control station on another substation can be configured on this controller. Equally, the substation connection must be activated for both interfaces.

## Network redundancy with 1 network and 2 Ethernet ports in substation


(1) Control station with 2 paths to reach substation
(2) 1 Substation with 2 Ethernet interfaces

## Configuration at <br> control stations

PM591-2ETH, PM595-4ETH, PM5650-2ETH:
The substation's IP addresses must be specified at the control stations (1). For this, in the network settings of the control station enter the IP addresses of the substation (in the example: 192.168.1.25 and 192.168.1.26). Option "Enable redundant connection" must be enabled.

Configuration at PM591-2ETH, PM595-4ETH, PM5650-2ETH:

## substation

Either the control station's IP addresses or the general address 0.0 .0 .0 must be specified at the substation (2). For this, in the network settings of the substation enter the IP addresses of the control station (in the example: 192.168.1.10 and 192.168.2.11). Option "Enable redundant connection" must be enabled.

If the general address 0.0 .0 .0 is used at the substation, no further control station on another substation can be configured on this controller. Equally, the substation connection must be activated for both interfaces.

## Network redundancy with 1 network and 1 Ethernet port in substation

No online redundancy.
Only one connection will be established.

(1) Control station with 2 paths to reach substation
(2) 1 Substation with 1 Ethernet interface

Configuration at PM591-2ETH, PM595-4ETH, PM5650-2ETH:
control stations The substation's IP addresses must be specified at the control stations (1). For this, in the network settings of the control station enter the IP addresses of the substation (in the example: 192.168.1.25 and 0.0.0.0). Option "Enable redundant connection" must be disabled.

Configuration at PM591-2ETH, PM595-4ETH, PM5650-2ETH:
substation
Either the control station's IP addresses or the general address 0.0 .0 .0 must be specified at the substation (2). For this, in the network settings of the substation enter the IP addresses of the control station (in the example: 192.168.1.10 and 192.168.2.11). Option "Enable redundant connection" must be enabled.

If the general address 0.0 .0 .0 is used at the substation, no further control station on another substation can be configured on this controller. Equally, the substation connection must be activated for both interfaces.

## Network redundancy with 2 Ethernet ports in substation


(1) Control station with 2 paths to reach substation
(2) 1 Substation with 2 Ethernet interfaces

Configuration at PM591-2ETH, PM595-4ETH, PM5650-2ETH:
The substation's IP addresses must be specified at the control stations (1). For this, in the network settings of the control station enter the IP addresses of the substation (in the example: 192.168.1.25 and 192.168.1.26). Option "Enable redundant connection" must be enabled.

## Configuration at PM591-2ETH, PM595-4ETH, PM5650-2ETH:

substation
Either the control station's IP addresses or the general address 0.0 .0.0 must be specified at the substation (2). For this, in the network settings of the substation enter the IP addresses of the control station (in the example: 192.168.1.11 and 0.0.0.0). Option "Enable redundant connection" must be disabled.


If the general address 0.0.0.0 is used at the substation, no further control station on another substation can be configured on this controller. Equally, the substation connection must be activated for both interfaces.

## Full control station redundancy

A control stationcan consist of two fully redundant units (not AC500s), which are connected via a redundancy link. These control stations must ensure that only one of them at a time is actively connected to the substation and communicates with it. The inactive control station, however, can establish non-active connection with a substation and monitor it with keep alive packages.

(1) 2 redundant Control stations (Not AC500)
(2) 1 Substation with redundant Control station and 2 Ethernet interfaces (2 ${ }^{\text {nd }}$ port)
(3) Redundancy link

Configuration at The substation's IP address must be specified at the control stations (1) (not AC500). For this, control stations in the network settings of the control station enter the IP addresses of the substation (in the example: 192.168.1.25 and 192.168.2.26).

Configuration at PM591-2ETH, PM595-4ETH, PM5650-2ETH:
substation Either the control station's IP addresses or the general address 0.0 .0 .0 must be specified at the substation (2). For this, in the network settings of the substation enter the IP addresses of the control station (in the example: 192.168.1.10 and 192.168.2.11). Option "Enable redundant connection" must be enabled.


If the general address 0.0 .0 .0 is used at the substation, no further control station on another substation can be configured on this controller. Equally, the substation connection must be activated for both interfaces.

## Multiple control stations on the same network

As of firmware version 2.4, an AC500 can be used as a substation for several control stations. For this, the control stations must be distinguished by their IP addresses. Should a control station have more than one IP address (redundancy), both possible IP addresses should also be entered for the allocated substation connection. As a result, even despite being equipped with several Ethernet interfaces, a device can only be one allocated control station at a time for a determined substation. Thus, several substations can be configured for different control stations on a AC500.

(1) Control station 1
(2) Control station 2
(3) 2 Substations (IEC60870-5-104 $2^{\text {nd }}$ Connection)

## Configuration at PM57x-/ PM58x-/ PM59x-ETH, PM5650-2ETH:

The substation's IP address must be specified at the control stations. For this, in the network settings of the control station (1 and 2) enter the IP addresses of the substation (in the example: 192.168.1.25). Option "Enable redundant connection" must be disabled.

Configuration at PM57x-/ PM58x-/ PM59x-ETH, PM5650-2ETH: 192.168.1.10 and 192.168.1.11). Option "Enable redundant connection" must be disabled.

## Multiple control stations on different networks

As of firmware version 2.4, an AC500 can have several local Ethernet interfaces which can be used for separate control station connections. For this, a control station must be identified via its IP address. The substation address used locally is not used to distinguish a connection in order to enable a network and therefore route redundancy. On AC500, the acception of IEC60870-5-104 connections on an interface can only be prevented.

(1) Control station 1
(2) Control station 2
(3) 2 Substations with 2 Ethernet interfaces ( $2^{\text {nd }}$ port and $2^{\text {nd }}$ connection)

## Configuration at <br> control stations

PM57x-/ PM58x-/ PM59x-ETH, PM5650-2ETH:
The substation's IP addresses must be specified at the control stations (1 and 2). For this, in the network settings of the control station enter the IP addresses of the substation (in the example: 192.168.1.25 and 192.168.2.26). Option "Enable redundant connection" must be disabled.

## Configuration a substations

Both control station's IP addresses must be specified at the substation (3) under both substation connections. For this, in the network settings of the substation enter the IP addresses of the control stations (in the example: 192.168.1.10 and 192.168.2.11). Option "Enable redundant connection" must be disabled.

## Double connection



## This configuration does work.

But it is senseless!

It is possible to configure a double connection between 2 stations using 2 separate networks (at least logically separated sub-networks).
However, such a setup has no advantages via-à-vis the minimal structure right at the start $\boldsymbol{y}$ Chapter 6.3.4.1.2.2.7.2 "Minimal structure" on page 1658.
For this setup, connection data must be double configured and double resources are also required at the stations, not providing any advantages whatsoever.
Rather the opposite is true, because such configurations are highly prone to errors.

(1) 2 Control stations with 2 Ethernet interfaces
(2) 2 Substations with 2 Ethernet interfaces


## This configuration does not work!

If an AC500 is configured as a control station, the interface which is used to reach the substation is not defined.

The decision as to which interface is used for this is taken by TCP/IP when running.
It is also dependent on the current network configuration.
Here, the current link status and the order of link recognition may be decisive for the interface to be used.

Such a scenario would not result in stable communication as both substations cannot clearly distinguish the control stations.

Instead, the connection management for a substation will assume that the control station has lost the connection and then establishes a connection.

(1) 2 Control stations with 2 Ethernet interfaces
(2) 2 Substations with 2 Ethernet interfaces

## Export a CSV file

As an alternative many values can be modified at a time by exporting the configuration to a CSV file. After modifying the file data, import the CSV file.
« ${ }^{〔}$ Chapter 6.3.4.1.2.3 "Import/Export functionality" on page 1669

## Import/Export functionality

As of Automation Builder 1.1 (CBP >= 2.4) configuration of control stations and substations can be exported/imported via CSV file. Open the CSV file with a spreadsheet software (e.g. Microsoft Excel) and modify the values within the file to your convenience:

1. Export configuration data: right-click the node of the control station or substation to be exported.
```
-Q Interfaces
    -\ COM1_Online_Access (COM1 - Online Access)
    \[] COM2_Online_Access (COM2 - Online Access)
    - 5%My Ethernet
        [5] ETH1 (ETH1)
        (5) ETH2 (ETH2)
        -(5) Protocols (Protocols)
            ## IEC_60870_5_104_Controlstation (IEC 60870-5-104 Controli
            #1/ IEC_60870_5_104_Substation(IEC 60870 Copy
    = 5TM Fieldbus_on_Ethernet
        (7) ETH3 (ETH3)
            缃) PNIO_Controler (PROFINET-IO-Control
        [5 ETH4 (ETH4)
            }囯 PNIO_Controler_1 (PROFINET-IO-Contr Add object
- [3) Extension_Bus
    & Slot_1 (TA524)
    -} Slot_2(TA524)
```



- Data in black color means the existing data and the data to be imported is identical.
- Data in red color means the existing data and the data to be imported differ. Decide whether to import the new data (and to overwrite the existing data) or not.
- Data in blue color means, the data to be imported is new and will be added to the existing data.
- Data that has been confirmed for the import already is displayed in green color (after clicking [Accept Single]).

In order to move data from one side of the window to another, select the data and click [Accept Single]. Data is highlighted in yellow.


To confirm the import of all new data, click the top entry (here: All: ASDU name - ASDU tpye Common addr - ...). Then, click [Accept Single].
Close the "Project Compare - Differences" tab, save your project and confirm the message. The changes are displayed in the "Information objects" tab.


## Validity check of configuration

We recommend you to verify the IEC configuration of control stations and substations: Rightclick a control station or substation -> "Check configuration".


The check will look for the following topics:

- Duplicate addresses.
- Stations without any Information objects.
- ASDU names, which are not unique.

When a check finds errors or incompatibilities this will be reported in a separate messages view at the buttom of the window:

```
Messages - Totally 62 error(s), 0 warning(s), 0 message(s)
All messages * (% 31 error(s)
Description
() Address is not unique: Measured_value_26 '0.0.0.0.0'.
(4ddress is not unique: Single_point_information_21 '0.0.0.0.0'.
(%) Address is not unique: Set_point_command_31 '0.0.0.0.0'.
(4ddress is not unique: Measured_value_26 '0.0.0.0.0'.
(3) Address is not unique: Set_point_command_31 '0.0.0.0.0'.
* A common address definition is missing for: IEC_60870_5_104_Substation
```

Precompile: $\downarrow$ OK

With a double-click on the error line, the part of the configuration with the violation will be opened. Now, you can correct the error.

### 6.3.4.1.3 IEC60870 compatibility list

In order to implement specific telecontrol systems, this companion standard presents sets of parameters and alternatives from which subsets must be selected.
AC500 - IEC60870-5-104 Protocol Interoperability / Compatibility List

### 6.3.4.2 IEC 61850 Server

### 6.3.4.2.1 Introduction to IEC 61850

The IEC 61850 communication protocol standard for electrical substations enables the integration of all protection, control, measurement, and monitoring functions by one common protocol. It provides the means of high-speed substation applications, station wide interlocking and other functions requiring intercommunication between IEDs = blue boxes in the following figure.


The well described data modelling, the specified communication services for the most recent tasks in a station makes the standard to a key element in modern substation systems.

This manual describes mainly how the IEC 61850 standard is applied in AC500. References and brief descriptions of the IEC 61850 standard are also included in this introduction chapter. Anyhow, it is assumed that the reader has basic knowledge about the IEC 61850.

To understand the IEC 61850 standard and to be able to find the related information, the following parts of the standard are important:


Part 5: Communication requirements for functions and device models

Part 7-4: Compatible logical node classes and data classes

IEC 61850-5 explains how substation functions are realized by logical nodes = function blocks which are running in the IEDs. Operating an isolator from station level for example requires the logical nodes: "IHMI", "CSWI", "CILO" and "XSWI".

The logical nodes and their data objects are defined in IEC 61850-7-4, for example "CSWI", which is an abbreviation for "Switch controller".

| CSWI class |  |  |  |
| :--- | :--- | :--- | :---: |
| Data object name | Common data class | Explanation |  |
| Status information |  |  |  |
| LocKey | SPS | Local or remote key |  |
| Loc | ACT | Local control behavior |  |
| OpOpn | SPS | Operation "Open switch" |  |
| SelOpn | ACT | Selection "Close switch" |  |
| OpCIs | SPS | Operation "Open switch" |  |
| SeLCls | INC | Selection "Close switch" |  |
| Controls | SPC | Resettable operation counter |  |
| OpCntRs | DPC | Switching authority at station level |  |
| LocSta | DPC | Switch, general |  |
| Pos | DPC | Switch L1 |  |
| PosA | DPC | Switch L2 |  |
| PosB | Switch L3 |  |  |
| PosC |  |  |  |

Part 7-3:
Common data classes

Data objects are based on common data classes, which are defined in IEC 61850-7-3, for example the data object "Pos" (Position) of "CSWI" is based on data class "DPC" (Double Point Controllable).

| DPC class |  |  |  |  |
| :--- | :--- | :--- | :--- | :--- |
| Attribute name | Attribute type | FC | TrgOp | Value / Value range |
|  |  |  |  |  |
| Control and status | BOOLEAN | CO |  | off (FALSE), on (TRUE) |
| ctIVal | TimeStamp | CO |  |  |
| operTm | Originator | CO, ST |  |  |
| origin | INT8U | CO, ST |  | $0 . . .255$ |
| ctINum | CODED ENUM | ST | dchg | Intermediate-state, off, on, bad- <br> state |
| stVal | Quality | ST | dchg |  |
| q | TimeStamp | ST |  |  |
| t | BOOLEAN | ST | dchg |  |
| stSeld | Configuration, description and extension |  |  |  |
| pulseConfig | PulseConfig | CF |  |  |
| ctIModel | CtlModels | CF |  |  |


| DPC class |  |  |  |  |
| :--- | :--- | :--- | :--- | :--- |
| sboTimeout | INT32U | CF |  |  |
| sboClass | SboClasses | CF |  |  |
| d | VISIBLE <br> STRING255 | DC |  | Text |
| dU | UNICODE <br> STRING255 | DC |  |  |

Data attributes can be addressed by their full path, for example "CSWI.Pos.stVal" contains the switch status of a certain isolator and can have the values: on, intermediate-state or off (bad-state only in case of defect auxiliary switches).

Part 7-2: Abstract communication services between logical nodes are described in IEC 61850-7-2. FolAbstract Communication Service Interface (ACSI) lowing figure shows how a client (e.g. "IHMI") sends an operate command ("Oper_req") to a control object (e.g. "CSWI"). The control objects sends a confirmation ("Oper_rsp") immediately and a report with the updated status ("Report_req") after the "CSWI.Pos.stVal" has changed to off.


Another service is GOOSE (Generic Object Oriented Substation Event), which is used for fast communication like tripping a circuit breaker or interlocking. GOOSE messages are always retransmitted. Slowly in stable conditions and fast after an event has occurred.


T0 Retransmission in stable conditions (no event for a long time).
(T0) Retransmission in stable conditions may be shortened by an event.
T1 Shortest retransmission time after the event.
T2, T3 Retransmission times until achieving the stable conditions time.

Part 8-1: Specific Communication Service Mapping (SCSM)

Communication profile (IEC 61850 stack) is described in IEC 61850-8-1. This part of the standard maps the abstract services from IEC 61850-7-2 to concrete communication profiles: While core ASCI services like control and reporting are using MMS and TCP/IP, the fast GOOSE service is working directly on the Ethernet layer.

|  | IEC 61850-7-2 |  |
| :---: | :---: | :---: |
|  | Client/server | GOOSE |
| Application | MMS (ISO 9506) |  |
| Presentation | ASN. 1 |  |
| Session | Session |  |
| Transport | TCP |  |
| Network | IP | Ethertype |
| Data Link |  |  |
| Physical |  |  |

Part 6: Configuration description language

SCL (Substation Configuration Language) is described in IEC 61850-6. The SCL is an XML based definition of how to describe substation single line, IEDs with logical nodes and their communication through a communication bus. The following types of SCL files are distinguished:

- SSD (System Specification Description), describing the single line and required logical nodes.
- ICD (IED Capability Description), containing the capability (possible logical nodes) of one IED type.
- CID (Configured IED Description), containing the configuration of a concrete IED with instantiated logical nodes, services and communication addresses.
- SCD (Substation Configuration Description), containing description of the complete substation, consisting of single line, communication bus and IEDs with logical nodes, which are linked to the primary equipment in the substation section.


The part IEC 61850-6 also includes the roles of different tools as well as the engineering concepts. It distinguishes between system wide engineering with a system configurator and IED specific engineering with an IED Configurator.

Part 10: Conformance testing
Abbreviations
Short list of IEC 61850 related abbreviations.

| ASCI | Abstract Common Services Interface |
| :--- | :--- |
| CID | SCL type: Configured IED Description |
| CILO | Logical node in Group "C -Control functions": Interlocking |
| CSWI | Logical node in Group "C - Control functions": Switch Controller |
| DPC | Common data class "controllable double point" |
| GOOSE | Generic object oriented substation event |
| ICD | SCL type: IED Capability Description |
| IED | Intelligent Electronic Device (e.g. control or protection relay, AC500, ...) |
| IHMI | Logical node in Group "I - Interface and archiving": Generic human - machine |
| MICS | Modeling Information Conformance Statement |
| MMS | Manufacturing Message Specification, ISO 9506 |
| PICS | Protocol Extra Information |
| PIXIT | Position of a switch or circuit breaker |
| Pos | SCL type: Substation Configuration Description |
| SCD | Substation Configuration Language |
| SCL | SCL type: System Specification Description |
| SSD | status Value |
| stVal | Transmission Control Protocol / Internet Protocol |
| TCP/IP | Tissue Information Conformance Statement |
| TICS | Logical node in Group "X - Switchgear": Switch, disconnector or earth-switch |
| XSWI | Panformance Statement |

Further abbreviations can be found in the standard:
IEC 61850-2: Glossary - Terms and definitions, abbreviations
IEC 61850-7-3: List of all Common data classes, like DPC
IEC 61850-7-4: List of all Logical nodes, like CSWI

### 6.3.4.2.2 AC500 and IEC 61850

## General

After installation of the IEC 61850-server package, the AC500 can act as an Intelligent Electronic Device (IED) in an IEC 61850 network. The AC500 IEC 61850 package allows to connect substation automation systems to PLC applications Automation Builder is used as IED configurator tool for modelling IEC 61850 data and communication in AC500.

## Capabilities

This IEC 61850 server package $4 . x$ allows the AC500 V3 to act as an IED with IEC61850 server, Edition 1, providing following functionality:

- Configure a data model with logical nodes and map their data attributes to IEC 61131 application.
- Configure data sets to bundle data objects and attributes for publishing them as MMS reports or GOOSE control blocks.
- MMS server for core ASCI services like sending MMS reports over TCP/IP.
- GOOSE Publisher, sending GOOSE control blocks in a very fast way by bypassing TCP/IP layers.
- GOOSE Subscriber, receiving GOOSE reports from other IEDs and map their data attributes to the IEC 61131 application.
- SNPT client or server for time synchronization (included in the standard package of AC500).


To get information and an understanding about the implemented possibilities of IEC 61850 in the AC500, the details are described in the IEC 61850 conformance documents.
Conformance documents, which can be found in the example folder.

## Example projects for the libraries can be found in the example folder: "Help $\rightarrow$ Project examples" in the Automation Builder menu.

Prerequisite: Install the related library package with "Automation Builder $\rightarrow$ Tools $\rightarrow$ Installation Manager" - [Modify].

- MICS (Modeling Information Conformance Statement), contains the declaration of the used logical node types.
- PICS (Protocol Information Conformance Statement), contains the details and what is supported regarding protocol facilities.
- PIXIT (Protocol Extra Information), contains additional information on how the IEC 61850 is implemented and used.
- TICS (Tissue Information Conformance Statement), contains the supported tissues, which are handled in the tissues process as defined by UCA, Utility Communication Architecture forum. The tissues handling is found in IEC TICS.

The same folder also contains the TÜV certificate for AC500 V3 midrange CPUs: PM5630, PM5650, PM5670 and PM5675. AC500-eCo V3 as released after the certification but is capable of IEC 61850 as well. The certificates can also be accessed online from the UCA database. Search for organization "ABB Automation GmbH".

Automation Builder is used as IED configurator tool according to IEC 61850-6 for modelling the IEC 61850 data structures and connecting them to the PLC application. It allows import and export of SCL to transfer detailed configuration information between Automation Builder and other IEC 61850 configuration tools:

- .cid file (from another IED) or .scd file (complete substation with several IEDs) can be imported.
- .cid file (AC500 configured as IED) can be exported.


## Limits/technical data

- Edition 1 of IEC 61850 is supported only.
- MMS server only.
- Data sets for MMS and GOOSE reporting: Max. 20 data sets with max 50 entries each.
- MMS reporting: Max. 5 MMS clients.
- GOOSE Subscriber: Max. 50 Ethernet frames per cycle. If this is not sufficient, the workaround is to reduce the cycle time.
- Not possible to have 2 or more IEC 61850 server in one Automation Builder project. Workaround: Create 2 or more projects.
- Only one logical device per IEC 61850 server.
- Only one report per data set.
- Automation Builder is an IED Configurator but no system configurator tool as defined in IEC 61850-6. It cannot export .ssd or .scd files.


## Installation

The "IEC 61850 Server (PS5602)" package can be installed with the Automation Builder Installation Manager.

## Automation Builder

## Engineering Productivity



To set the IEC 61850 server into RUN, a runtime license PS5602-61850 per CPU must be purchased through the local sales support.

* ${ }^{4}$ Chapter 6.3.2.2.2 "PLC runtime licensing" on page 1446


## Engineering workflow

## Examples

AC500 as IED can act as IEC 61850 MMS server (Example A) and / or GOOSE Publisher and Subscriber (Example B). Most of the engineering steps are common and are explained in this chapter. The examples projects $A$ and $B$ can be found in the Automation Builder example folder.


Example projects for the libraries can be found in the example folder: "Help $\rightarrow$ Project examples" in the Automation Builder menu.

Prerequisite: Install the related library package with Automation Builder Installation Manager - [Modify].

Example A Communicating through MMS server:
A simulated switch in AC500 which can be operated from an operator IED and sends back its status through an MMS report.


Example B Communication though GOOSE:
AC500 publishes a GOOSE control block (e.g. "ping") to another IED (for example REF615 or a second AC500), which itself sends back a GOOSE control block (e.g. "pong"), that is subscribed by AC500 IED.


For engineering of the above functionalities，the Automation Builder is acting as IED configu－ rator tool according to IEC 61850－6．The following engineering steps（yellow circles in the figures above）need to be executed：
1．Configure the＂IEC 61850 Server＂with logical nodes，which are connected to global variables in the IEC 61131 application．
（3）Chapter 6．3．4．2．2．5．2＂Configuration of IEC 61850 server with logical nodes＂ on page 1680

2．Group data objects and attributes of the logical nodes into data sets．
枓 Chapter 6．3．4．2．2．5．5＂DataSet＂on page 1688
3．Publish data sets as MMS report．
« Chapter 6．3．4．2．2．5．6＂Report（MMS）＂on page 1689
4．Publish data sets as GOOSE Publisher．
² Chapter 6．3．4．2．2．5．7＂GOOSE Publisher＂on page 1691
5．Export server as＊．cid file．So that other IEDs can access logical nodes and data sets and subscribe to MMS reports or GOOSE control blocks．
Chapter 6．3．4．2．2．5．8＂Export Server＂on page 1693
6．Configure GOOSE Subscriber to subscribe GOOSE control blocks from other IEDs after importing＊．cid or＊．scd files．The control blocks are connected to the＂PLC＿PRG（PRG）＂ application．

ヶ Chapter 6．3．4．2．2．5．9＂GOOSE Subscriber＂on page 1694
7．Generate code for IEC 61850 POUs．
② Chapter 6．3．4．2．2．5．10＂Generate IEC61850 code＂on page 1696
8．Generate the application＂PLC＿PRG（PRG）＂including logical node functionality．
乡 Chapter 6．3．4．2．2．5．11＂Generate the application＂on page 1697

## Configuration of IEC 61850 server with logical nodes

The following describes how to create an＂IEC 61850 Server＂and configure the IEC 61850 data model with logical nodes，data objects and data attributes which can be connected to global variables in the IEC 61131 application．Starting point is a new or existing project with AC500 V3 CPU．

Add IEC 61850 Add an＂IEC 61850 Server＂object below Ethernet port＂ETH1＂or＂ETH2＂，depending on which Server IP address the MMS server should use．The IP addresses of＂ETH1＂and＂ETH2＂are configured with the＂IP－Configuration＂tool．


> GOOSE Publisher and Subscriber is independent of the IP address. Instead, the Ethernet MAC addresses must be configured in GOOSE Publisher and/or Subscriber. Even if only GOOSE is configured like in example B, an "IEC 61850 Server" is required, which can be placed below any Ethernet port.
̌ Chapter 6.3.4.2.2.5.7 "GOOSE Publisher" on page 1691
${ }^{4}$ Chapter 6.3.4.2.2.5.9 "GOOSE Subscriber" on page 1694
The properties of the "IEC 61850 Server" can be configured. For basic functionality, the default values can be used.


| Property | Description |
| :--- | :--- |
| Server Name | Name of the server. |
| Max.client count | The maximum number of clients that can connect to the IED, possible <br> values: $1,2,3,4,5$ |
| Allowed IPs | Allowed IPs for clients $1 \ldots 5$. <br> Default is 0.0 .0 .0, whereby an IP address that equals 0.0 .0 .0 means that <br> no IP address validity test will take place. If more than one client connec- <br> tion was selected above, additional IPs must be configured for each one. <br> As soon as an IP address is parameterized with 0.0.0.0, testing for all <br> connected clients is deactivated. |


| Property | Description |
| :---: | :---: |
| Time synchronisation | Selection: SNTP <br> SNTP (only possible option): (S)NTP time synchronization <br> « Chapter 6.3.4.2.2.7.1 "Time sync" on page 1707 <br> The following values should only be used if the PLC time is interpreted as UTC time: <br> 1. Input field Time zone: Offset between Greenwich (GMT)- and the local time (for Germany 1 h , for example). The value is limited between -12 and +14 . <br> 2. Input field: DLS Mode: Day light saving. Possible values: <br> - $0=$ No automatic summertime/wintertime changeover. <br> - 1 = Timeover from wintertime to summertime on last the Sunday in March, changeover from summertime to wintertime on the last Sunday in October. |
| Task Prio I Interval | Task, which will be generated by Generate IEC 61850 code: <br> 1. Input field - entry of the priority, <br> 2. Input field - entry of the interval in ms. |
| TCP KeepAlive [sec] | The KeepAlive is to check the connection to the client. |
| User application name | Enter the name of the user application for optimized GOOSE performance. <br> The IEC 61850 task will execute in the following order: <br> 1. GOOSE Subscriber <br> 2. User application <br> 3. GOOSE Publisher and MMS reporting <br> If this field is empty, the user application (step 2) will not be called by the IEC 61850 task, but by the standard task. <br> Two tasks are asynchronous and might lead to slower performance. <br> Chapter 6.3.4.2.2.6.7 "GOOSE performance optimization" on page 1705 |

The server is the root object for creating the IEC 61850 data model consisting of "LogicalDevice", logical nodes, data objects and data attributes.

Add LogicalDe- Add the "LogicalDevice" by selecting it in the left window and inserting it below the server with vice the arrow button ">".


Following properties of the logical device can be configured.

| Property | Description |
| :--- | :--- |
| Device Name | Name of the LogicalDevice |

Add logical nodes

Two standard logical nodes ("LLNO" and "LPHD1") are added automatically. Further logical nodes (e.g. "XSWl") can be added by selecting them from the left window and inserting them below the logical device with the arrow button ">".

Following properties of the logical device can be configured.


| Property | Description |
| :--- | :--- |
| Node prefix | Optional prefix for the selected LNC instance. <br> The prefix is put in front of the LN name in the <br> server tree. |
| Logical Node index | Index to differentiate between several <br> instances of the same logical node type. It is <br> assigned automatically with increasing order, <br> starting with 1. If it is changed manually, dupli- <br> cates should be avoided. |

Add data objects

Logical nodes are instantiated with their mandatory data objects "DO". Optional data objects can be added by selecting them in the left window and inserting them below the logical node with the button " $>$ ".


Data objects have no properties to be configured.

Add data attributes and mapping to variables

Data objects are instantiated with their mandatory data attributes "DA". Optional data attributes can be added by selecting them in the left window and inserting them below the data object with the button " $>$ ".


Data attributes can be mapped to variables in the IEC 61131 application by the following properties.


| Property | Description |
| :--- | :--- |
| Monitoring Var I Initvalue | Name of the monitoring variable, which can <br> be written by AC500 and read by an IEC <br> 61850 client, for example the status stVal of <br> a switch position / An initial value can be <br> defined optionally. |
| Control Variable | Name of the control variable which can be <br> written from an IEC 61850 client and read <br> by AC500, for example a switch command <br> Oper.ctlVal. |


| Property | Description |
| :--- | :--- |
| Autom. declare | By activating the Autom. declare checkbox <br> the variable is declared by the IEC 61850 con- <br> figurator and stored in the IEC61850_Gener- <br> ated_GVL (of the IEC61850 generated POUs <br> folder) after generate code of the IEC 61850 <br> server. <br> If it is not declared automatically, the variable <br> must be declared by the user. Instead, an IO <br> point can be chosen as monitoring or control <br> variable. |
| Trigger options | With the trigger options you select the events <br> which might trigger a report. The selected <br> trigger option is displayed in the status bar. |
| \& Chapter 6.3.4.2.2.5.4 "Status bar" <br> on page 1686 |  |

## Monitoring - and control direction, reading and writing

It may be the case that the IEC 61850 client will read the monitoring variable of an attribute and will write the control variable of the same attribute. Monitoring variable and control variable must not be the same CODESYS variable.

In monitoring direction, the data flow takes place from the "IEC 61850 Server" to the connected IEC 61850 client to read the CODESYS monitoring variable, for example MMS report service.

In control direction the data flow takes place from the connected IEC 61850 client to the "IEC 61850 Server" to write the CODESYS control variable, for example operate service.
The following figure shows an example where the "ctriModel" can be changed from server side (Monitoring variable "control_model_mon" with initial value 1) or from client side (Control variable "control_model_ctrl").


| Properties: |
| :--- | :--- |
| Monitoring Var I Initvalue:  <br> Autom. declare:  <br> Control Variable: Oper_var_ctrl <br> Autom. declare: None $\bigcirc$ Dchg $\bigcirc$ Qchg $\bigcirc$ Dupd <br> Trigger option:  |

## Status bar

In the status bar of the IEC 61850 editor you find object-specific detail information about the selected object.

Object information: Server and logical device

| Name |
| :--- |
| Sever |

Only the information "Name" is displayed for the server and the logical device.

Object information: LN

| Name | Beschreibung | Gruppe | Prafix |
| :---: | :---: | :---: | :---: |
| LPHD | Physical device ifformot | [Dood-System |  |


| Status | Description |
| :--- | :--- |
| Name | Name of the selected LN instance. |
| Description | Description of the selected LN instance. |
| Group | Associated group of the LN instance . |
|  | Examples: |
|  | $[$ Axxx]-Automatic |
| [Cxxx]-Control |  |
| [Gxxxx]-Generic |  |
|  | $\ldots$ |
| Prefix | Prefix of the LNC instance, entered by the user. |

Object informa-
tion: Common
Data Class
Object (CDC
Object)

| Name | Beschreibung | Option | CDC | Instnr |
| :--- | :--- | :--- | :--- | :--- |
|  | Beh | Behaviour | M | INS |


| Status | Description |
| :--- | :--- |
| Name | Name of the attribute. |
| Description | Description of the selected CDC instance. |
| Option | Option of the selected CDC instance. <br> $\mathrm{M}=$ Mandatory <br> $\mathrm{O}=$ Optional |
| CDC | Type of the selected CDC instance. |
| Instno | Instance number of the CDC instance. <br> Only optional CDCs can have instance numbers. If there is <br> only one optional CDC, it has no instance number. Otherwise <br> 1 to n. |

Objects with the mandatory options are inserted automatically when adding the logical device.


| Status | Description |
| :--- | :--- |
| Trigger option | Trigger option of the attribute. <br> dchg = data change <br> dupd = data update |
| qchg = quality change |  |
| <empty = no trigger option |  |, | Associated variable |  |
| :--- | :--- |
| Value | Access for the attribute. <br> R = Read <br> W = Write <br> RW = Read and write |
| Writeable |  |

## DataSet

In the tab of the IEC 61850 editor it is possible to create and delete data sets, assign data attributes (DA) and data objects (DO) to a data set and delete existing assignments.

"DataSet" tab with "LLNO.DS1" data set consisting of data objects and data attributes.

Structure of the The left window displays the created "IEC 61850 Server".
tab
\& Chapter 6.3.4.2.2.5.2 "Configuration of IEC 61850 server with logical nodes" on page 1680
The right part is for creating, editing and deleting of data sets. In the upper right window data sets are listed. In the lower right window, the attributes and data objects can be inserted by selecting them in the left window and inserting them by the button " $>$ ".
The maximum number of data sets is given in Limits/technical data.
«2 Chapter 6.3.4.2.2.3 "Limits/technical data" on page 1678
The status bar at the bottom shows more details of the chosen items.


#### Abstract

The order of the attributes of a data set is important for the receiving and the sending of GOOSE messages. Type and order of the entries from sender and recipient must be identical for GOOSE communication.


※ Chapter 6.3.4.2.2.5.9 "GOOSE Subscriber" on page 1694

## Buttons:

- [New]: Create a new data set. This is displayed in the "DataSets" section and is named "LLNO.DataSet_Suffix". The suffix is incremented beginning with 0 (1. Data set: LLNO.DataSet_0 ...)
- [Delete]: Delete a data set: Select the desired data set in the "DataSets" section and activate the [Delete] button.
- [>]: Assign an attribute or a data object to the selected data set. First select the data set in the upper right window then select the attribute or the data object in the left window and activate the ">" button.
- [<]: Deletes an element from a data set. First select the data set in the upper right window then select the attribute or the data object in the lower right window and activate the " $<$ " button.
- $\mathbf{V}$ : Moves the selected entry one row up.
- $\boldsymbol{\triangle}$ : Moves the selected entry one row down.

Input field:

- "Name: "Name of the data set can be edited. The name gets the prefix "LLNO.".


## Report (MMS)

In this tab of the IEC 61850 editor it is possible to create and parameterize buffered und unbuffered reports. A report transports the data, that are assigned to it, to the connected client in the event of a trigger. Each report a data set must be assigned to.


In the left part the created report control blocks (RCB) are listed. The following buttons are available:

- [New]: Create a new report control block.
- [Delete]: Delete the selected report control block.

In the middle part you make general settings for the reporting configuration.

| Setting | Description |
| :--- | :--- |
| Buffersize | Buffer size of buffered reports (in bytes). |
| Name | Unique report block name within the logical node. |


| Setting | Description |
| :--- | :--- |
| Description | Description of the report block. |
| Integrity Period [ms] | Stealthy general interrogation. After this time the referenced data set will be actuated. <br> Time (ms) between two messages. <br> The messages are transferred cyclic, independent from other events. |
| Buffered | Enable / disable the report buffering. <br> A buffered report stores the data, even if there's no connection to the client. In the <br> case of an unbuffered report, the messages will get lost, if there is no connection to <br> the client. |
| Buffer Time [ms] | Buffer time is the amount of time that the server waits to transfer a report after a <br> given event occurs. Events that occur during this time period are collected and then <br> transferred as a batch. |
| If the buffer time is 0, the telegram will be sent immediately. For example, if the buffer <br> time 10s the telegram will be sent after this time period or when the value changes the <br> second time. |  |
| Config Revision | Versioning is used to identify whether or not a member was deleted from a data <br> set or whether member order has changed. Such changes cause values to not be <br> transferred, or cause values to be in a different location within the report. Such an <br> event is communicated to the client with a new version number. <br> Since all data sets are firmly defined, this identifier does not apply to the solution <br> described here. |
| DataSet | Data set reference. |

The right part is for the setting of the following options:

- "General options": Control of the reporting behavior. An activated checkbox means, that the information is transferred by the message.
- "Buffered specific options": can be activated, if the option "Buffered:" is activated.
- "Trigger options": Determining of attributes to select the events which may trigger a message.

If the checkbox is activated the information will be transferred by the message.

| Setting | Description |
| :--- | :--- |
| Send Config Revision | "Config Revision" information. |
| Send Data Reference | Enable/disable to transfer the complete reference information, <br> for example: LogicalDevice/GGIO1.ST.Mod.ctINum |
| Send DataSet name | Enable/disable to transfer the data set name. |
| Send Reason for Inclusion | Enable/disable to transfer the reason of transmission for each <br> attribute. |
| Send Sequence Number | Enable/disable to transfer a unique sequence number for each <br> message. |
| Send Time Stamp | Enable/disable to transfer the timestamp of transmission for <br> each message. |


| Setting | Description |
| :--- | :--- |
| Send Entry ID | Enable/disable to transfer the entry ID. |
| Send Buffer overflow | Enable/disable to transfer the message if a buffer overflow <br> occurs. |


| Setting | Description |
| :--- | :--- |
| Data Change | Enable/disable to trigger the report if a "Data Change" event <br> occurred. |
| Data Update | Enable/disable to trigger the report if a "Data Update" event of <br> an attribute occurred. |
| Quality Chance | Enable/disable to trigger the report if a "Quality Change" event <br> of an attribute occurred. |
| Integrity | Enable/disable the cyclic transmission of the report inde- <br> pendent of any data changes (Stealthy general interrogation). <br> The time period has to be defined in the "Integrity Period" <br> general setting. |
| General Interrogation |  |

Create a report control block and assign it to a data set

1. Activate the [New] button.
2. Select the desired data set from the "DataSet" selection list.

## GOOSE Publisher

In the "GOOSE Publisher" tab of the IEC 61850 editor you create, edit and delete GOOSE messages. If a value changes in the selected data set, a GOOSE message is sent.


## NOTICE!

The order of the attributes of a "DataSet" is important for the receiving and the sending of GOOSE messages. Type and order of the entries from sender and recipient must be identical for GOOSE communication.

## NOTICE!

To receive a GOOSE message from an IED, sender and recipient must have the identical settings in the following input fields:

- "APPID"
- "GOOSE ID"
- "DataSet" structure (with regard to order and data type of the attributes).

After the data set is sent, it is sent again after a time interval of 500 ms . The repeat time then doubles and the data set is sent again. The data set is sent repeatedly until the value set in the "Repeat Time" input field is reached. The data set is then sent again at the "Repeat Time" interval.

Sections of the tab:
List of the GOOSE control blocks (GCB) in the left part.
A GOOSE control block is a GOOSE message.
Buttons:

- [New]: Create a new GOOSE control block.
- [Delete]: Delete the selected GOOSE control block.

General settings in the top right part:

| Setting | Description |
| :--- | :--- |
| Name | Name of the GOOSE control block, editable. |
| Description | Description of the GOOSE control block. |
| GOOSE ID | Unique character string of the GOOSE control block, editable. |
| DataSet | Data set sent as a GOOSE message. |
| MAC | Multicast addressing <br> Multicast addressing is used to send GOOSE messages. Addressing <br> allows an entire group of devices to exchange data with each other. <br> Requirement: Unique address allocation of the different device groups. <br> Valid range of values: 01-0C-CD-01-00-00....01-0C-CD-01-01-FF |
| APPID | Application-ID <br> Number for the system-wide unique identification of a GOOSE control <br> block. To exchange GOOSE telegrams, this number must be identical for <br> sender and recipient. <br> Valid range of values: 0 ... 4095 |
| Source Address (MAC) | [Browse] button: Looks for an Ethernet port in the network. <br> Requirement - an existing network path to the PLC. |

GOOSE Publisher settings in dem middle right part.

| Setting | Description |
| :--- | :--- |
| Needs Commissioning | Indicates whether the control block must be checked. <br> Value is provided from the configurator. Usage of the flag is customer-spe- <br> cific. |
| DataSet Config Revision | Integer value with the version of the GOOSE control block. |
| Max. Time [ms] | "timeAllowedtoLive" for slow retransmission phase (no new values). <br> "timeAllowedtoLive" $=2$ * resend-time. <br> \& Chapter 6.3.4.2.2.6.6 "Resend of GOOSE messages" on page 1704 |
| Min. Time [ms] | Resend-time of first package after change of values. "timeAllowedtoLive" <br> = * resend-time, This value should be greater than the cycle time of the <br> IEC61850 task. <br> \& Chapter 6.3.4.2.2.6.6 "Resend of GOOSE messages" on page 1704 |


| Setting | Description |
| :--- | :--- |
| VLAN | 'Virtual Local Area Network' <br> Logical subnet within a physical network. Multicast messages can be <br> passed through and filtered. The configuration is done in managed <br> Ethernet switches. <br> If the "VLAN" checkbox is activated, values can be entered into the "VLAN- <br> ID" and "VLAN-Priority" input fields, concerning the passed through of mas- <br> sages via switches. |
| VLAN-ID | A value of 0 is a non-configured "VLAN" in which the switch performs no <br> filtering. This value is recommended when no logical network should be set <br> up. <br> Valid range of values: $0 \ldots 4095$. |
| VLAN-Priority | Messages within a managed Ethernet switch can be forwarded depending <br> on the priority. <br> Valid range of values: $0 \ldots 7$. <br> Default value for GOOSE: 4. |

## Create an GOOSE control block and assign it to a data set <br> Export Server

Content of the data set assigned to the GOOSE control block in the bottom part.

1. Activate the [New] button.
2. Select the desired data set from the "DataSet" selection list.

After configuration of the "IEC 61850 Server" including logical nodes, data sets, MMS reports and GOOSE control blocks the configured IED can be exported as .cid file, by the menu entry "IEC61850 $\rightarrow$ Export Server".


You can choose a folder and a filename, for example: IEC61850_ExampleA_AC500.cid.
This file is based on XML and can be opened in any text editor:


The extension ".cid" (Configured IED Description) contains the configuration of a concrete IED with instantiated logical nodes (line 41, 61 and 62), data sets (line 42) services (line 47) and communication addresses (line 8-17).
This file can be imported in other IEC 61850 engineering tools, either system configurator or IED configurator for other IEDs.
The IP address information (line 9-11) is empty, since in AC500 the IP address configuration is decoupled; it is done in the "IP-Configuration"tool, possibly in a later step.

If the correct IP address of the AC500 IED is required by the other IEC 61850 engineering tool, it must be updated in the ".cid" file, which can be done using any text editor:

```
#IEC61850_ExampleA_AC500.cid \boxtimes 嘈IEC61850_ExampleA_AC500_withIP.cid 区 
    G<SCL xmlns:xsd="http://www.w3.org/2001/xMLSchema" xmlns="http://www.iec.ch/61850/2003/SCL">
        <Header id="IEC61850_Server" nameStructure="IEDName" />
        <!--Create on ABB AG Configurator Version V2.0.0 at 22/07/2022 09:40:15-->
        <Communication>
            <SubNetwork name="node">
                <ConnectedAP iedName="IEC61850_Server" apName="W1">
                        <Address>
                            <P type="IP">192.168.|0.120</P>
                    <P type="IP-SUBNET">255.255.255.0</P>
                    <P type="IP-GATEWAY">0.0.0.0</P>
                            <P type="OSI-AP-Title">1,1,1,999,1</P>
```

(3) Chapter 6.3.2.9.2 "Configuration of the IP settings with the IP configuration tool" on page 1506

## GOOSE Subscriber

In the tab " GOOSE Subscriber" of the IEC 61850 editor you make settings for the receiving of GOOSE messages.


## NOTICE!

The order of the attributes of a "DataSet" is important for the receiving and the sending of GOOSE messages. Type and order of the entries from sender and recipient must be identical for GOOSE communication.

## NOTICE!

To receive a GOOSE message from an IED, sender and recipient must have the identical settings in the following input fields:

- "APPID"
- "GOOSE ID"
- "DataSet" structure (with regard to order and data type of the attributes).

Sections of this tab:
List of the GOOSE control blocks in the left part.
Buttons:

- [New]: Create a new GOOSE control block.
- [Delete]: Delete the selected GOOSE control block.
- [Import]: Import a GOOSE control block in the SCL format.

General settings in the top middle part:

| Setting | Description |
| :--- | :--- |
| Name | Name of the GOOSE control block, editable. |
| Description | Description of the GOOSE control block. |
| GOOSE ID | Unique character string of the GOOSE control block, editable. |
| DataSet | Data set sent as a GOOSE message. |
| MAC | Multicast addressing <br> Multicast addressing is used to send GOOSE messages. Addressing <br> allows an entire group of devices to exchange data with each other. <br> Requirement: Unique address allocation of the different device groups. <br> Valid range of values: 01-OC-CD-01-00-00....01-OC-CD-01-01-FF |


| Setting | Description |
| :--- | :--- |
| APPID | Application-ID <br> Number for the system-wide unique identification of a GOOSE control <br> block. To exchange GOOSE telegrams, this number must be identical for <br> sender and recipient. <br> Valid range of values: $0 \ldots 4095$ |
| Source Address (MAC) | [Browse...] button: Looks for an Ethernet port in the network. <br> Requirement - an existing network path to the PLC. |

List to assign GOOSE messages to global variables in the bottom right part.
All attributes within the selected data set are listed in this list. You can assign incoming GOOSE messages to global CODESYS variables. For this, select the desired attribute in the list and edit the name of a global variable in the "Varname" column. If you edit a new variable name a global variable will be created, if you activate the "Use default name" checkbox, a variable name is generated automatically. This variable will be written by incoming GOOSE messages.

The variables will be stored "IEC61850_Generated_GVL" (of the "IEC61850 Generated POUs" folder) after generating the code of the "IEC 61850 Server".

* Chapter 6.3.4.2.2.8.1 "Generate code" on page 1712

| No. | Reference | Type | Varname | Use default name |
| :--- | :--- | :--- | :--- | :--- |
| 1 | LogicalDevice/GGIO1.ST.Mod.ct\|Num | INT8U | Var_ctINum | $\square$ |
| 2 | LogicalDevice/GGIO1.ST.Mod.t | Timestamp | Subs_0_Entry1 | $\nabla$ |

## Generate IEC61850 code

When all IEC 61850 related configuration is done according to the previous chapters, it must be translated into IEC 61131 code using the menu entry "IEC61850 $\rightarrow$ Generate code".


After a few seconds (depending on the IEC 61850 server size), the following IEC 61131 code is generated below "Application" in the "Devices" tree:

- Folder "IEC61850 Generated POUs", containing IEC 61131 code for all IEC 61850 functionality including the global variable list "IEC61850_Generated_GVL", which contains auto generated variables which must be connected to the application "PLC_PRG (PRG)".
- Task "IEC61850Task1", calling the code in a task which is configured in the configuration window of IEC 61850 Server object.
${ }^{4}$ ) Chapter 6.3.4.2.2.5.2 "Configuration of IEC 61850 server with logical nodes" on page 1680


## Generate the application

Finally，the application（typically＂PLC＿PRG（PRG）＂）including the logical node functionality needs to be configured．The IEC 61850 generated code only contains the logical node inter－ faces $=$ control and monitoring values．

The＂PLC＿PRG（PRG）＂application consumes the control values which were received by the ＂IEC $618 \overline{50}$ Server＂（e．g．Operate）or GOOSE Subscriber and calculates monitoring values which are then sent by the＂IEC 61850 Server＂（e．g．MMS Report）or GOOSE Publisher．

The following figure shows a simple example of a switch simulation in＂PLC＿PRG（PRG）＂， based on the global variables which were mapped to＂XSWI．Pos＂．
（y）Chapter 6．3．4．2．2．5．2＂Configuration of IEC 61850 server with logical nodes＂on page 1680
The simulated switch is controlled by the control variable＂oper＿var＿ctrl＂and the resulting status （＿ON or＿OFF）is written to the monitoring variable＂status＿var＿mon＂．

```
Devices
```

Devices

- IEC61850_ExampleA_AB251
- IEC61850_ExampleA_AB251
T IEC61850_ExampleA_AB251
T IEC61850_ExampleA_AB251
= 自\ PLC Logic
= 自\ PLC Logic
- Application
- Application
+D Diagnosis
+D Diagnosis
- IEC61850 Generated POUs
- IEC61850 Generated POUs
+ Logical Nodes
+ Logical Nodes
Tools
Tools
| IEC61850_Generated_GVL
| IEC61850_Generated_GVL
\& 目 IEC61850_LogicalDeviceFB (FB)
\& 目 IEC61850_LogicalDeviceFB (FB)
\# IEC61850_ServerPRG (PRG)
\# IEC61850_ServerPRG (PRG)
filil Library Manager
filil Library Manager
-] PLC_PRG (PRG)
-] PLC_PRG (PRG)
* Task Configuration
* Task Configuration
TIO_Bus
TIO_Bus
    - 曾 Interfaces
    - 曾 Interfaces
F. COM (<Empty>)
F. COM (<Empty>)
5. CAN (<Empty>)
5. CAN (<Empty>)
        - ETH Ethernet
        - ETH Ethernet
-7) ETH1 (IP Settings)
-7) ETH1 (IP Settings)
0.0
0.0
        + 目 IEC661850_Server (IEC 61850 Server)
        + 目 IEC661850_Server (IEC 61850 Server)
-4 X
-4 X
IEC61850_Generated_GVL X
IEC61850_Generated_GVL X
(* Generated on: 22/07/2022 09:32:18 4)
(* Generated on: 22/07/2022 09:32:18 4)
(* IEC61850 Configurator version: V2.0.0 - 2013-10-14 )
(* IEC61850 Configurator version: V2.0.0 - 2013-10-14 )
VAR_GLOBAL
VAR_GLOBAL
gfbIEC61850_LogicalDevice: IEC61850_LogicalDeviceFB;
gfbIEC61850_LogicalDevice: IEC61850_LogicalDeviceFB;
oper_var_ctrl : tyIEC61850_AT_BOOLEAN;
oper_var_ctrl : tyIEC61850_AT_BOOLEAN;
status_var_mon : IEC61850_ENOM_DA_DBPOS;
status_var_mon : IEC61850_ENOM_DA_DBPOS;
control_model_mon
control_model_mon
: IEC61850_ENOM_DA_CTLMODELS;
: IEC61850_ENOM_DA_CTLMODELS;
control_model_ctrl : IEC61850_ENUM_DA_CILMODELS;
control_model_ctrl : IEC61850_ENUM_DA_CILMODELS;
END_VAR

```
        END_VAR
```

```
```

File Edit View Project Build Online Debug Tools IEC61850 Window Help

```
```

```
```

File Edit View Project Build Online Debug Tools IEC61850 Window Help

```
```

```
    LC_PRG x
```

    LC_PRG x
    PROGRAM PLC_PRG
    PROGRAM PLC_PRG
    VAR
    VAR
    END VAR
    END VAR
    // Simulation of a double position switch (status_val_mon)
    // Simulation of a double position switch (status_val_mon)
    // which is controlled by oper_var_ctrl
    // which is controlled by oper_var_ctrl
    IF oper_var_ctrl = IRUE THEN
    IF oper_var_ctrl = IRUE THEN
        status var mon := IEC61850 ENUM DA DBPOS.IEC61850 ENUM DA DBPOS ON;
        status var mon := IEC61850 ENUM DA DBPOS.IEC61850 ENUM DA DBPOS ON;
    ELSE
    ELSE
        status_var_mon := IEC61850_ENOM_DA_DBPOS.IEC61850_ENOM_DA_DBPOS_OFF;
        status_var_mon := IEC61850_ENOM_DA_DBPOS.IEC61850_ENOM_DA_DBPOS_OFF;
    END_IF
    END_IF
        +用 -TU\%montiman
    ```

Real applications will contain more logic and connections to physical inputs or outputs．

\section*{Runtime and diagnosis}

\section*{General}

After the engineering workflow is completed，the resulting code（＂IEC 61850 generated POUs＂ and＂PLC＿PRG（PRG）＂）can be compiled and downloaded to the CPU．

In the following，the runtime behavior of operate，reporting and GOOSE，as well as possible diagnostic and performance optimizations are described．

\section*{Operate and reporting}

The following explanation is based on the example A，which was configured according to engineering workflow as MMS server with a switch simulation．The project file of this example can be found in the example folder of Automation Builder－Example A．


First step is to create a physical Ethernet connection between AC500 IED and operator IED = MMS client and configure IP addresses in a common IP address range. Then, both devices can be commissioned.

\section*{AC500 IED}
1. Configure AC500 according to engineering workflow as MMS server with a switch simulation or load the example project "IEC61850_ExampleA_ABxxx.project".
2. Download program to AC500 and set it to RUN.
\(\Rightarrow\) Now the IEC 61850 server is running and ready to operate.

\section*{Operator IED}

Any IEC 61850 MMS client can be used. In this example the Omicron tool "IEDScout" is used. Following figure shows the menu bar.


\section*{Connect to} AC500 IED by one of the following options
1. Click on menu button [Discover IED] and enter IP address of AC500.
\(\Rightarrow\) "IEDScout" will find the AC500 IED through Ethernet and read its configuration dynamically.
2. Click on the menu button [Open SCL] and browse for the .cid file which was exported in chapter Export Server, for example "IEC61850_ExampleA_AC500_withIP.cid".
出 Chapter 6.3.4.2.2.5.8 "Export Server" on page 1693

The browser opens and shows the active connection to the AC500, indicated by the green checkmark.

1. Browse to "DataModel \(\rightarrow\) Logical Device \(\rightarrow\) XSWI1"
2. Ensure that "Pos.ctrIModel" is set to "direct-with-normal-security". Otherwise, set the value to "direct-with-normal-security" by clicking the menu button [Write].
3. Click on "XSWI1 \(\rightarrow\).Pos".
4. Click the menu button [Control] to open the "Control" dialog.
5. Choose the "Value" (false or true) and press the button [Operate] to send this value to AC500 IED.


In AC500 online mode, the control and status variables can be monitored in "IEC61850_Generated_GVL" as well as in the "PLC_PRG" containing the switch simulation.


The Operator IED (Omicron IED Scout), can subscribe (menu button [Enable]) the MMS report "SwitchReport", which contains "XSW11.Pos" including data atributes status ("stVal"), quality (" \(q\) ') and timestamp ( ( 4 ').


The IEC 61850 generated POUs - "Logical Nodes" contain more details and can be used for diagnosis in online mode. For example, the following property shows, that the MMS report "SwitchReport" was enabled by one MMS client: "IEC61850_LLNO.fbSwitchReport.fbSwitchReport[1].rptEna".


\section*{GOOSE}

The following explanation is based on the example B , which was configured according to engineering workflow with GOOSE Publisher and Subscriber. The project file of this example can be found in the example folder of Automation Builder: Example B.


First step is to create a physical Ethernet connection between AC500 IED and another IED. Then, both devices can be commissioned.

\section*{AC500 IED}
1. Configure AC500 according to engineering workflow with GOOSE Publisher and GOOSE Subscriber or load the example project "IEC61850_ExampleB1_ABxxx.project".
2. Ensure that the source MAC address for GOOSE Publisher and GOOSE Subscriber are set correctly.
«y Chapter 6.3.4.2.2.5.7 "GOOSE Publisher" on page 1691
\({ }^{\circledR}\) Chapter 6.3.4.2.2.5.9 "GOOSE Subscriber" on page 1694
3. After a change of MAC address it is required to generate IEC61850 code again.
\({ }^{\wedge}\) Chapter 6.3.4.2.2.5.10 "Generate IEC61850 code" on page 1696
4. Download program to AC500 and set it to RUN.
\(\Rightarrow\) Now the IEC 61850 server with GOOSE Publisher and Subscriber is running and ready to operate.

\section*{Another IED}
1. Configure any other IED as bay IED with GOOSE Publisher and GOOSE Subscriber or load the example project "IEC61850_ExampleB2_ABxxx.project".
2. Set the source MAC addresses correctly.
3. Generate the IEC 61850 code.
4. Download to a second AC500 CPU.

The ping-pong between the two IEDs can be monitored by opening the 2 projects with 2 Automation Builder instances.


GOOSE diagnosis and troubleshooting
The IEC 61850 library has an object "IEC61850_GooseManager" with several variables, which can be accessed for diagnosis of the GOOSE manager. Example B1 contains a "GooseDiagnosis (PRG)", showing the relevant diagnosis variables.

- "IEC61850_GooseManager.iAdapterCnt"

Number of the Ethernet adapters used for GOOSE Publisher and GOOSE Subscriber, should be at least 1.
- "IEC61850_GooseManager.wRxCounter" Counter for received GOOSE messages. Should increase continuously, when other IEDs are sending GOOSE, otherwise:
- Check if the 'Source Address (MAC)' is configured correctly in the GOOSE Subscriber.
- Check with Wireshark if there are GOOSE messages in the network of the configured adapter.
- "IEC61850_GooseManager. wRxCounterSubscribed"

Counter for received GOOSE messages, which are subscribed. Should increase continuously, when other IEDs are sending GOOSE and same GOOSE messages are subscribed in AC500. If not:
- Check if "IEC61850_GooseManager.wSubscriberCnt" is \(>0\).
- Check configuration in GOOSE Subscriber
- Check further diagnosis properties in" prgIEC61850Subscriber".
- "IEC61850_GooseManager.wMaxLoopCnt"

Maximum number of received Ethernet packages in one cycle. Maximum value is 50 . If the maximum value is reached, reset it by setting
"IEC61850_GooseManager.xResetIndications" to TRUE for one cycle. If "wMaxLoopCnt" immediately reaches again the maximum value, the IEC 61850 cycle time is too slow for the high network traffic and GOOSE messages might be lost. Workaround is to reduce the IEC 61850 cycle time to make the GOOSE Subscriber faster.
- "IEC61850_GooseManager.wTxCounter"

Counter for transmitted GOOSE messages. Should increase continuously when GOOSE Publisher is configured. If not:
- Check GOOSE Publisher.
- Check "IEC61850_GooseManager.sStatus".
- "IEC61850_GooseManager.sStatus"

Contains error status of the GOOSE manager and can have the following values:
- '.'< empty> \(\rightarrow\) OK.
- Sending of GOOSE messages failed! Result: 20 (or other results). \(\rightarrow\) Check if the 'Source Address (MAC)' is configured correctly in GOOSE Publisher.
- GOOSE Publisher Eth. frame length > = 1500 bytes, reduce DO/DAs in data set. \(\rightarrow\) Reduce the number of data objects and/or data attributes in a data set.

For the GOOSE Subscriber there is an additional possibility for diagnosis. The "prgIEC61850Subscriber" has an array "afbSubscriber", containing one entry for each subscription.
```

AE IEC61850_ExampleB1_AB250.project** - Automation Builder |I - Premium
File Edit View Project Build Online Debug Tools IEC61850 Window Help

```



Following properties should be checked in case of problems:
- "dtLastAction" and "wRxCounter": Should increase continuously, when other IEDs are sending GOOSE and same GOOSE messages are subscribed in this subscriber.
- "sStatus":

Should show "GOOSE Block receive <GOOSE ID> succsessful".
- "xErrorTimeAllowedToLive":

Should always be "FALSE". If not, a GOOSE message was received only after the "timeAllowedtoLive" (as indicated in the previous GOOSE message) expired.
- Decrease cycle time of the IEC 61850 task.
\& Chapter 6.3.4.2.2.5.2 "Configuration of IEC 61850 server with logical nodes"
on page 1680
- Check Ethernet connection between GOOSE Publisher and GOOSE Subscriber.
- Check in Wireshark if GOOSE Publisher is sending next packages fast enough. Resend of GOOSE messages shows a correct example where "timeAllowedtoLive" is always double of the resend time, so that the subscriber has enough time to receive it.

\section*{Resend of GOOSE messages}

Resending of GOOSE is done according to the IEC 61580 specification.
Chapter 6.3.4.2.1 "Introduction to IEC 61850" on page 1671
In example B as described in the previous chapters, there are not resends because the ping-pong values are changing quickly before the configured GOOSE "Publish Min. Time" has expired.

The resend gets visible in Wireshark when the quick value changing is stopped (e.g. by stopping the other IED in the ping-pong example B). The following figure shows an AC500 GOOSE Publisher with max. time \(=4000 \mathrm{~ms}\), min. time \(=20 \mathrm{~ms}\) and configured IEC 61850 task cycle time \(=5 \mathrm{~ms}\).


Two phases are visible in the Wireshark screenshot.
1. Slow retransmission in stable conditions (no event for a long time) above the blue line. When the value does not change (column "integer" = pong value \(=2\) ):
- The GOOSE message is resent continuously and slowly.
- "timeAllowedtoLive" is 4000 ms (= max. time).
- Message is retransmitted every 2000 ms (half of "timeAllowedtoLive" according to IEC61850-8-1).
- "sqNum" is increasing, "stNum" is not changing.
2. Change of value and fast retransmission. When the value changes (column integer = pong value was forced from 2 to 3 ):
- A new package ("stNum" increased, "sqNum" = 0) is sent immediately (*), see blue line.
- Same package is resent continuously, starting with a high frequency.
- The resend time is starting at 20 ms (= configured min. time) and doubled with every resend until it has reached half of the max. time to go back to slow retransmission.
- The "timeAllowedtoLive" is always double of the next resend time (*).

The configured GOOSE min. time should always be higher than the configured IEC 61850 task cycle time!
(*) Sending GOOSE messages is executed in the IEC 61850 task and therefore can be delayed by the configured task cycle time (worst case).

\section*{GOOSE performance optimization}

The automatic generated IEC 61850 task is optimized in such a way, that GOOSE Publisher and GOOSE Subscriber is executed the same task. Anyhow, the example B needs 60 ms for one complete ping-pong procedure. This is plausible and can be explained with the following figure.


Although the application task ("PLC_PRG") could increase the ping or pong every 10 ms , it must wait for the IEC 61850 task ( 20 ms cycle time) to subscribe or publish the new data from or to the other PLC. In consequence it takes \(70-10=60 \mathrm{~ms}\) for one complete ping-pong.

\section*{Optimization steps}
1. Reduce the cycle time of the IEC 61850 task to 10 ms as well. This would increase the performance, but both tasks are still asynchronous and have to wait for each other.
2. Merge both tasks and by that optimize the execution order according to the IPO principle.
- Input_GOOSE Subscriber
- Process: In this example - Increase ping or pong.
- Output: GOOSE Publisher
\begin{tabular}{|c|c|c|c|}
\hline & IEC61850_ExampleB1_opt & IEC61850_ExampleB2_opt & \\
\hline & IEC61850Task1 & IEC61850Task1 & \\
\hline & 10 ms & 10 ms & \\
\hline time/ms & & & time/ms \\
\hline 10 & GOOSE Subscriber Ping \(=\) Pong +1 GOOSE Publisher & & 10 \\
\hline 15 & & \begin{tabular}{l}
GOOSE Subscriber
\[
\text { Pong = Ping + } 1
\] \\
GOOSE Publisher
\end{tabular} & 15 \\
\hline 20 & \begin{tabular}{l}
GOOSE Subscriber
\[
\text { Ping = Pong + } 1
\] \\
GOOSE Publisher
\end{tabular} & & 20 \\
\hline 25 & & \begin{tabular}{l}
GOOSE Subscriber
\[
\text { Ping = Pong + } 1
\] \\
GOOSE Publisher
\end{tabular} & 25 \\
\hline 30 & 4 & & 30 \\
\hline
\end{tabular}

This behavior can be achieved by the following configuration in the IEC 61850 editor.

1. Set "Task Prio I Interval" of IEC 61850 task to 10 ms .
2. Enter the "User application name" (was empty before), here: "PLC_PRG".
3. Since now the "PLC_PRG" is called by the IEC 61850 task, the "normal" task calling the "PLC_PRG" must be deleted (red cross in the figure above).
4. Do the same procedure for example B1.

With these settings the total ping-pong roundtrip time is reduced to 10 ms , which can be observed in Wireshark.
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline \multicolumn{8}{|l|}{4 *Ethernet 5} \\
\hline \multicolumn{2}{|l|}{File Edit} & dit View Go & - Capture & e Analyze Statistics & Telephony & Wireless Tools & Help \\
\hline \multicolumn{8}{|l|}{} \\
\hline \multicolumn{8}{|l|}{Q goose} \\
\hline \multirow[t]{10}{*}{No.} & & Time & integer & Source & Destination & & gocbref \\
\hline & & 330.005590 & 6781 & ABBAutom_ed:ed:es & こ̌ec-1c57 & _02:00:00 & AC500_B2Logicaldevice/Llnoscosccb_ping \\
\hline & & 40.004323 & 6782 & ABBAutcm_ed:ee:fc & Iec-Tc57 & _01:00:00 & AC500_B1LogicalDevice/LLHeŞGOSGCB_Pong \\
\hline & & 60.025622 & 6783 & AbBAutom_ed:ed:e6 & Iec-ic57 & _02:00:00 & AC500_B1Logicaldevice/Llno\$GO\$GCB_Ping \\
\hline & & 70.004369 & 6784 & ABBAutcm_ed:ee:fc & Iec-Tc57 & _01:00:00 & AC500_B1LogicalDevice/LLHeşGoscce_pong \\
\hline & & 60.025508 & 6785 & ABBAutom_ed:ed:e6 & Iec-TC57 & -01:00:00 & AC500_BILogicaldevice/LlnosGo\$GCB_Ping \\
\hline & & 590.004899 & 6786 & ABBAutom_ed:ee:fc & Iec-Tc57 & _01:00:00 & AC500_81LogicalDevice/LLHeşosccce_pong \\
\hline & & 00.025135 & 6787 & ABBAutom_ed:ed:es & Iec-Te57 & -01:00:00 & AC500_BILogicaldevice/Llnos60\$6CB_Ping \\
\hline & & 120.004829 & 6788 & ABBAutom_ed:ee:fc & ปес-イc57 & -01:00:00 & ACS00_BlLogicalDevice/L.tuescoscci_pong \\
\hline & & 330.005328 & 6789 & ABBAutom_ed:ed:es & Iec-Tc57 & _01:00:00 & AC500_B1Logicaldevice/LLHOSGO\$GCB_Ping \\
\hline
\end{tabular}

The time format in Wireshark was set to "Seconds since the previous displayed packet". In this log the ping takes \(\sim 4,5 \mathrm{~ms}\), the pong takes \(\sim 5,5 \mathrm{~ms} \rightarrow\) ping-pong \(=10 \mathrm{~ms}\).
Depending on the CPU type and the size of the application, the performance can be further improved by reducing the IEC 61850 task interval to less than 10 ms.

\section*{Advanced features}

\section*{Time sync}

If a (S)NTP server (e. g. Meinberg clock) is available in the network, the AC500 should be configured as (S)NTP client in order to receive the correct time information, which is set in the MMS and GOOSE reports along with the correct status bits. This is done in two steps:
1. A (S)NTP client node must be added below "Protocols" and the IP address of the (S)NTP server must be entered (192.168.84.250 in this example).

\(\Rightarrow \quad{ }^{4}\) Chapter 6.3.4.4.2 "(S)NTP client configuration" on page 1718
2. Set the time quality bit "ClockNotSynchronized" of the IEC 61850 time correctly. Therefore, the bit must be read from the "fbPmSntpInfo1. SourceInfo1.SrcState" and written to the IEC 61850 variable "IEC61850_SyncClockOnRTC.xClockNotSynchronized".

Example A contains a program "prgSNTP", which reads and updates this time quality bit every 30 seconds.

If the SNTP time is working correctly the "ClockNotSynchronized" is false.


If the SNTP server fails or connection is broken, the "ClockNotSynchronized" is true.


\section*{Bulk data engineering}

When many logical nodes, related data sets and reports need to be created, there are repeating configuration steps for similar modules. Copy and paste of such modules or working with MS Excel is currently not possible, but there is a way to do bulk data engineering via XML file.
1. Create an IEC 61850 configuration by the engineering workflow steps 1 (logical nodes), 2 (DataSets), 3 (Report (MMS)), 4 (GOOSE Publisher).
2. Export the configuration as XML: "Menu \(\rightarrow\) IEC61850 \(\rightarrow\) Export Server" Save as type XML.

3. Enhance XML file by copying logical nodes, related data sets and reports.

Ensure that new instances have new and unique IDs.

For example, when duplicating logical node "XSWI1", the new logical node should be renamed to XSWI2 by modifying the tags "NodeText" and "NodeInstanz".

\(\Rightarrow\)
This step requires some experience with XML and an understanding how the IEC 61850 logical nodes, data sets and reports are represented and related to each other inside this XML.
4. Import the configuration from the enhanced XML: "Menu \(\rightarrow\) IEC61850 \(\rightarrow\) Import Server", type XML.
\(\Rightarrow\) The existing configuration is overwritten by the enhanced configuration.
5. Perform the remaining engineering workflow steps: 5 (Export Server (as .cid)), 6 (GOOSE Subscriber), 7 (Generate IEC61850 code) and 8 (Generate the application).

\section*{Additional or new logical node types}

Basic logical nodes are defined in IEC 61850, part 8, further logical nodes are defined in related standards, for example IEC 61400-25.
AC500 was certified according to IEC61850 for a subset of logical nodes which is listed in the MICS document.

Documents can be found in the example folder: "Help \(\rightarrow\) Project examples \(\rightarrow\) PS5602-IEC61850 \(\rightarrow\) Datasheets" in the Automation Builder menu.

Prerequisite: Install the related library package with Automation Builder Installation Manager [Modify].

Anyhow, additional logical nodes can be enabled, and even new logical node types can be defined.

Logical nodes for AC500 are defined in the IEC61850 device description file: "C
\(\rightarrow\) ProgramData \(\rightarrow\) AutomationBuilder \(\rightarrow\) AB_Devices_2.5 \(\rightarrow 512 \rightarrow 10200001 \rightarrow\) 4.0.7.2
\(\rightarrow\) IEC61850DeviceDesc.xmI". This file can be edited in order to enable prepared logical nodes or to create new ones.


In this example the "<enable>" of "CSWI" is set from 0 to 1.
After restart of the Automation Builder, the "CSWl" can be chosen.


Limitation of logical nodes which are not enabled by default:
- Their compliance to IEC 61850-7-4 (Ed1) is not fully tested.
- Common Data Classes ("CDC") are not fully implemented.


They are needed by some (not enabled) logical nodes, for example "MHAl". Enabling such a logical node leads to compilation errors.

\section*{Disable GOOSE Publisher}

For some applications it might be useful to disable the GOOSE activity, for example for high availability solutions where one of two parallel CPUs should act in standby and not send GOOSE messages.
Since Automation Builder 2.4.0 and "IEC61850_Server" library 4.0.5.7, the "IEC61850_GooseManager" has a new property "xEnable". This property is "TRUE" by default, but can be set to "FALSE" if required.


\section*{Menu Command sorted by Categorie}

\section*{Generate code}

\section*{Symbol:}

On activating the "Generate code" command of the "IEC61850" category the code generation is started and the generated IEC 61850 code is stored in the folder "IEC61850 Generated POUs" in the device tree.

② Chapter 6.3.4.2.2.5.10 "Generate IEC61850 code" on page 1696

This command of the "IEC61850" category exports the current configuration. In the "Safe as " dialog select the format filter:
- XML files: For IEC 61850 format with all specific data, variable mapping, for example.
- SCL-Files: For IEC 61850 format to export data to other IEC 61850 tools.

If you have changed the configuration since the latest code generation, you will be asked whether new code should be generated before export.

\section*{Import Server}

This command of the "IEC61850" category discards the current configuration and imports a new configuration. In the "Save as" dialog select the format filter:
- XML files: For IEC 61850 format with all specific data, variable mapping, for example.
- SCL Files: For IEC 61850 format to import data from other IEC 61850 tools.

\section*{Options}

The "Options" command of the "IEC61850" category opens a dialog for the setting of different display options for the IEC 61850 configurator.
\begin{tabular}{|l|l|}
\hline Option & Description \\
\hline "Show FC besides data attribute" & Display option, shows functional constraint of attribute as a comment. \\
\hline "Show type besides data attribute" & Display option, shows type of attribute as a comment. \\
\hline \begin{tabular}{l} 
"Show trigger option besides data \\
attribute"
\end{tabular} & Display option, shows trigger option of attribute as a comment. \\
\hline "Show description besides data \\
objects" & Display option, shows description of attribute as a comment. \\
\hline "Enable SCL Private block" & \\
\hline "Select all Data Objects" & Debug-Option: Selection of all data objects (DO). \\
\hline "Select all Data Attributes" & Debug-Option: Selection of all data attributes (DA). \\
\hline
\end{tabular}

\subsection*{6.3.4.3 Modbus protocol}

\subsection*{6.3.4.3.1 Modbus on TCP/IP protocol}

Modbus TCP/IP is simply the Modbus RTU protocol with a TCP interface running on Ethernet. The Modbus message structure is the application protocol that defines the rules for organizing and interpreting the data independent of the data transmission medium.
A client is a device that actively polls for data from one or more devices. A server device is the passive component that waits for the client to poll for data in order to actually send it.
* \(\boldsymbol{y}^{2}\) Chapter 6.8.2.10.1 "Protocol description" on page 4523

\subsection*{6.3.4.3.2 Configuration of Modbus TCP/IP server}


A "Modbus TCP/IP Server" instance can be added to any specific Ethernet interface/IP address. Each interface supports max. one instance of "Modbus TCP/IP Server". Other protocols can be added in parallel.
1. Right click on ETH interface and click "Add object".
\(\Rightarrow\) The window "Add object below: ETH" appears.
2. Select "Modbus TCP/IP Server" and click "Add object".
\(\Rightarrow\) The node "Modbus_TCP_IP_Server" is added.
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline Setting the & Parameter & Type & Value & Default Value & Unit & Description \\
\hline parameters of & - Byte order & Enumeration of BYTE & Big endan & Big endian & & Big endian \(=1\); Litte endian \(=0\) \\
\hline Modbus_TCP_IP & - Port & WORD(1.65535) & 502 & 502 & & TCP Port on which the Server lstens \\
\hline Server & - Startup behaviour & Enumeration of BYTE & Active & Active & & Startup behaviour \\
\hline & - Behaviour in state inactive & Enumeration of BYTE & No activity & No activity & & Behaviour in state inactive \\
\hline & - Disable write to \%MBB0.x from & WORD(0..65535) & 0 & 0 & & Disable write access begining wth byte in area \%MB0.x \\
\hline & - Disable write to \%MB0.x to & WORD(0..65535) & 0 & 0 & & Disable write access ending with byte in area \%MB0.X \\
\hline & - Disable read to \%MB0.x from & WORD(0..65535) & 0 & 0 & & Disable read access beginning with byte in area \%MB0.x \\
\hline & - Disable read to \%MB0.x to & WORD(0..65535) & 0 & 0 & & Disable read access ending with byte in area \%MB0.x \\
\hline
\end{tabular}

\section*{Byte Order}

Format/Endianess for the transmission of WORD values (register) within the request/response telegram (default: "Big Endian").

Port
TCP port on which the server listens.

\section*{Startup Behaviour}

This parameter specifies how the server behaves when configuration data is loaded (e.g. on download). It's default value is "Active". This means the server is immediately addressable after configuration has been performed. In case the server should be activated later on during run time by means of function block ModTcpServOnOff this parameter value has to be set to "No activity". Parameter Behaviour in state inactive then specifies the server's behaviour during the inactive phase.

Behaviour in state inactive

This parameter specifies how the server behaves in inactive state. This state may be set at the very beginning (parameter Startup Behaviour = "No activity") and/or requested during run time calling function block ModTcpServOnOff. It's default value is "No activity". This means the server is not addressable at all (no listening socket on TCP/IP) when it is inactive. Using this setting, any requests by Modbus TCP clients lead to the result Failed to connect to Server or Timeout. All other parameter values make the server respond with an exception code to any requests by Modbus TCP clients.

The presentation of the icon next to the "Modbus TCP/IP Server" in the device tree depends on the state of the server:

\section*{Inactive}


\section*{Active}
- 50 Ethernet (Ethernet) - 6 ETH1 (IP Settings)

Modbus_TCP_IP_Server

\section*{Attention:}

Exception code 9 is actually not defined by Modbus specification. This may cause problems using a different Modbus TCP client than AC500 V3.

Disable
\begin{tabular}{|l|l|l|l|}
\hline Parameter & Default & Value & Description \\
\hline Disable write to \%MB from & 0 & \(0 \ldots 65535\) & \begin{tabular}{l} 
Disable write access starting at \\
\(\% \mathrm{MBx}\)
\end{tabular} \\
\hline Disable write to \%MB to & 0 & \(0 \ldots 65535\) & Disable write access up to \%MBx \\
\hline \begin{tabular}{l} 
Disable read from \%MB \\
from
\end{tabular} & 0 & \(0 \ldots 65535\) & \begin{tabular}{l} 
Disable read access starting at \\
\(\% \mathrm{MBx}\)
\end{tabular} \\
\hline \begin{tabular}{l} 
Disable read from \%MBx \\
to
\end{tabular} & 0 & \(0 \ldots 65535\) & Disable read access up at \%MBx \\
\hline
\end{tabular}

It is possible to disable read and/or write access to individual segments. Reading/writing is disabled beginning at the set start address and is valid up to the set end address (inclusive).
« \(\downarrow\) Chapter 6.8.2.10.1 "Protocol description" on page 4523

\subsection*{6.3.4.3.3 Configuration of Modbus TCP/IP client}

\section*{Adding a Modbus TCP/IP client to the device tree}

Depending on a server's IP address the client sends it's requests via the Ethernet interfaces available.

矢 Chapter 6.8.2.10.1 "Protocol description" on page 4523

\section*{Setting the Modbus TCP/IP client does not have any parameters. parameters of Modbus_TCP_IP \({ }^{4}\) Chapter 6.8.2.10.1 "Protocol description" on page 4523}

The "Modbus_TCP_IP_Client" instance has to be added to the common Ethernet client protocols' node. This nō̄e supports max. one instance of Modbus TCP/IP client. Other protocols can be added in parallel.
1. Right click on the node "Protocols" and click "Add object".
\(\Rightarrow\) The window "Add object below: Protocols" appears.
2. Select "Modbus TCP/IP Client" and click "Add object".
\(\Rightarrow\) Node "Modbus_TCP_IP_Client" is added.


Node "Modbus_TCP_(I)

Client

\subsection*{6.3.4.3.4 Modbus on RTU protocol}

Protocol description can be found in the chapter for serial interfaces.
« Chapter 6.3.2.14.2 "Configuring Modbus RTU on serial interface" on page 1588.
Chapter 6.8.2.9.1 "Protocol description" on page 4506

\subsection*{6.3.4.4 NTP/SNTP protocol}

\subsection*{6.3.4.4.1 Introduction of the NTP/SNTP protocol}

AC500 V3 support the NTP and the SNTP protocol ((Simple) Network Time Protocol). Compared to SNTP, the NTP protocol achieves higher accuracy in time synchronization, meeting advanced requirements for accuracy and reliability of a PLC solution.
In case a configured NTP protocol cannot be used, SNTP protocol is used as a fallback solution.

The S500 I/O modules do not support the functionality to create and transfer a timestamp with an event directly at the I/O channel. A workaround is using SoE logging (Sequence of Event) for a root-cause analysis. For a better understanding on how events are processed on a AC500 V3 PLC, refer to the application example AC500 sequence of events with V3.

The protocols NTP and SNTP provide the functionality to synchronize the clock of a PLC to an external time source. The application example AC500 V3 - SNTP demonstrates how to synchronize the PLC time with a time server.
A specification of the (S)NTP protocol itself can be found in the document RFC4330.

The following modes are supported by the implementation of the AC500 V3 PLC:
- (S)NTP client
- (S)NTP server
- (S)NTP client and server

The function block PmSntpInfo can be used to read diagnosis information of the protocol.
Refer to the documentation of the library \(A B B \_P m_{-} A C 500 . l i b\) for further information.
- If a high precision of system time is wanted, use a fully functional NTP server or at least an SNTP server with a high-precision time-source (e.g. DCF-77 receiver). Avoid cascading several levels of (S)NTP server / (S)NTP clients.
- Client requests are normally sent at intervals depending on the frequency tolerance of the client clock and the required accuracy. However, under no conditions requests should be sent at less than one minute intervals (see RFC 4330). Keep that in mind when setting polling-interval of the (S)NTP client, especially if a huge amount of clients use one single server.
- Be sure not to use broadcast or multicast addresses as server or backupserver since current (S)NTP implementation does not support manycast mode.

\subsection*{6.3.4.4.2 (S)NTP client configuration}


Implementation of (S)NTP client and (S)NTP server is based on protocol version 4.

For (S)NTP client configuration add a new object "SNTP Client" under "Protocols (Client Protocols)".
For a PLC only one instance of an (S)NTP client is possible.


The following parameters are available:

Not all parameters are shown in the user interface.
It should not be necessary to change the default values of the other parameters for the most applications.

But there is the possibility to edit them in the generic parameter editor.

Server 1
\begin{tabular}{|l|l|l|l|}
\hline Parameter & Default & Value & Description \\
\hline IP address & 0.0 .0 .0 & Valid IP address & \begin{tabular}{l} 
IP address of a server which is \\
used as external time source.
\end{tabular} \\
\hline Minimum polling rate & \begin{tabular}{l}
\(2^{\wedge} 6=64 \mathrm{~s}\)
\end{tabular} & \(-4 \ldots 24\) & \begin{tabular}{l} 
Specifies the lower limit of the \\
polling rate. \\
It is calculated as power of 2 and \\
has the unit [s]. \\
The actual polling rate is deter- \\
mined by the protocol itself but it \\
will not be lower than this limit.
\end{tabular} \\
\hline Maximum polling rate & \begin{tabular}{ll}
10 \\
\(2^{\wedge 10 ~=1024 ~ s ~}\) & \(0 \ldots 24\) \\
\begin{tabular}{l} 
Specifies the upper limit of the \\
polling rate. \\
It is calculated as power of 2 and \\
has the unit [s]. \\
The actual polling rate is deter- \\
mined by the protocol itself but it \\
will not be higher than this limit.
\end{tabular} \\
\hline
\end{tabular} \\
\hline
\end{tabular}

\section*{Server 2}
\begin{tabular}{|l|l|l|l|}
\hline Parameter & Default & Value & Description \\
\hline Enable & FALSE & TRUE or FALSE & Enable server \\
\hline IP address & 0.0 .0 .0 & Valid IP address & \begin{tabular}{l} 
IP address of a server which is \\
used as external time source.
\end{tabular} \\
\hline Minimum polling rate & 6 & \(-4 \ldots 24\) & \begin{tabular}{l} 
Specifies the lower limit of the \\
polling rate. \\
It is calculated as power of 2 and \\
has the unit [s]. \\
The actual polling rate is deter- \\
mined by the protocol itself but it \\
will not be lower than this limit.
\end{tabular} \\
\hline Maximum polling rate & 10 & \(2^{\wedge 10=1024 \mathrm{~s}}\) & \begin{tabular}{l} 
Specifies the upper limit of the \\
polling rate. \\
It is calculated as power of 2 and \\
has the unit [s]. \\
The actual polling rate is deter-
\end{tabular} \\
mined by the protocol itself but it. \\
will not be higher than this limit.
\end{tabular}

Time jumps
\begin{tabular}{|l|l|l|l|}
\hline Parameter & Default & Value & Description \\
\hline Enable & TRUE & TRUE or FALSE & Enables the option 'Time jumps' \\
\hline Time jump threshold & 1.0 s & \(0 \ldots 3.403 \mathrm{e}+38\) & \begin{tabular}{l} 
Specifies the threshold value for \\
time steps in seconds
\end{tabular} \\
\hline Limit & -1 & \begin{tabular}{l}
\(-1 \ldots\) \\
2147483647
\end{tabular} & \begin{tabular}{l} 
Number of first clock updates \\
after that this option is deactivated, \\
a negative value activates this \\
option permanently.
\end{tabular} \\
\hline
\end{tabular}

\section*{Max change}
\begin{tabular}{|l|l|l|l|}
\hline Parameter & Default & Value & Description \\
\hline Enable & TRUE & TRUE or FALSE & Enables the option 'Max change' \\
\hline \begin{tabular}{l} 
Maximum change \\
offset
\end{tabular} & 3600 s & \(0 \ldots 4294967295\) & \begin{tabular}{l} 
Maximum allowed clock offset in \\
seconds
\end{tabular} \\
\hline Start after & 3 & \(0 \ldots 4294967295\) & \begin{tabular}{l} 
Specifies the number of first clock \\
updates after that this option is \\
activated
\end{tabular} \\
\hline Ignore after & -1 & \begin{tabular}{l}
\(-1 \ldots\) \\
2147483647
\end{tabular} & \begin{tabular}{l} 
Specifies the number of ignored \\
clock updates which exceed the \\
maximum offset. \\
The protocol will be stopped when \\
this value will be exceed. \\
It is never stopped when a negative \\
value is set.
\end{tabular} \\
\hline
\end{tabular}

\subsection*{6.3.4.4.3 (S)NTP server configuration}

Implementation of (S)NTP client and (S)NTP server is based on protocol version 4.

For (S)NTP server configuration add a new object "SNTP Server" under of the available "Ethernet interfaces (ETH1-ETHn)".

For a PLC only one instance of an (S)NTP server is possible.


The following parameters are available:

Not all parameters are shown in the user interface.
It should not be necessary to change the default values of the other parameters for the most applications.

But there is the possibility to edit them in the generic parameter editor.

Access control
\begin{tabular}{|l|l|l|l|}
\hline Parameter & Default & Value & Description \\
\hline Enable & FALSE & TRUE or FALSE & Enables Access Control \\
\hline Network address & 0.0 .0 .0 & Valid IP address & Network address of allowed clients \\
\hline Subnet mask & 24 & \(8 \ldots 32\) & \begin{tabular}{l} 
Subnet mask of the network \\
address
\end{tabular} \\
\hline
\end{tabular}

Local server
\begin{tabular}{|l|l|l|l|}
\hline Parameter & Default & Value & Description \\
\hline Enable & FALSE & TRUE or FALSE & \begin{tabular}{l} 
This option enables the protocol to \\
run as local server. \\
That means without synchroniza- \\
tion to an external time source.
\end{tabular} \\
\hline Stratum & 10 & \(1 \ldots 15\) & \begin{tabular}{l} 
Stratum of the server \\
when it is used as local server
\end{tabular} \\
\hline Distance & 1 s & \(0 \ldots 3.403 \mathrm{e}+38\) & \begin{tabular}{l} 
Distances in seconds of the server \\
when it is used as local server
\end{tabular} \\
\hline Orphan & FALSE & TRUE or FALSE & \begin{tabular}{l} 
Enables or disables the orphan \\
mode
\end{tabular} \\
\hline
\end{tabular}

\subsection*{6.3.4.5 FTP server}

As of SystemFW 3.1.0 the FTP server is listening only on the Ethernet interface, which the protocol is configured on. It is not possible to have an FTP server on both Ethernet interfaces.

AC500 V3 PLCs only support explicit authorization. AC500 V3 PLCs do not support implicit authorization.
1. Under "Ethernet -> ETH [1,2,...]" add a new object and select "FTP Server" from the list.
2. Double-click the "FTP_Server" item to open FTP server configuration and change the default settings of the parameters, if required.
\begin{tabular}{|l|l|l|l|l|}
\hline \multicolumn{2}{|l|}{ Parameter } & Default & Value & Description \\
\hline \multicolumn{2}{|l|}{ FTP Server } & Port & 21 & 21 \\
\hline Sessions & 1 & \(1 \ldots 4\) & \begin{tabular}{l} 
Do not change the default setting. The parameter \\
specifies the port which is used to connect to the \\
FTP server on the PLC.
\end{tabular} \\
\hline Passwords & - & \begin{tabular}{l} 
Enter the max. number of allowed simultaneous \\
and parallel connections to the FTP server. Each \\
session uses one socket. Note: Some FTP cli- \\
ents require several connections to work.
\end{tabular} \\
\hline & system & - & - & \begin{tabular}{l} 
Set each user's passwords for login. No entry \(=\) \\
no password.
\end{tabular} \\
\hline & sdcard & - & - & System RAM disk \\
\hline & userdisk & - & - & Inserted memory card. \\
\hline & flashdisk & - & - & User section of the flash disk. \\
\hline
\end{tabular}

\subsection*{6.3.4.6 MQTT client protocol}

\subsection*{6.3.4.6.1 System technology}

\section*{NOTICE!}

If the MQTT client protocol is enabled, we recommend using the secure (TLS) variant of the MQTT protocol for protection of information.
* Chapter 6.1 "Cyber security" on page 1340

The MQTT protocol is a lightweight communication protocol which is widely used on the internet to connect embedded device to the cloud.
The MQTT (Message Queuing Telemetry Transport) client library allows to integrate an AC500 processor module to act as a client in the MQTT protocol. Thus, it is possible to exchange data between the AC500 and other devices connected to the MQTT network.
In the figure below, there is an MQTT network with one broker (MQTT broker in the middle) and five clients. The figure shows the main functions of MQTT to send and receive data: publish and subscribe. The clients can publish messages with a specific topic to send data (e. g. the temperature of a connected sensor with a timestamp) to the MQTT broker. For example, the client "AC500_1" publishes a message to topic "topic/2". On the other hand side clients can also subscribe to topics to receive data. For example, the client "Laptop" has subscribed topic "topic/2". So all messages with the topic "topic/2" which has been published to the MQTT broker will be sent immediately to the client "Laptop". This creates a message flow from the client "AC500_1" to the laptop.


To realize the MQTT behavior, there are several function blocks implemented in the \({ }^{*}{ }^{4}\) Chapter 6.5.11 "MQTT client library" on page 4048.

Table 376: Function blocks overview
\begin{tabular}{|l|l|}
\hline Function Block & Description \\
\hline \begin{tabular}{l} 
MqttConnectWithCertBuffer \\
MqttConnectWithCertFile
\end{tabular} & \begin{tabular}{l} 
Every MQTT use case starts with establishing a connection to \\
an MQTT broker. Therefore, a connection structure needs to be \\
created. The connection structure is used to identify the connec- \\
tion for subsequent operations like publish or subscribe. \\
It is possible to establish an SSL connection. Using an SSL \\
connection, at least a certificate for the server is needed. Certif- \\
icates can be loaded from a buffer (program variable) or a file \\
which is stored on the PLC.
\end{tabular} \\
\hline \begin{tabular}{l} 
MqttGetReceivedPacket \\
MqttPing MqttPublish \\
MqttSubscribe \\
MqttUnsubscribe
\end{tabular} & \begin{tabular}{l} 
These function blocks can be used on an established MQTT \\
connection to realize the desired use case.
\end{tabular} \\
\hline MqttDisconnect & This function block is the end of each use case. \\
\hline
\end{tabular}

One MQTT send use case could look like this:

*) It makes sense for several publish messages in a row (e. g. one message per second) not always open a new connection.
One MQTT receive use case could look like this:
\begin{tabular}{|c|c|c|c|c|c|}
\hline MqttConnectWith CertBuffer(FB) & MqttSubscribe (FB) & MqttGetReceived Packet (FB) & MqttGetReceived Packet (FB) & MqttUnsubscribe (FB) & MqttDisconnect (FB) \\
\hline At first, a connection to an MQTT broker is needed. & For getting all messages to a specific topic, a subscription is necessary. & The client gets a message from the broker. & \(\cdots\) & If a specific topic is no longer important, the broker will be informed with an unsubscribe message. & In the end, the device disconnects from the broker. \\
\hline
\end{tabular}

TLS version The MQTT client uses the TLS version 1.2.

Configuration in For the MQTT client no configuration is needed.
Automation
Builder
Configuration in All function blocks have to be called in tasks with cyclically processing. CODESYS

You can use the function blocks with:
- PLC_PRG with automatic task configuration or manual task configuration.
- One single program or different programs.
- One single task or different tasks.

With different programs assigned to different tasks you can define different cycle times and priorities.

Limitations - No persistent session. After an interrupted connection, the client needs to subscribe on topics again in case of reconnect.
- One connection (MQTT_CONNECTION) cannot be shared between multiple tasks. Different connections can be used by different tasks or even within the same task.
- Only one FB can operate on a single connection at the same time. Always wait for the FB to complete before calling the next FB. To use two different FB's in parallel (like publish and receive) it is necessary to have two different connections, otherwise they must be called one after the other.

Hardware The MQTT protocol requires AC500 devices with integrated Ethernet.

\subsection*{6.3.4.6.2 Application examples}

MQTT can be used with an MQTT client library or JSON. Application examples are available for both methods, which demonstrate the practical implementation:

JSON program- An introduction to programming with JSON is given in the application example ming JSON programming.

MS Azure cloud How to set-up a cloud-based condition monitoring system with AC500 and the MS Azure cloud is described in the application example PLC configuration and MS Azure cloud.
\(\begin{array}{ll}\text { Setting up an } & \text { How to set-up a local MQTT message broker is described in the application example } \\ \text { MQTT message } & \text { PLC configuration and Mosquitto message broker. }\end{array}\) MQTT message PLC configuration and Mosquitto message broker. broker

Connecting an How to connect an MQTT message broker with an AC500 PLC is described in the application MQTT message example Connection to Mosquitto message broker. broker

Use cases and how to handle the function blocks of a certain library is described in sample projects. After the respective library has been installed the corresponding sample projects are available in the default path AutomationBuilder Examples. If the default path is inaccessible, click "Help \(\rightarrow\) Project examples" in the Automation Builder menu.

\subsection*{6.3.4.7 AC500 V3 secure protocols}

\subsection*{6.3.4.7.1 Introduction}

The following protocols can be secured using certificates:
- Communication between Automation Builder and the PLC (e.g. Programming, Monitoring)
- Communication between the PLC's webserver and visualization clients (browsers)
- Communication between the PLC's FTP server and FTP clients
- Communication between the OPC UA server and OPC UA clients

As a prerequisite to enable secure communication on one or more protocols, the required certificates need to be present on the PLC.

For security reasons \(A B B\) does not encourage the use of self-signed certificates. \(A B B\) shall not be held liable for any damage or loss that arises due to the use of self-signed certificates on AC500 PLCs.
Self-signed certificates protect against eavesdropping if used correctly. They do not offer any secure means of authentication.

\subsection*{6.3.4.7.2 Certificate handling}

Automation Builder offers a convenient "Security Screen" to manage certificates on connected PLCs.


It can be accessed through the shield icon on the lower right corner of the main window:


Use the tab "Devices" to manage certificates on the PLC.
It offers to:
- show certificates available on the PLC
- import and export certificates
- create new (self-signed) certificates
- trust or untrust certificates

\subsection*{6.3.4.7.3 Configuring secure protocols}

\section*{Encrypted communication between Automation Builder and the PLC}

Via tab "Communicatio n Settings"

闻 PLC_AC500_V3 x
\begin{tabular}{|c|c|c|c|}
\hline Communication Settings & Scan Network... Gateway - & \multicolumn{2}{|l|}{Device -} \\
\hline PLC Settings & & Add current device to favorites Manage favorite devices... & \\
\hline \begin{tabular}{l}
Version information \\
Statistics
\end{tabular} & & Rename active device... Wink active device Send echo service... & \\
\hline Files
\[
\log
\] &  & Store communication settings in project Confirmed online mode Filter network scans by target ID & \\
\hline PLC Shell & & Encrypted communication & - \\
\hline Users and Groups & & IP-Address: localhost & \\
\hline
\end{tabular}

Via "Security
Screen" in
Automation Builder

Security Screen \(\times\)

Digital signature
\begin{tabular}{l}
\begin{tabular}{|c|cc|c|c|c|}
\hline\(\times\) & Issued for & Issued by & Valid from & Valid until & Thumbprint \\
\hline\(\times\) & Issued for & Issued by & Valid from & Valid until & Thumbprint \\
\hline
\end{tabular} \\
Project file decryption \\
\hline
\end{tabular}

Security level
Activate the use of certificates for enhanced security:
( Enforce encrypted communication
\(\square\) Enforce encryption of project files

Install a trusted certificate

Ensure the PLC clock is set to the current time and date when using certificates on the PLC. Otherwise the certificate cannot be used to secure a protocol (see also \({ }^{\star}\) Chapter 6.8.2.3.2 "AC500 battery" on page 4445 and \(\Leftrightarrow\) Chapter 6.3.4.4.2 "(S)NTP client configuration" on page 1718).

When trying to log in or when you set the PLC as active path, there will be a one-time pop-up asking you to add the PLC's certificate to the trusted certificates:


After trusting the PLC's certificate, the communication between the Automation Builder and the PLC is now encrypted.
This is shown by additional yellow lines around the communication path on the "Communication Settings" page.

\section*{Secure web server}

\section*{- NOTICE!}

We recommend configuring an HTTPS connection (encryption with SSL/TLS) between the visualization clients and the web server to protect the system from unwanted access.
② Chapter 6.1 "Cyber security" on page 1340
1. Generate or import a certificate for the web server

Ensure the PLC clock is set to the current time and date when using certificates on the PLC. Otherwise the certificate cannot be used to secure a protocol (see also \(\stackrel{y}{ }\) Chapter 6.8.2.3.2 "AC500 battery" on page 4445 and \({ }^{4}\) chapter 6.3.4.4.2 "(S)NTP client configuration" on page 1718).
2. Attach a web server node to either ETH1 or ETH2 or both and configure security mode.
\(\Rightarrow\) This will automatically insert a visualization into the project.
The available modes of operation are:
- http only
- https only
- Both (http and https)
- Redirect http to https
3. Download and set the PLC to RUN.
4. Connect to the web server using the configured method: https://<your PLC's IP address>/ webvisu.htm.

In case you are using a self-signed certificate, your browser will show some warnings.

If you are aware of the risks of self-signed certificates, this can be ignored.
を) Further information on page 1724

\section*{Secure FTP}
1. Import a certificate to the PLC for FTP or create a self-signed certificate.

Ensure the PLC clock is set to the current time and date when using certificates on the PLC. Otherwise the certificate cannot be used to secure a protocol (see also \(\stackrel{y}{ }\) Chapter 6.8.2.3.2 "AC500 battery" on page 4445 and \({ }^{\Perp}\) Chapter 6.3.4.4.2 "(S)NTP client configuration" on page 1718).
2. Add an FTP server to either ETH1 or ETH2
3. Set the parameter "Security Mode" to either "BOTH" or "FTPS only". \(\Rightarrow\) You can use any FTP client that supports FTPS explicit mode (FTPES).

OPC UA secure
In case you are using a self-signed certificate, the FTP client will show some warnings or notice that it does not know the certificate and wants you to check it.

③ Further information on page 1724

OPC UA uses mutual authentication, which means that both partners must have their own certificate and know the other's certificate, before being able to establish a connection!

1. Create a new certificate in your OPC UA client.

Ensure the PLC clock is set to the current time and date when using certificates on the PLC. Otherwise the certificate cannot be used to secure a protocol (see also \({ }^{4}\) Chapter 6.8.2.3.2 "AC500 battery" on page 4445 and \(\stackrel{\xi}{ }{ }^{\circ}\) Chapter 6.3.4.4.2 "(S)NTP client configuration" on page 1718).
2. Import that certificate to the "Trusted Certificates" in your PLC using the "Security Screen".
3. Import a certificate for the OPC UA server on the PLC or create a self-signed certificate.
4. Export that cert to the PC and provide it as a trusted certificate to your OPC UA client.
5. Reboot the PLC and check that it is in RUN and both certificates are on the PLC (via the "Security Screen").
6. Add the PLC as OPC UA server in your OPC UA client.
7. Connect to the OPC UA Server.
\(\Rightarrow\) You can interact normal with the UA server.

In case you are using a self-signed certificate, you will see some warning message (depending on the OPC UA client).
If you are aware of the risks of self-signed certificates, this can be ignored.
\& Further information on page 1724

The certificate warnings will disappear only when using a certificate from a trusted certification authority or a certificate derived from this by an intermediate certification authority (e.g. a company CA).
That process is done via PLCShell command "cert-createcsr", then getting the file from the PLC via the filebrowser tab in "cert/export" and getting that signing request turned into a real certificate by a certification authority.
Import the certificate generated by your certification authority using the security screen.

\subsection*{6.3.4.8 KNX configurator}

\subsection*{6.3.4.8.1 Introduction}

Refer to the general description for information about the following tabs of the device editor.
- 乡 Chapter 6.4.1.21.2.8.12 "Tab '<device name> I/O Mapping'" on page 2444
- \(\quad\) ② Chapter 6.4.1.21.2.8.13 "Tab '<device name> IEC Objects'" on page 2449
- \({ }^{\sharp}\) Chapter 6.4.1.21.2.8.4 "Tab 'Parameters'" on page 2433
- \(\quad\) Chapter 6.4.1.21.2.8.19 "Tab 'Status'" on page 2460
- \(\quad \stackrel{y}{c}\) Chapter 6.4.1.21.2.8.20 "Tab 'Information'" on page 2460

Only in the case of special features is there an additional help page for the specific device editor.
If the "<device name> Parameters" tab is not shown, then select the "Show generic device configuration editors" option in the CODESYS options ("Device Editor" category).

With the KNX editor from CODESYS, you define the communication objects of your building automation. The communication objects are exported and made available to the ETS5 program. Linking the communication objects to the different KNX devices is performed exclusively in the ETS5 program. Therefore, only the objects are generated in CODESYS. The objects are linked to variables from the PLC program by means of "I/O mapping".

You add an Ethernet adapter below the controller. Then you add the KNX device below the adapter. You can insert only one KNX device per controller.


See also
- \({ }^{4}\), Chapter 6.4.1.21.4.14.7 "Dialog 'Options' - 'Device Editor"' on page 2786

\subsection*{6.3.4.8.2 ETS5 Software - 'DCA' Plug-In}

Linking the communication objects of the different KNX devices is performed exclusively in the ETS5 program. To do this, you need the ETS5 software (light or professional version). You also need the KNX product file available from KNX.

\section*{Programming steps}
1. Create a project in CODESYS.
2. Download the CODESYS project to the controller.
\(\Rightarrow\) The checksum (CRC) is also downloaded to the controller.
3. Create an export file in CODESYS.
\(\Rightarrow\) The checksum (CRC) is also saved in the export file.
4. Read the export file into the configuration of ETS5.
5. Parameterize the objects in ETS5.
6. Start the program on the controller.
7. Transfer the KNX configuration to the controller.
\(\Rightarrow\) The checksum (CRC) is also transferred. The runtime system checks whether or not both checksums match. When they match, the KNX device is identified as functional by the green arrow. If not, then an error is issued in the logger. In case of error, the process data (inputs/outputs) is not updated.

\subsection*{6.3.4.8.3 Tab 'KNX - General'}

Object: KNX
The tab in the configurator of the KNX editor shows an overview of all communication objects. The I/Os of the communication objects are applied automatically to the I/O mapping.
Entries can be edited directly in the table or in the "Communication object" dialog. Existing entries can be copied via copy\&paste. The next free channel number is used automatically in this case.

Table 377: "Address settings"
\begin{tabular}{|l|l|}
\hline "Add" & Opens the "Communication object" dialog for adding objects \\
\hline "Edit" & Opens the "Communication object" dialog for editing objects \\
\hline "Delete" & Deletes the selected communication objects \\
\hline "Export to ETS" & \begin{tabular}{l} 
Exports the list of communication objects in an XML file. This file can be \\
imported by ETS5 if ETS5 has the DCA plug-in installed. \\
Note: The command is also available in the context menu when the KNX node is \\
selected.
\end{tabular} \\
\hline "Export CSV" & Exports the communication objects to a CSV file \\
\hline "Import CSV" & Imports the communication objects from a CSV file \\
\hline "Identification" & \begin{tabular}{l} 
Checksum (CRC) of the communication object. This must be identical to the \\
checksum in ETS5.
\end{tabular} \\
\hline
\end{tabular}

\section*{Dialog 'Commu- \\ nication object'}
\begin{tabular}{|l|l|}
\hline "Group Object Number" & \begin{tabular}{l} 
Unique channel number. Gaps in the numbering is permitted. \\
If the channel number is already assigned, then an error text is displayed and the \\
"OK" button is disabled.
\end{tabular} \\
\hline "Type" & \begin{tabular}{l} 
Determines whether or not the object in CODESYS is used as "Input" or \\
"Output".
\end{tabular} \\
\hline "Data point type" & \begin{tabular}{l} 
The data types (DPT = Data Point Types) are specified in the KNX standard. \\
In CODESYS, a selection of the most common data types is available. Only the \\
basic data type can be selected, without units (for example, DPT9.*).
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Group Object Name" & \begin{tabular}{l} 
Any object name. Depending on the data type, a predefined text is automatically \\
added.
\end{tabular} \\
\hline "Group Object Function" & \begin{tabular}{l} 
Any function name. Depending on the data type, a predefined text is automati- \\
cally added.
\end{tabular} \\
\hline "Watchdog Timeout" & \begin{tabular}{l} 
If a new message has not been received after this elapsed time, then the \\
Timeout status bit is set.
\end{tabular} \\
\hline
\end{tabular}

\subsection*{6.3.4.8.4 Tab 'I/O Mapping'}

Object: KNX
The I/O channels are generated for each communication object:

Table 378: General I/Os
\begin{tabular}{|l|l|}
\hline Signal & Description \\
\hline "Program LED Status" & \begin{tabular}{l} 
This input is set by the ETS program. The signal can be used for identifying a special \\
controller when several controllers are used (for example, by switching a LED). \\
The status is also set when the "Program Button" is set to TRUE. Then the device is \\
in programming mode. As soon as ETS5 has successfully set the physical address, \\
this input switches to FALSE.
\end{tabular} \\
\hline "Program Button" & \begin{tabular}{l} 
The "Program Button" is needed for assigning the physical address from ETS5. If the \\
output is set to TRUE, then the device is in programming mode and then ETS5 can \\
assign the address specified there to the device.
\end{tabular} \\
\hline
\end{tabular}

Table 379: I/O channels of the communication object
\begin{tabular}{|l|l|}
\hline Signal & Description \\
\hline "Status byte" & \begin{tabular}{l} 
Status byte as defined in the KNX stack. This allows you to determine in the applica- \\
tion whether or not data has been received. The status can be reset by means of the \\
ResetStatusFlags method.
\end{tabular} \\
\hline \begin{tabular}{l} 
"Trigger/Disable Cyclic, \\
send on change" \\
"Trigger Output"
\end{tabular} & \begin{tabular}{l} 
Depending on the configuration in the ETS program, this output has the following \\
function: \\
- If at least one of the options "send on difference", "send on change", or "Cyclic \\
sending" is enabled, then the output is defined as deactivation. If it is set to \\
TRUE, then cyclic sending or send on change is stopped. \\
If none of the options "send on difference", "send on change", or "Cyclic sending" \\
is enabled, then sending is triggered by a rising edge.
\end{tabular} \\
\hline "Value" & \begin{tabular}{l} 
Value for the input or outputs - depending on the corresponding communication \\
object.
\end{tabular} \\
\hline
\end{tabular}

\subsection*{6.3.4.8.5 ETS5 - Tab 'Parameter'}

The parameter page of the ETS5 configuration software is available only after you have imported the CODESYS configuration file. The parameter page is where you define the sending behavior of the values.

Table 380: "General Information"
\begin{tabular}{|l|l|}
\hline "Default gateway" & Default gateway for sending \\
\hline "Telegram rate" & \begin{tabular}{l} 
Sending rate of telegrams \\
Note: This restriction should be applied in exceptional cases only, because this \\
causes the reaction times to be extended.
\end{tabular} \\
\hline "Project title" & \begin{tabular}{l} 
In CODESYS, these parameters can be defined in the project information. They \\
are imported to ETS5 in the XML file and displayed here.
\end{tabular} \\
\hline "Application date" & \begin{tabular}{l} 
"Identifier": checksum (CRC) of the configuration. The checksum is also dis- \\
played in CODESYS and must be identical to the checksum displayed here so \\
that communication can be started.
\end{tabular} \\
\hline "Identifier" & \\
\hline "Version" & \\
\hline "Dpplication state" &
\end{tabular}

The objects are subdivided into groups of ten (1 .. 10, 11 .. 20, 21 .. 30, etc.). A maximum of 1000 communication objects is possible.

Table 381: "Object 1 .. 10"
\begin{tabular}{|l|l|}
\hline "<type>" & \begin{tabular}{l} 
Type of the object \\
The parameter cannot be changed.
\end{tabular} \\
\hline "Communication direction" & \begin{tabular}{l} 
"Output (PLC to KNX)": The value is sent from the CODESYS controller to the \\
KNX object. For this communication direction, more settings are possible ("Send \\
condition", etc.). \\
"Input (KNX to PLC)": The value is sent to the CODESYS controller. \\
The parameter cannot be changed.
\end{tabular} \\
\hline "Send condition" & \begin{tabular}{l} 
"No automatic sending": No sending when value is changed \\
"Send on change": Sending each time value is changed \\
"Send on difference": Sending when the change in value corresponds to at least \\
the value for "Sending difference".
\end{tabular} \\
\hline "Sending difference" & \begin{tabular}{l} 
Requirement: "Send condition" is "send on difference". \\
The value is passed when its change is at least this value.
\end{tabular} \\
\hline "Cyclic sending" & \begin{tabular}{l} 
"Disable": No cyclic sending \\
"Enable (seconds)", "Enable (minutes)": Cyclic sending - regardless of the "Send \\
condition".
\end{tabular} \\
\hline "Cycle time [hh:mm:ss]" & \begin{tabular}{l} 
Rate for cyclic sending (in hours/minutes/seconds) \\
Requirement: "Cyclic sending" is set to "Enable (seconds)".
\end{tabular} \\
\hline "Cycle time [hh:mm]" & \begin{tabular}{l} 
Rate for cyclic sending (in hours/minutes) \\
Requirement: "Cyclic sending" is set to "Enable (minutes)".
\end{tabular} \\
\hline
\end{tabular}

\subsection*{6.3.4.9 BACnet-BC}

\subsection*{6.3.4.9.1 Introduction to BACnet}

BACnet is a standardized data communication protocol for Building Automation and Control networks as defined in the ANSI/ASHRAE standard 135 and ISO 16484-5.

The advantage is interoperability between devices of different vendors.
The BACnet protocol defines services to allow communication between devices. Examples include 'Who is', 'I am', 'Who has' and 'I have' for device and object search and identification, "Read Property" and "Write Property" for the exchange of data, up to more complex services for alarm and event management, scheduling and trending.

The BACnet protocol defines a number of object types on which the services operate. Each object is characterized by its properties.

The BACnet objects are combined in a BACnet device. A BACnet device represents the functionality of a physical device.

More background information and introduction can be found here:
http://www.bacnet.org
http://www.bacnet.org/Bibliography

\subsection*{6.3.4.9.2 AC500 and BACnet}

A BACnet device can be described by its "BACnet Interoperability Building Blocks" (BIBB)s, which are needed to establish services. They are grouped in different areas:
- "Data Sharing" (DS)
- "Alarm and Event Management"(AE)
- "Scheduling" (SCHED)
- "Trending" (T)
- "Device and Network Management" (DM)
"Data Sharing" for example contains two BIBBs which are needed for the "Service Read Property":
- Client side: DS-RP-A (Data Sharing - Read Property - A)
- Server side: DS-RP-B (Data Sharing - Read Property - B)

The BACnet standard defines profiles by the minimum required BIBBs, see table below. "BACnet Simple Sensor" (B-SS) is the simplest one, only containing one BIBB. More complex devices contain more BIBBs (from right to left).
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline & & CP600 & \multicolumn{2}{|l|}{AC500 V3} & \multicolumn{3}{|l|}{AC500 V2} \\
\hline \multirow[t]{2}{*}{Interoperability Areas (IA)} & \multicolumn{2}{|l|}{BACnet Devicep-profiles !} & & & & & \\
\hline & B-OWS & B-OD & B-BC & B-AAC & B-ASC & B-SA & B-SS \\
\hline \multirow{6}{*}{Data Sharing} & DS-RP-A, B & |DS-RP-A, B & DS-RP-A, B & DS-RP-B & DS-RP-B & DS-RP-B & DS-RP-B \\
\hline & DS-RPM-A & IDS-RPM-A, B & DS-RPM-A, B & DS-RPM-B & & & \\
\hline & DS-WP-A & iDS-WP-A, B & DS-WP-A, B & DS-WP-B & DS-WP-B & DS-WP-B & \\
\hline & DS-WPM-A & 1 & DS-WPM-B & DS-WPM-B & & & \\
\hline & & DS-V-A & & & & & \\
\hline & & DS-M-A & & & & & \\
\hline \multirow{5}{*}{\begin{tabular}{l}
Alarm \& \\
Event \\
Management
\end{tabular}} & AE-N-A & AE-N-A & AE-N-I-B & AE-N-I-B & & & \\
\hline & AE-ACK-A & 1 & AE-ACK-B & AE-ACK-B & & & \\
\hline & AE-INFO-B & 1 & AE-INFO-B & AE-INFO-B & & & \\
\hline & AE-ESUM-A & 1 & AE-ESUM-B & & & & \\
\hline & & AE-VN-A & & & & & \\
\hline Scheduling & SCHED-A & 1 & SCHED-E-B & SCHED-E-B & & & \\
\hline \multirow[b]{2}{*}{Trending} & T-VMT-A & I & T-VMT-I-B & & & & \\
\hline & T-ATR-A & 1 & T-ATR-B & & & & \\
\hline \multirow{7}{*}{Device \& Network Management} & DM-DDB-A, B & \({ }_{1}\) DM-DDB-A, B & DM-DDB-A, B & DM-DDB-B & DM-DDB-B & & \\
\hline & DM-DOB-A, B & DM-DOB-B ! & DM-DOB-B & DM-DOB-B & DM-DOB-B & & \\
\hline & DM-DCC-A & 1 & DM-DCC-B & DM-DCC-B & DM-DCC-B & & \\
\hline & DM-TS-A & 1 & DM-TS-B or DM-UTC-B & DM-TS-B or DM-UTC-B & & & \\
\hline & DM-UTC-A & 1 & & & & & \\
\hline & DM-RD-A & 1 & DM-RD-B & DM-RD-B & & & \\
\hline & DM-BR-A & 1 & DM-BR-B & & & & \\
\hline
\end{tabular}

The AC500 V2 supports BIBBs qualifying it as "BACnet Application Specific Controller" (BASC), by installing the BACnet B-ASC library.
AC500 V3 supports many more BIBBs qualifying it as "BACnet Building Controller" (B-BC), which contains a server (all BIBBs ending with -B) and a client (all BIBBs ending with -A). In fact, the AC500 contains some more BIBBs. All BIBBs under B-BC in the table above, plus:
\begin{tabular}{ll} 
DS-COV-A, -B & (Change of Value-A, -B) \\
DS-COVP-A, -B & (Change of Value of Properties-A, -B) \\
AE-N-E-B & (Alarm and Event-Notification External-B) \\
AE-ASUM-B & (Alarm and Event-Alarm Summary-B) \\
SCHED-I-B & (Scheduling-Internal-B) \\
T-VMT-E-B & (Viewing and Modifying Trends External-B) \\
DM-TS-B & (Time Synchronization-B) \\
DM-UTC-B & (UTC Time Synchronization-B) \\
DM-MTS-A & (Manual Time Synchronization-A) \\
DM-LM-B & (List Manipulation-B) \\
DM-OCD-B & (Object Creation and Deletion-B) \\
NM-BBMDC-B & (BBMD Configuration-B)
\end{tabular}

A list with all details can be found in the Automation Builder pdf document ABB-B-BC-PICSAC500_V3.pdf. Direction: Help/Project examples/Examples.
The figure below shows a typical application for an AC500 V3, acting as B-BC.


A drive with several actuators and sensors is acting as B-ASC, for example providing a temperature value as "Analog Input" (AI) object on the MS/TP network.
(y) Chapter 6.3.4.9.3.2 "Supported BACnet networks " on page 1736

AC500 B-BC as client can read this temperature value, perform some processing (scaling, limit check) and on the server side provide the processed value as "Analog Value" (AV) object and as "Trend" object on the IP network. Higher level clients like BACnet Operator Workstation (B-OWS) can access the processed objects "Analog Value" and "Trend" for supervision.
The following chapters describe the possible applications and how to configure an AC500 V3 as B-BC.

\subsection*{6.3.4.9.3 AC500 V3 as BACnet Building Controller (B-BC)}

\section*{General}

The BACnet integration into CODESYS implements the ANSI/ASHRAE standard 135-2012 (ISO 16484-5) protocol revision 14 and is based on the AMEV AS-A and AS-B standards. Integration allows access to the properties of BACnet objects and the configuration parameters of a BACnet device by means of an IEC application. You can program a dynamic BACnet configuration and have access to the BACnet functions in the BACnet network by reading and writing BACnet object properties.

\section*{Supported BACnet networks}

BACnet can run on different local area network types. The AC500 B-BC supports the following ones:
- MS/TP (Master Slave / Token Passing), based on serial RS-485
- BACnet IP, based on Ethernet / UDP / IP
\begin{tabular}{|c|c|c|c|c|c|c|c|c|}
\hline \multirow[b]{2}{*}{Layer} & \multicolumn{8}{|c|}{BACnet Internetwork} \\
\hline & & & & & & & & \\
\hline Application & \[
\begin{gathered}
\hline \text { Device } \\
\text { ID } 10 \\
\hline
\end{gathered}
\] & \[
\begin{gathered}
\hline \text { Device } \\
\text { ID } 11 \\
\hline
\end{gathered}
\] & \multicolumn{2}{|c|}{Device
\[
\text { ID } 12
\]} & \[
\begin{gathered}
\hline \text { Device } \\
\text { ID } 13 \\
\hline
\end{gathered}
\] & \multicolumn{2}{|c|}{\[
\begin{gathered}
\text { Device } \\
\text { ID } 14
\end{gathered}
\]} & \[
\begin{gathered}
\hline \text { Device } \\
\text { ID } 15 \\
\hline
\end{gathered}
\] \\
\hline Network & & & \multicolumn{2}{|c|}{BACnet router} & & \multicolumn{2}{|c|}{BACnet router} & \\
\hline Datalink & \begin{tabular}{l}
MSTP \\
BACnet-MAC \\
1
\end{tabular} & MSTP
BACnet-MAC
2 & MSTP
BACnet-MAC
3 & \begin{tabular}{l}
IP \\
BACnet MAC UPD port .47808
\end{tabular} & \begin{tabular}{|c|}
\hline IP \\
BACnet MAC \\
UPD port \\
.47808 \\
\hline
\end{tabular} & \begin{tabular}{l}
IP \\
BACnet MAC UPD port . 47808
\end{tabular} & \begin{tabular}{l}
IP \\
BACnet MAC UPD port .47809
\end{tabular} & \begin{tabular}{l}
IP \\
BACnet MAC UPD port .47809
\end{tabular} \\
\hline Physical & \[
\begin{gathered}
\text { RS485 } \\
19200 \text { baud }
\end{gathered}
\] & \begin{tabular}{c} 
RS485 \\
19200 baud \\
\hline
\end{tabular} & RS485 19200 baud & \[
\begin{gathered}
\hline \text { IIP address } \\
.0 .120 \\
\hline
\end{gathered}
\] & \begin{tabular}{|c|}
\hline +IP address \\
.0 .130 \\
\hline
\end{tabular} & \begin{tabular}{|c|} 
+IP address \\
.0 .140 \\
\hline
\end{tabular} & \[
\begin{gathered}
\hline \text { IIP address } \\
.2 .140 \\
\hline
\end{gathered}
\] & \[
\begin{array}{c|}
\hline \text { +IP address } \\
.2 .150 \\
\hline
\end{array}
\] \\
\hline & \multicolumn{3}{|c|}{\multirow[b]{2}{*}{BACnet network \#1 (MSTP)}} & \multicolumn{3}{|c|}{IP subnet 1: 192.168.0.x} & \multicolumn{2}{|l|}{IP subnet 2: 192.168.2.x} \\
\hline & & & & \multicolumn{3}{|c|}{BACnet network \#2 (IP)} & \multicolumn{2}{|l|}{BACnet network \#3 (IP)} \\
\hline
\end{tabular}

Different networks can be combined to one common "BACnet internetwork". The figure above shows an example of some BACnet devices in one "BACnet internetwork". Each device has a device ID (10 to 15) which must be unique on application level. Services on application level (e.g. read or write request) are working with these device IDs and need no addressing information of the lower levels.

The example "BACnet internetwork" consists of different BACnet networks:
- BACnet MS/TP network connecting device 10, 11 and 12
- BACnet IP network (UDP port 47808), consisting of one IP subnets with IP range 192.168.0.x, connecting device 12, 13 and 14
- BACnet IP network (UDP port 47809), consisting of one IP subnet with IP range 192.168.2.x, connecting device 14 and 15

Addressing in a BACnet network is done through datalinks which must have a unique BACnet MAC address (which is different to an Ethernet MAC address).
- In a MS/TP network the BACnet MAC address is just one octet (1, 2, 3 in the example). * Chapter 6.3.4.9.3.5.5 "Configuration of datalinks" on page 1745
- In an IP network the BACnet MAC address is the combination of the IP address and the UDP port number (for example 192.168.0.130.47808 for device 13). The following 16 UDP ports are reserved for BACnet: BAC0 (=47808 decimal) to BACF.
② Chapter 6.3.4.9.3.5.5 "Configuration of datalinks" on page 1745

To form a common "BACnet internetwork" the single BACnet networks must be combined by BACnet routers. AC500 can act as a BACnet router between BACnet MS/TP and IP networks (device 12 in the figure above) or between two different BACnet IP networks (device 14).
Two IP subnets using the same UDP ports can be combined to one BACnet IP network with an internet router.


The problem is that internet routers block local broadcast messages, which are required for BACnet communication. This can be solved by "Broadcast Management Devices" (BBDM). AC500 V3 can be configured as BBDM. In the figure above the devices 12 and 14 should be configured as BBDM in order to enable the BACnet communication across the internet router.
An alternative is to configure AC500 V3 as foreign BACnet device if an IP subnet contains no BBDM device to pass broadcast messages over internet routers.

Configuring the AC500 as BBDM or foreign device is described in \({ }^{\sharp}\) Chapter 6.3.4.9.3.5.5 "Configuration of datalinks" on page 1745.

\section*{Supported objects and properties}

Communication with BACnet is done through objects and properties.
The AC500 B-BC server of the figure below is represented as a BACnet device object with "ID 12". The device contains more objects like the Analog Input object, representing the input of a temperature measurement device. An object contains several properties, like "ID, Description, Present Value, Unit" etc.
Further possible objects of an AC500 B-BC are:
- "Binary Input" for example from connected to a switch
- "Analog / Binary Output" for actuators
- "Analog / Binary Values" for local variables
- "Calender"
- "Schedule"
- "Trend Log"
- ...
- A list with all details can be found in the Automation Builder pdf document ABB-B-BC-PICSAC500_V3.pdf. Help/Project examples/.


Fig. 273: BACnet objects, properties, services and BIBBs

\section*{Supported BIBBs and services}

While objects and properties describe which data are communicated, the communication itself is done with services between clients and servers. A certain service can only be executed if client and server have the related BIBBs. The Fig. 273 BACnet objects, properties, services and BIBBs shows a simple "Service Read Property" which is possible because the client on the right supports DS-RP-A and the server on the left supports DS-RP-B. The service is executed in two steps:
1. The client initiates a confirmed request "Read Property", asking for the present value of the "Analog Input" of object with "ID 1010".
2. The server answers with an acknowledge, sending the present value which is \(21,89^{\circ} \mathrm{C}\) in the example.

A list of all supported BIBBs and services of AC500 V3 is given in the Automation Builder pdf document ABB-B-BC-PICS-AC500_V3.pdf. Help/Project examples/Examples.

\section*{BACnet configuration in Automation Builder}

\section*{General}

To act as a BACnet server or client, the AC500 must be configured accordingly. The figure below shows the basic configuration of a BACnet server (left) and a BACnet server with client functionality (right). It is also possible to have server and client functionality in parallel.


Following objects need to be created:
1 "BACnet Server" root object. This is the root object for the server functionality, as well as for the client functionality. It is mandatory, even if only client functionality is required. \(\Rightarrow\) Chapter 6.3.4.9.3.5.2 "Configuration of BACnet server root object" on page 1739

2 BACnet server objects, for example "BACnet Analog Input" Temperature. The properties of the objects must be controlled (written or read) by the PLC logic. \({ }^{\circ} \Rightarrow\) Chapter 6.3.4.9.3.5.3 "Adding BACnet server objects" on page 1741
3 BACnet client objects, represented by a different symbol. For example, "BACnet Client Read Property". The functionality of the client objects must be programmed in the PLC logic. Inserting the client objects below the server is optional. It is also possible to instantiate the objects only in a PLC logic. 少 Chapter 6.3.4.9.3.5.4 "Adding BACnet client functionality" on page 1742
4 Datalink for the physical layer. This object links the physical interface (Ethernet IP or serial MS/TP) to the "BACnet Server" object. In the example above the IP address of ETH1 is automatically retrieved by inserting the "BACnet IP datalink" below the ETH1 port. \& "Configuration of an IP datalink" on page 1746. For MS/TP refer to \({ }^{\xi}\) "Configuration of an MS/TP datalink" on page 1745.

\section*{Configuration of BACnet server root object}
1. Create an empty project with an AC500 V3 CPU type and call it fpr example "Device_12".
2. Insert a "BACnet Server" object below the interfaces object in the device tree.
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline Devices & \multicolumn{6}{|l|}{Add object below : Interfaces} & \(\times\) \\
\hline \multirow[t]{7}{*}{} & & & & & & & \\
\hline & \multicolumn{6}{|l|}{\multirow[t]{2}{*}{\begin{tabular}{l}
Object path: \\
Device_12IInterfaces
\end{tabular}}} & \\
\hline & & & & & & & \\
\hline & \multicolumn{6}{|l|}{Object name: BACnet_Server} & \\
\hline & Categories & \(\checkmark\) & Search object name & & & & 8 \\
\hline & \multicolumn{2}{|l|}{\multirow[t]{2}{*}{\[
\begin{aligned}
& \text { Scripting } \\
& \text { Uncategorized }
\end{aligned}
\]}} & Name & Short Description & Version & Order Number & \\
\hline & & & BACnet Server & & 1.5.2.1 & & \\
\hline
\end{tabular}
3. Set the device InstanceNumber in the "BACnet Parameters" of the "BACnet Server", e.g. to 12 and the InstanceName to Device_12 (according to Fig. 273 BACnet objects, properties, services and BIBBs).
\begin{tabular}{|c|c|c|c|c|}
\hline Devices - \(\quad\) ¢ & \multicolumn{4}{|l|}{¢ BACnet_Server \(\times\)} \\
\hline \multirow[t]{17}{*}{} & & & & \\
\hline & BACnet Parameters & Parameter & Type & Value \\
\hline & & - UsedBACnetStringType & Enumeration of DINT & BACNET_STRING_U... \\
\hline & BACnet IEC Objects & - SupportUnsolicitedCOV & BOOL & FALSE \\
\hline & Status & - HideToOutside & BOOL & FALSE \\
\hline & & - AutoStart & B00L & TRUE \\
\hline & Information & AutoGenerateBACstackConfig & BOOL & TRUE \\
\hline & & -- Device Properties & & \\
\hline & & + propertyAttributes & & \\
\hline & & - InstanceNumber & DWORD & 12 \\
\hline & & - InstanceName & WSTRING(255) & Device_12" \\
\hline & & - SystemStatus & Enumeration of DINT & DEV_OPERATIONAL \\
\hline & & - VendorName & WSTRING(255) & "ABB" \\
\hline & & - VendorIdentifier & UDINT & 808 \\
\hline & & - ModelName & WSTRING(255) & 'PM5650-2ETH" \\
\hline & & - FirmwareRevision & WSTRING(255) & \({ }^{3}{ }^{*}\) \\
\hline & & - ApplicationSoftwareversion & WSTRING(255) & \({ }^{1} 1.14 .1 .1 .15 .2 .1{ }^{\text {c }}\) \\
\hline
\end{tabular}
4. Add a datalink, IP or MS/TP. In the example an IP datalink is inserted below ETH1. Default parameters are sufficient if only one datalink is used.
« "Configuration of an IP datalink" on page 1746
Devices
5. Build the project, download to the PLC and set it to [RUN]. The status of the "BACnet Server" should be green (running). If not, please ensure that you have installed the runtime license BACnet Protocol B-BC Runtime, verifiable by right-click on the PLC node and select [Show license information] from the runtime licensing menu. The project is scanned for required licenses. If you are logged in to a PLC, then the licenses available on the PLC are displayed. A missing required license is highlighted.
«y Chapter 6.3.2.2.2 "PLC runtime licensing" on page 1446

6. Start any BACnet client to find the server, for example Inneasoft BACnet Explorer.


\section*{Adding BACnet server objects}

Goal is to publish an analog value as BACnet server object. This example is according to Fig. 273 BACnet objects, properties, services and BIBBs, left part containing a temperature value.
1. Configure a "BACnet Server" root object according to \({ }^{〔}\) Chapter 6.3.4.9.3.5.2 "Configuration of BACnet server root object " on page 1739.
2. Add a "BACnet Analog Input" object below the "BACnet Server".

3. Rename it to Temperature, adjust the parameters: InstanceNumber: 1010, Description: Temperature, Units: UNIT_DEGREES_C.
\begin{tabular}{|c|c|c|c|c|}
\hline Devices \(\quad-9 \mathrm{x}\) & 1) sacrectever & _12 Temperature & & \\
\hline \multirow[t]{15}{*}{} & Bachet Parameters & Parameter & Tpe & Value \\
\hline & & - - xocertrambutes & & \\
\hline & EACnet IIC Objets & - Irotancolvite & DWCRS & 2030 \\
\hline & & - Inetanceliane & WSTRNGC(255) & Temerane* \\
\hline & & - fresentialve & REM & 0.0 \\
\hline & Information & - Descrition & WSTRDM(25s) & Tenceratre* \\
\hline & & - Devicetipe & WSTRDVG(255) & \\
\hline & & - Scautlugs & String & 0009 \\
\hline & & - Evenssate & frumesson of opr & state iokem \\
\hline & & - Relustay & brumesson of obit &  \\
\hline & & - Outofervice & \(80 ¢\) & FMSE \\
\hline & & - Usdotelnterval & L000 & - \\
\hline & & - unts & Enumersm of Dist & unitpecrees_c \\
\hline & & - Mroresvar & RES & 0.0 \\
\hline & & - Mapresvave & Rek & 1500. \\
\hline
\end{tabular}
4. The present value of the objects Temperature needs to be fed with the value from the real temperature device. Alternatively, a simple PLC program can simulate this value.

5. Download the program and observe the temperature value in the BACnet client.
\begin{tabular}{|c|c|c|c|}
\hline \multicolumn{4}{|l|}{- Inneasoft BACnet Explorer (Free Edition)} \\
\hline \multicolumn{4}{|l|}{\multirow[t]{2}{*}{File View ?}} \\
\hline & & & \\
\hline \multirow[t]{11}{*}{\begin{tabular}{l}
Favorites \\
Local network \\
Device_12 \\
(1) Alarms \\
困 COV subscriptions \\
(1) Analog Input \\
Temperature \\
Device
\end{tabular}} & Id & Name & Value \\
\hline & 85 & Present Value & 25,36049 \\
\hline & 168 & Profile Name & \\
\hline & 103 & Reliability & no-fault-detected (0) \\
\hline & 357 & Reliability Evaluation Inhibit & False \\
\hline & 106 & Resolution & \[
0,1
\] \\
\hline & 111 & Status Flags & \(\{0 ; 0 ; 0 ; 0\}\) \\
\hline & 113 & Time Delay & 0 \\
\hline & 356 & Time Delay Normal & 0 \\
\hline & 117 & Units & \({ }^{\circ} \mathrm{C}(62)\) \\
\hline & 118 & Update Interval & 0 \\
\hline \multicolumn{4}{|l|}{Number of properties : 33} \\
\hline
\end{tabular}

\section*{Adding BACnet client functionality}

Goal is to configure a second AC500 controller as BACnet client which reads an analog value from a server. This example is according to Fig. 273 BACnet objects, properties, services and BIBBs, right part.
1. Add a new controller and configure a "BACnet Server" root object according to \({ }_{\xi}\) Chapter 6.3.4.9.3.5.2 "Configuration of BACnet server root object" on page 1739.
2. Set InstanceNumber to 14 and InstanceName to Device 14.

3. In addition to BACnet objects, BACnet clients can also be inserted as devices under a "BACnet Server". Add a "BACnet Client Read Property" below the "BACnet Server" node.

4. The created object "BACnet Client Read Property" generates a function block instance which can be used to program the client read functionality. The figure below shows a simple example.

In line 1-5 of the code part the function block is called with the following parameter:
- Device ID of the server to read from (12) \({ }^{y} \boldsymbol{y}\) Chapter 6.3.4.9.3.3 "Supported objects and properties " on page 1737
- Object ID of the object to read from (1010 for the "Analog Input")
- Object type ("Analog Input")
- Property to read ("present value")
- triggerRead to start the read operation

When the user (or another program part) sets the variable triggerRead from FALSE to TRUE the edge triggered function block BACnet_Client_Read_Property starts operation and sends the read request to the server device. After receiving the reply from the Server, the output . xDone gets TRUE (line 8) and the temperature value can be read from the output. result (line 14).

5. Download this program to another AC500 V3 controller, which is in the same IP network as the server. Set it to run and read the temperature value by setting triggerRead to TRUE. In online mode the read temperature value can be observed in line 14.


Alternative con- Unlike BACnet objects, a BACnet client does not require a complex (static) configuration, thus a figuration client function block can be used without creating a BACnet client as device.


There is no BACnet_Client_Read_Property object created below the "BACnet Server". Instead a function blōck BACnēt_Client_Read_Property must be declared in the PRG (line 6 in the declaration) and initially "connected" to its "BACnet Server" in IEC-code via RegisterToServer(), and thus get activated (line 2 in the code) \(\Longleftrightarrow\) Chapter 6.5.14 "Reference, function blocks" on page 4086.


\section*{Configuration of datalinks}

For communication with other BACnet devices AC500 provides two different possibilities:
MS/TP and IP.
* \(\boldsymbol{y}^{2}\) Chapter 6.3.4.9.3.2 "Supported BACnet networks " on page 1736

For a non-routing device one MS/TP or IP datalink must be configured.
If more than one datalink is configured, routing between the datalinks is automatically enabled.

Configuration of • Add the "BACnet MS/TP COM" object below the COM port. an MS/TP datalink


In fact the empty COM port is replaced by the "BACnet MS/TP COM". By that the COM port is configured as RS-485 with fixed settings for MS/TP: No parity, 8 data bits, 1 stop bits.
- Below the "BACnet MS/TP COM" port object an "BACnet MS/TP datalink" is inserted automatically which can be configured according to the requirements.
\begin{tabular}{|c|c|c|c|c|}
\hline Devices \(\quad\) - \(\quad \mathbf{x}\) & \multicolumn{4}{|l|}{BACnet_MS_TP_datalink \(\times\)} \\
\hline \multirow[t]{10}{*}{} & & \multicolumn{3}{|l|}{\multirow[b]{3}{*}{\begin{tabular}{|l|l|l|}
\hline Parameter & Type & Value \\
\hline\(\quad\) DeviceName & STRING & '/dev/ttyO0' \\
\hline & \\
\hline
\end{tabular}}} \\
\hline & BACnet MS/TP Parameters & & & \\
\hline & \multirow[b]{2}{*}{BACnet MS/TP IEC Objects} & & & \\
\hline & & - NetworkNumber & WORD & 1 \\
\hline & \multirow[t]{2}{*}{Status} & - ConnectionType & Enumeration of USINT & Master \\
\hline & & - Baudrate & Enumeration of UDINT & 19200 \\
\hline & \multirow[t]{2}{*}{Information} & - ReplyTimeout & UDINT & 250 \\
\hline & & - TokenTimeout & UDINT & 20 \\
\hline & & - DatalayerAdress & UDINT & 1 \\
\hline & & - MessageQueuelength & WORD & 50 \\
\hline
\end{tabular}
- NetworkNumber: Use the default value 1 if no routing is required. For routing, use a unique network number in one controller.
- ConnectionType: Use the default value Master if no routing is required. For routing, use "Master - answering always postponed".
- Baudrate can be set according to requirements in the range of from 9600 to \(38400 \mathrm{bits} / \mathrm{s}\), higher values ( 57600 and 115200 bits/s) are not recommended.
- DatalayerAddress: This is the MAC address as described in \(\Leftrightarrow\) Chapter 6.3.4.9.3.2 "Supported BACnet networks" on page 1736. The MAC address must be unique in the MS/TP network.
- For all other parameters the default values are recommended for typical applications.

Configuration of - Add a "BACnet_IP_datalink" object below the Ethernet port ETH1 or ETH2. an IP datalink

- NetworkNumber: Use the default value if no routing is required. For routing, use a unique network number in one controller.
- UPDport: Use the default value (47808 decimal) in the normal case. Range is possible from BAC0 (= 47808 decimal) to BACF. UDPport + IP address form the MAC address of the IP datalink as described in \({ }^{4}\), Chapter 6.3.4.9.3.2 "Supported BACnet networks" on page 1736. The IP address cannot be specified here. It is automatically taken from the parent Ethernet node (ETH1 or ETH2); its IP address is set in the communication settings of the CPU node, "Device_14" in the example.
- ForeignDevice and BBMD: Special configuration is only needed if an internet router is located between two BACnet devices.
② Chapter 6.3.4.9.3.2 "Supported BACnet networks" on page 1736
AC500 can be configured as ForeignDevice or BBMD, but not the combination of both. An example for BBDM can be found in the example folder.

Configuration of Routing enables the combination of different BACnet networks to one common "BACnet Routing internetwork".
② Chapter 6.3.4.9.3.2 "Supported BACnet networks" on page 1736
BACnet devices from different BACnet networks can communicate with each other.
If more than one datalink is configured in one CPU, routing between the different networks is automatically enabled. It must only be ensured that the network number is unique in one controller.
\({ }^{*}\) ² Chapter 6.3.4.9.3.2 "Supported BACnet networks " on page 1736
For MS/TP the ConnectionType must be set to "Master - answering always postponed". An example for routing can be found in the example folder.

\section*{Time syncronisation}

The BACnet clients expect to receive the local time. Currently the AC500 V3 does not distinguish between UTC time and local time and its time zone is set to 0 . This will be improved in the near future. In the meantime, it is recommended to store the local time (green color in the following figure) in the AC500 as a workaround.


> Automation Builder (local time)
Time client
Meinberg clock (UTC time)

Using this workaround, the following time sync mechanisms can be used:
- Set local time from Automation Builder Tab "PLC Shell":

Set the time by the command "time hh:mm:ss"

- Read the local time from the Automation Builder Tab "Statistics":
"Current PLC Date and time" shows the PLC time as local time without conversion, if the tab "Show PLC time in UTC" is enabled.


For storing the local time in AC500, do not use the button [Set PLC to PC Time] (Tab "Statistics"), since this is always converting from local time to UTC time.
- BACnet clients can read local or UTC time, both requests will deliver the same (local) time information, since the timezone is 0 .
- If an SNTP time sync is required (for example with a Meinberg clock), UTC times are exchanged. For conversion of UTC to local time in AC500 a proprietary STNP client must be programmed.
Please contact the PLC support for more information.

\section*{Package content}

\section*{General}

The BACnet package PS5607-BACnet-BC can be installed with the Installation Manager and contains the following components:
- BACnet runtime component, part of AC500 firmware.
- Automation Builder package: CODESYS BACnet
- BACnet plug-in component
- Device descriptions for "BACnet Server", BACnet objects, BACnet client and datalinks
- Libraries: BACnet, BACnetDefaultImpl and CmpBACnet.
\# Chapter 6.3.4.9.3.6.2 "BACnet libraries" on page 1748

Example folder - Example folder
- Examples and example documentation
¿ Chapter 6.3.4.9.3.6.3 "Application examples" on page 1749
- Datasheet and FAQ

BACnet Protocol Implementation Conformance Statement (PICS), acting as a datasheet, describing all BACnet objects, services and communication capabilities. BACnet Conformance Certificate FAQ - Frequently Asked Questions, including AC500 specific information, performance and limit

\section*{BACnet libraries}

The IEC library CmpBACnet represents the integration of the BACnet stack into a CODESYS IEC environment and provides the BACnet data types as well as the BACstack methods.
The sole use of the IEC library CmpBACnet (without the BACnet and BACnetDefaultImpl libraries) would result in complex and lengthy IEC application code.

The BACnet library simplifies BACnet application development considerably as compared to the sole use of CmpBACnet, especially in the following areas:
- Starting and stopping the BACnet stack
- Using BACnet server objects and their properties
- Triggering asynchronous requests (mainly client service requests) and processing the request transaction
- Processing of callbacks from the BACnet stack (see IBACnetEventConsumer) and distributing the callbacks to multiple receivers in the application
Furthermore, the BACnet library provides a plug-in mechanism (BACnetServerPlugin) for extending certain aspects of the BACnet library. BACnetServerPlugin is the basis for the BACnetDefaultImpl library.

The BACnetDefaultImpl library is used for the additional simplification of BACnet application development. The BACnet standard ASHRAE 135 leaves some aspects of the practical use of BACnet open. The most notable examples include the following:
- Persistence of server objects
- Storage and persistence of Trend Log, Trend Log Multiple, and Event Log entries
- Update of the date/time information of the device object

The IEC library BACnet is intended as a layer over the IEC library CmpBACnet. However, the layer does not hide the library because this would require the BACnet library to have "facade" functions for CmpBACnet functions. These facade functions would result in larger application code and increased runtime requirements. This is difficult for the PLC to accept. For this reason, it is necessary to know when elements from the BACnet library or CmpBACnet library are to be used.

General rules:
- Starting and stopping the BACnet stack

Always use BACnetServer. StartBACnetStack and
BACnetServer.StopBACnetStack or AutoStart. Never directly use the corresponding
functions of the CmpBACnet library, such as CmpBACnet. BACnetServerInit.
- Using BACnet server objects and their properties

Always use the specified function blocks in IEC-lib-BACnet, such as BACnetAnalogValue. Never directly use the corresponding functions of the BACnet library, such as CmpBACnet.BACnetStorePropertyInstance.
- Triggering of asynchronous requests Always use the specified client function blocks of the BACnet library, such as BACnetClientReadProperty. Never directly use the corresponding functions of the CmpBACnet library, such as CmpBACnet. BACnetReadProperty. All functions of the CmpBACnet library that require a BACnetAsyncTransactionToken belong to this category and should never be used directly.
- Processing of callbacks from the BACnet stack and distributing the callbacks to multiple receivers in the application
Always use IBACnetEventConsumer and BACnetServer. RegisterHook/ UnregisterHook/RegisterCallback/UnregisterCallback. Never directly use the corresponding functions of the CmpBACnet library, such as CmpBACnet. BACnetSetHook or CmpBACnet. BACnetSetCallback.
When is it appropriate and safe to directly call the functions of the CmpBACnet library?
Basically, it is only necessary to call functions of CmpBACnet directly when a corresponding functionality is not provided in the BACnet library. Check the BACnet library first before trying to use CmpBACnet directly. It is possible to use blocking functions in CmpBACnet, such as BACnet*CbCompletion, BACnetIam(Ex), or BACnetIHave(Ex), BACnetUnconf*.

Most often, you will use BACnet*CbCompletion to implement your specific
IBACnetEventConsumer.BACnetEventCallbacks. But first check whether or not the BACnetDefaultImpl library already contains an appropriate standard implementation.

\section*{Application examples}
- AC500_V3_BACnet_B-BC_Example_ABxxx.project including simple read and write operations between curient and server.
- Use case 1: AC500 as BACnet client, read and write (with priority)
- Use case 2: AC500 as "BACnet Server", publish the analog value
- AC500_V3_BACnet_B-BC_Example_Routing_ABxxx.project
- Examples from 3S, including
- Read and write operations with more options, notification class, calendar, scheduler, etc.
- Device discovery
- BBMD
- Persistence
- Logging
- Routing

\subsection*{6.3.4.10 UDP protocol}

\subsection*{6.3.4.10.1 Contents of the UDP protocol configuration}

Functionality UDP and TCP are widely used in network communication. UDP is a connectionless transport protocol. It has a similar task as the connection oriented TCP. This means that the sender does not know whether the data packets it has sent have arrived. While TCP sends acknowledgements when receiving data, UDP does without. This has the advantage that the packet header is much smaller, and the transmission link does not have to transmit acknowledgements.
Typically, UDP is used for DNS requests, VPN connections, audio and video streaming.

UDP in V3 is no longer configured in the hardware tree but only implemented applicatively via a library.

There are two libraries, one that works on the lowest level (SysSocket2, library, to get access to the socket interface for TCP/IP communication on the target.) and a user library (Net Base Services, this library implements a TCP server, TCP client and a UDP peer as a collection of base services for Ethernet).

Chapter 6.5.3 "Library Manager functionality" on page 3773
* Chapter 5.2.1.2.1.3.3 "Overview of protocols, sockets and ports" on page 310

② Chapter 6.5.14 "Reference, function blocks" on page 4086
(4) "Data synchronization via UDP" on page 3882

② Chapter 6.3.4.9.3.2 "Supported BACnet networks " on page 1736

The application example Communication via UDP protocol demonstrates the communication between a AC500 V2 PLC and a AC500 V3 PLCs via UDP protocol.

\subsection*{6.3.4.11 OPC UA}

AC500 V3 controllers support the OPC UA protocol - a machine-to-machine communication protocol for industrial automation. For details on installation and configuration of an OPC UA server refer to the OPC UA server description \({ }^{*}\) Chapter 6.3.6.2 "OPC UA server for AC500 V3 products" on page 1787.

How to connect robot controllers to OPC UA is described in the \(\underline{A B B}\) News.

Application note Details on OPC UA communication with an AC500 V3 PLC are given in the application note AC500 V3 and OPC UA.

Application The application example How to use OPC server V3-for DA and UA is available to gain a example deeper understanding of the OPC protocol and to configure AC500 V3 accordingly.

\subsection*{6.3.5 Data transfer and programming}

\subsection*{6.3.5.1 Data exchange between AC500 V3 PLCs via network variables}

Network variables can be used to exchange data between two AC500 V3 PLCs. The variables must be defined in strict and identical lists on both PLCs which are sender and receiver. The lists can be in one or more projects.
A step-by-step description on how to use network variables can be found in the application example.

\subsection*{6.3.5.2 Data exchange between AC500 V3 PLCs via EtherCAT}

An EtherCAT bridge can be used to exchange data between two AC500 V3 PLCs in real time. The procedure is described in the application example using an EL6695 bridge terminal.

\subsection*{6.3.5.3 Source download/upload}

Source down- Prerequisites
load
- Communication settings are correct
- Project is saved on PC
- PLC is connected
1. Login to the PLC.
2. In the menu click "Online \(\boldsymbol{\rightarrow}\) Source download to connected device".
\(\Rightarrow\) Project archive will be downloaded to PLC.
3. To verify download double-click node "PLC_AC500_V3", select view "Files" and doubleclick folder "PlcLogic" of the Runtime view (if necessary click refresh button of Runtime view).
\(\Rightarrow\) File Archive.prj will appear if download was successful.
\begin{tabular}{|c|c|c|}
\hline Runtime Location: & \(\square\) PlcLog & - \(10 \times 6\) \\
\hline Name & Size & Modified \\
\hline t.. & & \\
\hline 1. trend & & \\
\hline 1. alarms & & \\
\hline [. _cnc & & \\
\hline visu & & \\
\hline - Application & & \\
\hline - ac_persistence & & \\
\hline ] Archive.prj & 322.70 ... & 1/1/197... \\
\hline
\end{tabular}

\section*{Source upload Prerequisite}
- Project archive on PLC available (from previous source download)
- PLC is connected

1. Open the Automation Builder.
2. In the menu click "File \(\rightarrow\) Source upload...".
\(\Rightarrow\) A window "Pick the device from where you want to upload the source archive..." appears.


If you get an error click [Scan for other devices].
3. Select your PLC with the archive and click [OK].
\(\Rightarrow\) Dialog "Extract Project Archive" appears.

4. Select your preferred folder and click [Extract].
\(\Rightarrow\) Then you are prompted to open the project archive.

\section*{Automation Builder - Premium}

Do you want to open the project which has been extracted from the project archive?
5. Click [Yes].
\(\Rightarrow\) The project opens.
Upload was successful.

\subsection*{6.3.5.4 Programming and testing}

For information on programming see
- \({ }^{4}\) Chapter 6.4.1.9 "Programming of Applications" on page 1844
- \(\boldsymbol{y}^{2}\) Chapter 6.4.1.11.2 "Configuring the Connection to the PLC" on page 1965
- \(y^{4}\) Chapter 6.3.5.5.2 "Enter a known PLC IP address" on page 1754
- \(\Leftrightarrow\) Chapter 6.4.1.11 "Downloading an Application to the PLC" on page 1965

For Information on testing/debugging see \({ }^{\text {² }}\) Chapter 6.4.1.12 "Testing and Debugging" on page 1980

\subsection*{6.3.5.5 Configuration of communication via Ethernet (TCP/IP)}

\subsection*{6.3.5.5.1 General}

Programming via Ethernet is only possible on a PC with Ethernet board and installed network. Programming can be done via the internal (onboard) Ethernet communication module.

An application note describes the configuration of an AC500 V3 PLC for EtherNet/IP communication.


\subsection*{6.3.5.5.2 Enter a known PLC IP address}
1. Right-click the top node "PLC_AC500 <...>" and select "Communication Settings" from the context menu.
\(\Rightarrow\) Dialog box "Communication Settings <...>" appears.

2. Enter your PLC IP Address and click [OK].

\subsection*{6.3.5.5.3 Enter PLC IP address by scanning devices}
1. Right-click the top node "PLC_AC500 <...>" and select "Communication Settings" from the context menu.
\(\Rightarrow\) Dialog box "Communication Settings <...>" appears.

2. Click [...].
\(\Rightarrow\) Dialog box "Communication Settings <...>" appears.
3. Click [Scan], select your desired PLC and click [OK].
\(\Rightarrow\) Entry is transferred to the dialog box "Communication Settings <...>".
Click [OK].

4. Click to log in the "PLC_AC500_V3" project.

\subsection*{6.3.5.5.4 Enter PLC IP address by [Advanced Settings...]}

If a remote gateway instead of a local one has to be used it can be configured in the [Advanced Settings...].
1. Right-click the top node "PLC_AC500 <...>" and select "Communication Settings" from the context menu.
\(\Rightarrow\) Dialog box "Communication Settings <...>" appears.

2. Enable checkbox "Use advanced settings" and click [Advanced Settings...].
\(\Rightarrow\) Tab "Communication Settings" opens.
3. Check gateway or change if required.
\(\Rightarrow\) Successful connection is indicated by green dot on the gateway icon.
4.


Check IP address or change if required.
5. Press ENTER to confirm changed IP address.
\(\Rightarrow\) Successful communication is indicated by green dot on the PLC icon.
6. Or instead of the last two steps:

\section*{Set the IP address via a scan.}

Click [Scan Network], select your desired PLC and click [OK].

\(\Rightarrow\) Successful connection is indicated by green dot on the gateway icon.
7. Click to log in the "PLC_AC500_V3" project.

\subsection*{6.3.5.6 PLC shell commands}

The PLC shell is used for requesting specific information from the controller. By entering a device-specific command the response is returned in a result window. The PLC shell can be issued without login.
Proceed as fol- 1. Ensure the gateway is configured properly and a connection to the controller can be lows:
2. In Automation Builder double-click the PLC node and open the tab "PLC Shell".
3. Enter "?" in the command line of the tab window. All available PLC commands are listed.

If the gateway is able to establish a connection to the controller, an online connection to the PLC is opened automatically.

The commands listed in online mode can differ from the commands shown when pressing the button [...] as Automation Builder version and firmware version can differ.
* \({ }^{*}\) Chapter 6.4.1.21.2.8.11 "Tab 'PLC Shell"' on page 2441

²) Chapter 6.2.6 "Version information" on page 1384
ⓨ Chapter 6.3.1.4 "Firmware identification and update" on page 1419.

\subsection*{6.3.5.7 Watchlists}
« Chapter 6.4.1.13.2.3 "Using watch lists" on page 2002
Chapter 6.4.1.13.2.4 "Changing Values with Recipes" on page 2003

\subsection*{6.3.5.8 Reference to libraries}

Library configuration is described in the chapter .

\subsection*{6.3.5.9 Reference to application libraries}

Application libraries can be used in AC500 V3 PLCs. The requirements for the use of the function blocks of the application libraries and information and prerequisites for the general handling of application libraries are described in the application examples:

HTTP library In order to be able to use the PLC as a client for web services, the HTTP function block library can be used. Setup and use of the HTTP library are described in the application example AC500 Webservices - HTTP Function Block Library.

MySQL library With the help of the MySQL function block library, MySQL databases can be used to store and access AC500 V3 data. Setup and use are described in an application example.

MSSQL library With the help of the MSSQL function block library, MSSQL databases can be used to store and access AC500 V3 data. Setup and use are described in an application example.

\subsection*{6.3.6 Server installation}

\subsection*{6.3.6.1 OPC server for AC500 V3 products}

\subsection*{6.3.6.1.1 Introduction}

Architecture of the CODESYS OPC server


\section*{Essential documents}
«4 Further information
\begin{tabular}{|l|l|l|l|}
\hline & File name & \begin{tabular}{l} 
Com- \\
ment
\end{tabular} & Where to find \\
\hline REF1 & \begin{tabular}{l} 
OPC_V3_how_to_use_E.pd \\
\(f\) \\
OPC_V3_how_to_use_D.pd \\
\(f\)
\end{tabular} & OPC V3 & \begin{tabular}{l} 
C:\Program Files\ABB\CoDeSys OPC Server \\
3 AE
\end{tabular} \\
\hline REF2 & \begin{tabular}{l} 
AeConfigurator_User- \\
Guide.pdf
\end{tabular} & OPC V3 & \begin{tabular}{l} 
C:\Program Files (x86)\3S \\
CODESYSICODESYS OPC Server 3
\end{tabular} \\
\hline REF3 & ReadMe.rtf & OPC V3 & \begin{tabular}{l} 
Installation ABB DM Suit 1.0.: \\
IPLC - AC500\OPC ServerlOPC- \\
ServerV3.xAE\
\end{tabular} \\
\hline REF4 & \begin{tabular}{l} 
ReleaseNotesOPCV3 AE for \\
HA
\end{tabular} & OPC V3 & \begin{tabular}{l} 
Installation ABB DM Suit 1.0.: \\
IPLC - AC500\OPC Server\OPC- \\
ServerV3.xAE\
\end{tabular} \\
\hline
\end{tabular}

\section*{Work flow}

\section*{Consideration and preparation}

*) \({ }^{\text {² }}\) Chapter 6.3.6.1.2.2 "Installation of OPC server" on page 1766

\section*{Commission OPC server}

*) 党 Chapter 6.3.6.1.2.3.1 "Define symbols" on page 1769

\({ }^{1}\) ) \({ }^{2}\) Chapter 6.3.6.1.2.5 "Configure AlarmEvents" on page 1782
\({ }^{2}\) ) * Chapter 6.3.6.1.2.5.1 "Check AlarmEvents" on page 1782

\section*{Adjustment to target OPC client}

\({ }^{1}\) ) 乡chapter 6.3.6.1.1.2 "Essential documents" on page 1759 REF4.
\({ }^{2}\) ) \& Chapter 6.3.6.1.2.6 "Configure user account for OPC server" on page 1783

\subsection*{6.3.6.1.2 Hints}

Default folder and contents
Windows 7, Windows Server 2008/2016 (64-bit)
\begin{tabular}{|l|l|}
\hline OPC Server V3 & \begin{tabular}{l} 
Windows 7 64-bit, Windows Server 2008 \\
64-bit, Windows Server 2016 64-bit
\end{tabular} \\
\hline \begin{tabular}{l} 
WinCoDeSysOPC.exe \\
OPCConfig.exe \\
AEConfiguration.exe \\
CoDeSys_OPC_Server_V3_User_Guide.pdf \\
CoDeSys_OPC_Server_V3_Benutzerhand- \\
buch.pdf \\
AeConfigurator_UserGuide.pdf
\end{tabular} & \begin{tabular}{l} 
C:IProgram Files (x86)l3S CoDeSys\CoDeSys \\
OPC Server 31
\end{tabular} \\
\hline \begin{tabular}{l} 
OPCServer.ini \\
OPCServerA.ini \\
OPCServer.log
\end{tabular} & C:IProgramDatalCoDeSysOPC\} \(\\
{\hline \text { Symbol file *.SDB, *.SYM }} &{\)\begin{tabular}{l}
\text { CBP open, after project build or rebuild all: in } \\
\text { the project folder }
\end{tabular}\(} \\
{\hline \text { Symbol file *.SDB }} &{\)\begin{tabular}{l}
\text { After login in AC500: } \\
\text { C:IProgramDatalGateway Files } \\
\text { After starting the OPC server: } \\
\text { C:IProgramDatalGateway FileslUpload }
\end{tabular}\(} \\
{\hline \text { Gateway.exe }} &{\text { C:IWindowsISysWOW64\ }} \\
{\hline}\)
\end{tabular}

Windows 7 (32-bit), Windows Server 2008/2016 (32-bit)
\begin{tabular}{|l|l|}
\hline OPC Server V3 & \begin{tabular}{l} 
Windows 7 32-bit, Windows Server 2008 \\
32-bit, Windows Server 2016 32-bit
\end{tabular} \\
\hline \begin{tabular}{l} 
WinCoDeSysOPC.exe \\
OPCConfig.exe \\
AEConfiguration.exe \\
CoDeSys_OPC_Server_V3_User_Guide.pdf \\
CoDeSys_OPC_Server_V3_Benutzerhand- \\
buch.pdf \\
AeConfigurator_UserGuide.pdf
\end{tabular} & \begin{tabular}{l} 
C:IProgram FilesI3S CoDeSys\CoDeSys OPC \\
Server 31
\end{tabular} \\
\hline \begin{tabular}{l} 
OPCServer.ini \\
OPCServerA.ini \\
OPCServer.Iog
\end{tabular} & C:IProgramDatalCoDeSysOPC\ \\
\hline Symbol file *.SDB, *.SYM & \begin{tabular}{l} 
CBP open, after project build or rebuild all: in \\
the project folder
\end{tabular} \\
\hline Symbol file *.SDB & \begin{tabular}{l} 
After login in AC500: \\
C:IProgramDatalGateway Files \\
After starting the OPC server: \\
C:IProgramDatalGateway FilesIUpload
\end{tabular} \\
\hline Gateway.exe & C:IWindowsISystem32 \\
\hline
\end{tabular}

Windows Server 2008/2016 (32-bit)
\begin{tabular}{|l|l|}
\hline OPC Server V3 & \begin{tabular}{l} 
Windows Server 2008 32-bit, Windows \\
Server 2016 32-bit
\end{tabular} \\
\hline \begin{tabular}{l} 
WinCoDeSysOPC.exe \\
OPCConfig.exe \\
AEConfiguration.exe \\
CoDeSys_OPC_Server_V3_User_Guide.pdf \\
CoDeSys_OPC_Server_V3_Benutzerhand- \\
buch.pdf \\
AeConfigurator_UserGuide.pdf \\
OPCServer.ini \\
OPCServerA.ini \\
OPCServer.log
\end{tabular} & \begin{tabular}{l} 
C:IProgram FilesI3S CoDeSysICoDeSys OPC \\
Server 31
\end{tabular} \\
\hline Symbol file *.SDB, *.SYM & \begin{tabular}{l} 
CBP open, after project build or rebuild all: in \\
the project folder
\end{tabular} \\
\hline Symbol file *.SDB & \begin{tabular}{l} 
After login in AC500: \\
C:IWINDOWSIGateway Files \\
After start CODESYS OPC server: \\
C:IWINDOWSIGateway FilesIUpload
\end{tabular} \\
\hline Gateway.exe & C:IWindowsISystem32 \\
\hline
\end{tabular}

If folder C:IProgramDatal is missing, select "Show hidden files, folders and drives" at "Control Panel \(\rightarrow\) All Control Panel Items \(\rightarrow\) Folder Options \(\rightarrow\) View \(\rightarrow\) Hidden files and folders".

\section*{Installation of OPC server}

\section*{Prerequisites}

\section*{The following applications are closed:}
- All OPC clients
- ABB OPC tunnel
- CODESYS gateway server

\section*{Ensure termination of the following processes:}
- Gateway.exe
- CoDeSysOPC.exe
- WinCoDeSysOPC.exe
- OCTsvc.exe

\section*{Installing with Automation Builder}
1. Go to homepage http://new.abb.com/plc/automationbuilder/platform/software .
2. Click [Latest Automation Builder version (recommended)] and run the installer.

3. Open "Installer Options and Additional Tools" and click [Install Additional Tools].
4. Agree to the "License Terms".

5. Select "Version 2 and/or 3" and install.
\(\Rightarrow\) All required files are installed for OPC and the OPC server is registered automatically as user application.

\section*{Manual registration and unregistration}

It is possible to register or to uninstall the OPC server manually either as COM server (user application) or as a service.

Register the OPC server as interactive software in the Windows registry:
Command for OPC 3: WinCoDeSysOPC/RegServer
Register the OPC server as system service:

\section*{Command for OPC 3: WinCoDeSysOPC/Service}

Unregister the OPC server from the Windows registry and from the service entry:

Command for OPC 3: WinCoDeSysOPC/UnRegServer
Please see REF1 chapter 3 (OPC 3) « Table on page 1759 for details.

\section*{Example: Register OPC server V3 as a system service}
\(\checkmark\) Prerequisites: All programs, processes and services which connect to the OPC server are closed.
1. Start the "Command Prompt" with command "cmd" in the "Start \(\rightarrow\) Run... " window.

2. Go to the CoDeSysOPC V2 installation folder.
3. Unregister the OPC server with WinCoDeSysOPC/UnRegServer.
4. Register the OPC server as system service with WinCoDeSysOPC/Service.

\section*{OPC clients for tests}

Free of charge test clients can be found in the web:
https://industrial.softing.com
http://www.matrikonopc.com/products/opc-desktop-tools/index.aspx

\section*{Symbol file}

\section*{Define symbols}
1. Right-click on "Application" in CODESYS V3 project and click "Add Object".

2. Choose "Symbol Configuration" and click [Add object].

3. Select your programs and/or single symbols and click [Build].
\(\Rightarrow\) A symbol file will be automatically downloaded to the PLC with Project Download.

With double-click in the device tree to "Symbol Configuration" you can change the "Symbol Configuration" settings.

To restrict traffic and load, choose only symbols you need.

\section*{Configure OPC server}

Configure OPC Server V3
1. Start CODESYS/ CoDeSysOPC Server V3/OPC Configurator.

2. If the configuration is needed furthermore, save the configuration.

The actual configuration at start of OPC server will always be read from OPCServer.ini.

\(\Rightarrow\)

\section*{Update rate}
- The "Update Rate" may not be 0 (ms)!
- The default value of 200 ms is a suitable value of many applications.
- The adjustment for the "Update Rate" depends on the number of symbols (variables).
- For a big number of symbols it can be better to increase the "Update Rate".

The checkboxes "Sync Init" and "Enable logging (Defaultevents)" must be enabled.

3．Select＂PLC＿FW3＂．

OPCConfig－C：\ProgramData\CoDeSysOPC\OPCServer．ini
File Edit ？
曰四 Server

Settings for PLC＿GW／3
Interface：
Project name：

\begin{tabular}{ll|}
\hline Active： & \(\boxed{\downarrow}\) \\
Matarala Bytearder： & \(\Gamma\) \\
Na Lagin－Service： & \(\Gamma\) \\
\hline
\end{tabular}
－Logging
\(\sqrt{\sim}\) Enable logging（Defaultevents）
「 Log Additional Events
「 Add Debug Events（slow）
\(\Rightarrow\)
－If the＊．sdb files should be loaded from the＂Gateway Files＂direc－ tory on PC，the project name must be identical with project name in CODESYS．The extension is not necessary．
－If the symbol information should be loaded from AC500 V2．x，the project name is not required and can also be empty．
－The parameters displayed in the screenshot above are recom－ mended default settings．
－The checkbox＂Active＂must be enabled．
－Enabled checkbox＂Enable logging＂allows a later diagnosis．
4. Select "Connection" and click [Edit].

5. Enter the TCP/IP address of the target PLC at "PLC name or address" and enable "Use Tcp/lp blockdriver".
6. Enter the TCP/IP address of the target PLC at "IP Address of PLC" and click [OK].
7. Click "File \(\rightarrow\) Save" OPCserver.ini and "File \(\rightarrow\) Exit" OPCConfig.

\section*{Check OPC function with AC500}

\section*{Prerequisites}


It is urgently recommended to check the function of the configuration steps.
Chapter 6.3.6.1.2.4.1 "Configure OPC Server V3" on page 1770

\section*{Check OPC server V3}
1. Start OPCExplorer.exe and connect "CoDeSys.OPC.DA".

2. Add Group, add Items, select availabe Items in Server "CoDeSys.OPC.DA".

Add to Tag List, close the Item browser.
\(\Rightarrow\) If anything is right, then "CoDeSys.OPC.DA" is connected, is running and the "Quality" of the items is good.

Check processes with windows task manager


Correct configuration: All processes run with the same "User Name" and with the same "Session ID".

Test OPC function without AC500

\section*{AC500 project}
1. Open CoDeSys Application.

2. Collect all OPC variables in a separate "Global Variables" list.
3. Under "Project \(\rightarrow\) Options" select the "Symbol configuration".

Enable checkbox "Dump symbol entries" and click [Configure symbol file].

4. Disable all the checkboxes and confirm twice with [OK].

5．Under＂Project \(\rightarrow\) Options＂＂Symbol configuration＂click［Configure symbol file］again．
\begin{tabular}{|c|c|}
\hline Set object attributes & \(\times\) \\
\hline \begin{tabular}{l}
litrary Sysint＿AC500＿V10．LIB 14．4．10 10：51．34：FOUSs
Gilobal Variables
Global＿Variables \\
Var［ WORD］ \\
Var2（WORD］ \\
Var3 WDRD］ \\
Variable＿Configuration［VAR＿CONFIG］ \\
田 library lecstrab lib 13．06 16：51：28：global variables \\
円 library SysLitl｜em．it 14．4．10 10：51：34：global variables \\
闸… \(\square\) library SysLibTime lib 14．4．10 10：51：34：global variables \\
円… \(\square\) library SysTaskInfo．lit 14．4．10 10：51：34：global wariables \\
円… \(\square\) library IUtil lib 18．5．10 16：14：28：glabal variables
\end{tabular} & Cancel \\
\hline \(\square\) Export wariables of object & \\
\hline －Export data entries & \\
\hline \(\Gamma\) Export structure compoherts & \\
\hline \(\Gamma\) Expart amay entries & \\
\hline \(\Gamma\) Write access & \\
\hline
\end{tabular}

6．Select the variables which should be communicated as symbol．
Enable the following checkboxes：
－＂Export variables of object＂
－＂Export structure components＂
－＂Export array entries＂
－＂Write access＂
7．Confirm twice with［OK］．
8．Under＂Project \(\rightarrow\) Rebuild all＂rebuild the project．

9. In the project folder is the subfolder "OPC_test1__AC500_PM573_ETH__OPC_test1". It contains symbol files *.SYM and *.SDB with the time of the "Rebuild all". The items in the file *.SYM can be checked with Notepad. The binary file *.SDB contains the items for the OPC server. With <Online> <Login> it will copied in the gateway files directory and optionally on the AC500.

10. The folder "OPC_test1__AC500_PM573_ETH__OPC_test1" is a temporary folder, if the CBP project is opened. For the simulation of the server OPC it is copied *.SDB by hand.

\section*{Configure OPC server V3}

1. Select "Edit", append "PLC" and keep the default values.

2. You must specify "Project name" with the "directory name".

Connection settings are not required for the simulation.
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline \multicolumn{6}{|l|}{W)MatrikonopC Explorer - [Untitied \({ }^{\text {P }}\) ]} & \multirow[t]{2}{*}{- \(\mid\) [] \(|x|\)} \\
\hline \multicolumn{6}{|l|}{Fie Server Group Item View Help} & \\
\hline \multicolumn{7}{|l|}{OPC_test1_V3_smulation Contents of 'OPC_test1_V3_simuation'} \\
\hline \multirow[t]{2}{*}{} & Item \(\mathbb{I D}\) & Acces...| Value & Qualty & Tmestamp & Status & \\
\hline & \multicolumn{6}{|l|}{Wiliters} \\
\hline Server Info & \multicolumn{3}{|l|}{\multirow[t]{2}{*}{\begin{tabular}{l}
MatrikonOPC \\
+Ctobel 24/7 Support \\
-Vondor Nectrat Training \\
-Superior Client Care \\
Fies Baemieads \\
NatrikonOPC
\end{tabular}}} & \multicolumn{3}{|c|}{Group Info} \\
\hline \begin{tabular}{l}
Server: CoDeSys.CPC.DA \\
Connected: Yes \\
State: Rumning \\
Groups: 1 \\
Total Items: 0 \\
Current Local Time: 06.29.2013 6: 14:59.654 PM \\
Update Local Time: n/d
\end{tabular} & & & & \begin{tabular}{l}
Group: OPC_test1_1/3_s \\
Connected (Async I/O) \\
Active: Yes \\
Items: 0 \\
Current Update Rate: \\
Percent Deadband: 0,00 \\
Data Change Rate: 0,0
\end{tabular} & & \\
\hline
\end{tabular}
1. OPC Server V3: "Connect CoDeSys.OPC.DA". Add "Group", add "Items", select "Availabe Tags" and add to "Tag List".

2. The OPC Server V3 ("CoDeSys.OPC.DA") is connected, running and the "Quality" is good.

One OPC client can read / write the values of the items.

3. Similar configuration as above.

The OPC Server V2 ("CoDeSys.OPC.02") is connected, running and the configured items are found. But the "Quality" is bad. One OPC client can not read / write the values of the items.

Check processes with windows task manager
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline \multicolumn{5}{|l|}{Windows Task Manager} & \multicolumn{2}{|r|}{\multirow[t]{2}{*}{- 믖}} \\
\hline \multicolumn{5}{|l|}{File Options View Help} & & \\
\hline Applications Processes Services & Performance | & Networking | Users & & & & \\
\hline Image Name & User Name & Session ID - & Memory (... & Description & & \(\triangle\) \\
\hline ControlBuilderPlus.exe *32 & DEU120565 & 1 & 72.240 K & Control Builder Plus & & \\
\hline pitmsvr.exe *32 & DEU120565 & 1 & 8.784 K & pitmsvr module & & , \\
\hline TOTALCMD64.EXE & DEU120565 & 1 & 9.800 K & Total Commander & & \\
\hline prmedsvr.exe *32 & DEU120565 & 1 & 4.672 K & PrmEdSvr module & & \\
\hline WINWORD.EXE *32 & DEU120565 & 1 & 84.700 K & Microsoft Word & & \\
\hline TOTALCMD64.EXE & DEU120565 & 1 & 8.004 K & Total Commander & & \\
\hline CODESY ~1.EXE *32 & DEU120565 & 1 & 2.364 K & CoDeSysOPC.EXE & & \\
\hline prmt.exe *32 & DEU120565 & 1 & 13.532 K & PROMT & & \\
\hline MobileOPCExplorer.exe *32 & DEU120565 & 1 & 9.760 K & MatrikonOPC Explorer & & \\
\hline PrmtSvr.exe *32 & DEU120565 & 1 & 38.924 K & PrmtSVR Module & & \\
\hline Gateway.exe *32 & DEU120565 & 1 & 6.020 K & GATEWAY.EXE & & \\
\hline i_view32,exe *32 & DEU120565 & 1 ¢ & 5.028 K & IrfanView & & \\
\hline LangDict.EXE *32 & DEU120565 & 1 & 3.716 K & LangDict & & \\
\hline I_view32.exe *32 & DEU120565 & 1 & 4.980 K & IrfanView & & \\
\hline i_ view32.exe *32 & DEU120565 & 1 & 14.988 K & Irfanview & & \\
\hline WinCoDeSysOPC.exe * 32 & DEU120565 & 1 & 2.772 K & WinCoDeSysOPC.EXE & & \(\nabla\) \\
\hline \multicolumn{5}{|l|}{- Show processes from all users} & End Process & \\
\hline Processes: 139 CPU Usage: 0\% & Physi & ical Memory: \(59 \%\) & & & & 0 \\
\hline
\end{tabular}

Correct configuration: All "Processes" run with the same "User Name" and with the same "Session ID".

\section*{Summary}

The correct function of OPC Server V2 and V3 can be checked without AC500.
With OPC Server V3 with the configuration "SIMULATION" the Project name with the directory name has to be specified. The values of the items can be read and write by one OPC client.
With OPC Server V2, as well as with OPC Server V3 in configuration "GATEWAY", only the project name may be specified. The configured items are found, but the quality is bad. The values of the items can not be read and not write by one OPC client.

Refer to REF5 Online Help of PS501 chapter OPC for details \({ }^{4}\) ) Table on page 1759.

\section*{Configure AlarmEvents}

\section*{Check AlarmEvents}

The function of the "AlarmEvents" can be checked with "MatrikonOPC Explorer".


The "AlarmEvents" can be simulated by writing the value of the Items.

\section*{Further information}

Refer to REF2 AeConfigurator_UserGuide.pdf for details \(\stackrel{\text { ® Table }}{ }\) on page 1759.

\section*{Configure user account for OPC server}

\section*{OPC server V3 on Windows Server 2003/ 2008/ 2012/ 2016}

When running the OPC server V3 on Windows Server 2003/ 2008/ 2012 /2016 multiple sessions need to be supported. Therefore the installation of the OPC server as service running with a dedicated user account is recommended.

Configuration steps

Create specific user
- Create specific user, no administrator account is required
- Register V3 OPC server as service
- Configure V3 OPC server as service


Registration


Register the OPC Server executable as service from the command line.

\section*{With command "WinCoDeSysOPC /Service" WinCoDeSysOPC.exe gets installed as system service.}

Started once, the service will stay "started" until the system gets terminated.
The communication to the configured PLCs survives.
Also here the service gets installed in the current position of WinCoDeSysOPC.exe.

\section*{Configuration}




At "Computer Management \(\rightarrow\) Services and Applications \(\rightarrow\) Services" open the "Properties" of the "CoDeSysOPCDAService".


Complete the Service Configuration


\section*{Testing}

Check Users and Session during Test Cases
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline \multicolumn{6}{|l|}{ElrommuterMAnMmamant} & \multicolumn{2}{|l|}{_ \(|\square| \times \mid\)} \\
\hline \multicolumn{7}{|l|}{F- Windows Task Manager} & [ \([\) ] \(\times\) ] \\
\hline Ele Sotions Yew Help & & & & & & & \\
\hline \multicolumn{8}{|l|}{Applications Processes |Performance | Networking | Users |} \\
\hline Image Name & PID & Session 10 & User Name & CPU & Mem Usage & & \(\star\) \\
\hline ctfmon.exe & 340 & 0 & Administrator & \(\infty\) & 2,920 K & & \\
\hline Codesys.exe & 1436 & 0 & Administrator & \(\infty\) & 19,796 K & & \\
\hline tasking.exe & 1652 & 0 & Administrator & \(\infty\) & 2,132 K & & \\
\hline explorer.exe & 1996 & 0 & Administrator & \(\infty\) & 22,552K & & \\
\hline mmc.exe & 2360 & 0 & Administrator & \(\infty\) & 23,232 K & & \\
\hline svchost.exe & 812 & 0 & LOCAL SERVICE & 00 & 6,168K & & \\
\hline svchost.exe & 1520 & 0 & LOCAL SERVICE & \(\infty\) & 1,708 K & & \\
\hline wmiprvse.exe & 300 & 0 & NETWORK SERV... & \(\infty\) & 5,364 K & & \\
\hline svchost.exe & 692 & 0 & METWORK SERV... & \(\infty\) & \(4,668 \mathrm{~K}\) & & \\
\hline svchost.exe & 784 & 0 & NETWORK SERV... & \(\infty\) & 5,052 K & & \\
\hline msdtc.exe & 1328 & 0 & NETWORK SERV... & \(\infty\) & 4,476K & & \\
\hline explorer.exe & 1040 & 2 & OPCService & \(\infty\) & 17,912 K & & \\
\hline MobleOPCExplorer.exe & 1464 & 2 & OPCService & \(\infty\) & \(14,748 \mathrm{~K}\) & & \\
\hline msseces.exe & 1872 & 2 & OPCService & \(\infty\) & \(10,956 \mathrm{~K}\) & & \\
\hline Gateway.exe & 1968
2436 & 0 & OPCService & +0 & 4,440K & & - \\
\hline ctmon.exe & 2436
2876 & 2 & OPCService & \(\infty\) & \(3,152 \mathrm{~K}\) & & \\
\hline Wwareiray.exe
rdpclip.exe & 2876
3152 & 2 & OPCService & +0 & \(3,656 \mathrm{~K}\)
\(3,736 \mathrm{~K}\) & & \\
\hline VMwareUser.exe & 3488 & 2 & OPCService & 00 & 5,240K & & \\
\hline Whcioesyropliexe & 3912 & 0 & Cficservise & 0 & 6.464 K & & \\
\hline System Ide Process & 0 & 0 & SYSTEM & 99 & 28 K & & \\
\hline System & 4 & 0 & SYSTEM & 00 & 240 K & & \\
\hline smss.exe & 292 & 0 & SYSTEM & \(\infty\) & 524 K & & \\
\hline csiss.exe & 348 & 0 & SYSTEM & \(\infty\) & \(4,500 \mathrm{~K}\) & & \\
\hline winloaon.exe & 372 & 0 & SYSTEM & 00 & \(4,560 \mathrm{~K}\) & & \(\cdots\) \\
\hline \multicolumn{6}{|l|}{F Show processes from all users} & End Process & \\
\hline \multicolumn{2}{|l|}{Processes: 44 CPUUsage: 0\%} & \multicolumn{4}{|l|}{Commit Charge: \(311 \mathrm{M} / 2143 \mathrm{M}\)} & & \\
\hline A) start \(\theta \in \square\) & \#) Comp & ter Manager & ment \# Wind & ws Ta & sk Manag-. & 车国 \(3^{3} 38\) & 558 \\
\hline
\end{tabular}

Check the "Session ID" and "User Name" of
- Gateway.exe,
- WinCoDeSysOPC.exe and
- OPC Client
on different test cases like multi session with terminal service sessions.

\section*{Further information}


Please refer to REF3 ReadMe.rtf and REF4 ReleaseNotes OPCV3 AE for HA.
\& Table on page 1759

\subsection*{6.3.6.1.3 Potential issues}

\section*{Session isolation}

Situation With Windows Server 2003, Windows Server 2008, Windows Server 2016 the Windows 7 services are alone in session 0 . User applications run in session 1 (2 and so on).
Services:
A Windows service is a computer program that operates in the background.
Windows services can be configured to start when the operating system is started or can be started manually and run in the background as long as Windows is running. They can operate when a user is not logged on.

\section*{Services are:}

Windows operating systems include numerous services. OPC client like S+ OPC scanner PGIM, Aspen CIM-IO Manager, ICONICS, .. can also installed as a service.
User applications are:
Microsoft Word, Notepad, MatrikonExplorer, ControlBuilderPlus.exe and Codesys.exe

Problem Service and user application are isolated in their session. They can not communicate with each other directly.

OPC Server uses, like the CBP and CODESYS, the gateway server from CODESYS (gateway.exe) for the communication with the AC500 and starts the gateway in their session. That creates undefined behavior, if the OPC Server runs as a service. The gateway server is not able to run in multi sessions.

Resolutions - Install all OPC clients and OPC Server, which use the gateway server, in the same session.
- The OPC Server as a service (session 0) may not be connected at the same time (in parallel) with an OPC server as a user application or CBP or CODESYS (all in session 1) with the AC500. If this function is necessary, different PC or virtual machines must be used.
- Use tools like OPC tunnel. In a DigiVis 500 setup context the OPC server must not be registered as service. The OPC tunnel itself starts the OPC server within its service.

\section*{See also}
http://msdn.microsoft.com/en-us/ windows7trainingcourse sessionisolation unit.

\subsection*{6.3.6.2 OPC UA server for AC500 V3 products}

\subsection*{6.3.6.2.1 General}

OPC UA server can be added as an object below the Ethernet interfaces ETH1 or ETH2.
The user can access the variable interface of the PLC via a client. At the same time, communication can be protected by means of encryption.

The CODESYS OPC UA server supports the following features:
- Browsing of data types and variables
- Standard read/write services
- Notification for value changes: subscription and monitored item services
- Encrypted communication according to "OPC UA standard (profile: Basic256SHA256)"
- Imaging of the IEC application according to "OPC UA Information Model for IEC 61131-3"
- Supported profile: Micro Embedded Device server Profile
- By default, there is no restriction in the number of sessions, monitored items, and subscriptions. The number depends on the performance of the respective platform.
- Sending of events according to the OPC UA standard.

Application note How to install and configure an OPC V2 or OPC V3 server and how to use it for AC500 PLCs is described in the application note How to use OPC server.
\(\begin{array}{ll}\text { Application } & \begin{array}{l}\text { The application example } \\ \text { example to } \\ \text { deeper understanding of the OPC server } V 3 \text { - for DA and UA is available to gain a }\end{array} \\ \text { deol and to configure } A C 500 \mathrm{~V} 3 \text { accordingly. }\end{array}\)

\subsection*{6.3.6.2.2 Creating a project for OPC UA access}
1. Click "File \(\rightarrow\) New Project \(\rightarrow\) AC500 project" in Automation Builder 2.1 or newer.
2. Choose a PLC - AC500 V3 and click [Add object].
3. Right-click on node ETH1 or ETH2 and "Add object".
4. Choose OPC UA Server in the dialog and click [Add object].
5. Declare some variables of different types in the program.
6. Right-click "Application \(\rightarrow\) Add object". Choose "Symbol configuration" and click [Add object].
7. Enable checkbox "Support OPC UA Features" in the dialog "Add symbol configuration".
8. Double-click "Symbol configuration" in the Devices tree to open the editor "Symbol configuration".
9. Click [Build].
\(\Rightarrow\) The variables are displayed in a tree structure.
10. Activate the variables that you want to publish to an OPC UA client. Specify the access rights.
11. Download the project to the PLC.

\subsection*{6.3.6.2.3 Use node name}
1. Double-click node "OPC_UA_Server".
2. Set parameter "Use node name" to TRUE.
3. Double-click node "PLC_AC500_V3 <...>".
4. Click "Device" and "Rename active device..."
5. Enter new device name in the following dialog and click [OK].

\subsection*{6.3.6.2.4 Use UaExpert client}

The OPC UA client UaExpert is available for download from the Unified Automation website and can be used free of charge (freeware license).

Using this client, you can connect to the AC500 OPC UA server.
The following description refers to this program. Other OPC UA clients work in a similar way.
1. Start the UaExpert program.

2. Click on the "blue cross symbol".
3. Double-click on the "blue cross symbol" in the "Add Server" dialog.
4. Enter URL and click [OK].
\(\Rightarrow\) The URL appears in the "Add Server" dialog.
5. Select "Advanced" tab and click [OK].
6. Click [Connect].
\(\square\)
7. Expand the project tree in the "Address Space" window.

8. Drag and drop the needed symbols to "Data Access View".

\subsection*{6.3.6.2.5 Working with encryption}

\section*{Creating a certificate for the OPC UA server}
\(\checkmark\) Prerequisite: A battery is inserted and the clock is set to actual time.
1. Double-click the Security symbol in the lower right corner of Automation Builder.
2. Select the "Devices" tab.
\(\Rightarrow\) The certificate information opens.
3. Select the PLC in the left "Information" view.
\(\Rightarrow\) All services of the PLC that require a certificate are displayed in the right Information view.
4. Select the service "OPC UA Server".
5. Click the icon \({ }^{\frac{1}{4} \text { ti }}\) to create a new certificate for the device.
\(\Rightarrow\) "Certificate Settings" dialog appears.

6. Define the certificate parameters according the figure above and click [OK].
\(\Rightarrow\) The certificate is created on the PLC.

7. Upload the certificate to your PC.
8. Restart the runtime system.

For further information see \(\stackrel{\xi}{ }{ }^{\circ}\) Chapter 6.3.4.7.3.4 "OPC UA secure" on page 1728.

\section*{Encrypted connection with UaExpert client}
1. Start the UaExpert program.

2. Click on the "blue cross symbol".
3. Double-click on the "blue cross symbol" in the "Add Server" dialog.
4. Enter URL and click [OK].
\(\Rightarrow\) The URL appears in the "Add Server" dialog.
5. Select "Advanced" tab.

6. Choose option "Basic256ha256" of drop-down list "Security Policy" and "Sign \& Encrypt" of drop-down list "Message Security Mode" and click [OK].

7. Click menu "Settings" and "Manage Certificates"

8. Click [Create new Application Certificate...].
\(\Rightarrow\) Dialog "New Application Instance Certificate" opens.

9. Enter the required informations and click [OK].
\(\Rightarrow\) Dialog "Manage Certificates" opens

10. Click [Copy Application Certificate To...] your PC.

11. Download the certificate to AC500 via the "Security Screen" view.
12. Click [Connect] in the UaExpert client.

\(\Rightarrow\) Dialog "Certificate Validation" opens.

Certificate Validation

Validating the certificate of server 'OPCUAServer@PM5670' returned an error:
BadCertificateChainIncomplete
Certificate Chain
\begin{tabular}{|cl|}
\hline Name & Trust Status \\
8 & OPCUAServer@PM5670 \\
& Trusted \\
& \\
\hline
\end{tabular}

Certificate Details


Working with a trusted certificate will avoid this error message.
14. Enable checkbox Accept the server certificate temporarily for this session and click [Continue].
\(\Rightarrow\) Dialog "Connect Error" opens

15. Click [lgnore]

16. Check settings in dialog "Manage Certificates".

\subsection*{6.3.6.2.6 Changing variables via UaExpert client}
1. Expand in view "Address Space" "Objects \(\rightarrow\) DeviceSet \(\rightarrow\) PM5670 \(\rightarrow\) Resources \(\rightarrow\) Application \(\rightarrow\) PLC_PRG".
\(\Rightarrow\) The variables of the global variable list are visible.

2. Drag and drop the variables to the Data Access View.
3. Change values in the column Value.

\subsection*{6.3.6.2.7 Configuring OPC UA client}

Operating modes
Polling
- Objects will be continuously updated in a defined interval
- Create higher load then Subscription
- Is recommended only for a few Symbols

Pub/Sub Not yet supported

Subscription (recommended mode)
- Updated objects depending on the publishing interval and filters
- Method to reduce load
- Different intervals
- Filter possible (coming in AC500)
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
Client defines a group of sym- \\
bols with
\end{tabular} & Description \\
\hline Publishing interval & Interval, in which server publish data to client \\
\hline Sampling interval & \begin{tabular}{l} 
Interval for sampling and storing data at server and send in \\
each publishing interval
\end{tabular} \\
\hline Queue size & \begin{tabular}{l} 
Array of data to save data if sampling Interval is faster than \\
publishing Interval (At AC500 in the moment only 1)
\end{tabular} \\
\hline Data change filter & \begin{tabular}{l} 
Can be used to reduce traffic from server to client. \\
Criteria: \\
- Change of data, \\
- Change of status \\
- Change of time stamp \\
AC500 is fix configured for change of data and change of \\
status.
\end{tabular} \\
\hline
\end{tabular}


\section*{Using OPC UA with subscription mode}

\section*{Recommendations:}
- Define only variables you need as symbols
- Do not configure publishing Intervals to short (increase load)
- Use different subscriptions with different publishing intervals in order to decrease load
- Do not use sampling intervals faster then publishing intervals as long as AC500 OPC UA server don't support Queue Size different from 1
- Be careful: Setting „0" at sampling Interval at client will be interpreted in server as „as fast as possible", which is 100 ms at AC500 and create a high load.

Publishing and sampling intervals in UaExpert
1. Right-Click on an Item in "Data Access View" and click "Subscription Settings".

2. Set the recommended values.

Life Time Count: Number of publishing intervals in which client has to send publish requests to the server. After this period without request from client, subscription in server will be deletet.
Max Keep Alive Count: If there are no new data to send, server can skip a publishing interval. After the alive count, server has to send, even if there are no new data.

Click [OK].
3. Right-Click on an Item in "Data Access View" and click "Monitored Item Settings".

4. Set the recommended values.

\subsection*{6.3.6.3 Web server}

HTTP library In order to be able to use the PLC as a client for web services, the HTTP function block library can be used. Setup and use of the HTTP library are described in the application example AC500 Webservices - HTTP Function Block Library.

\subsection*{6.3.6.4 Windows server}

Windows server usage in Automation Builder requires modifications of the system environment and of the settings in Automation Builder, e.g. permissions, licensing, session handling etc. How to setup system and software for Windows server is described in the application note Windows Server 2016 - RDM access.

\subsection*{6.3.7 Converting an AC500 V2 project to an AC500 V3 project}
\begin{tabular}{ll} 
Application & \begin{tabular}{l} 
A project that has been configured for an AC500 V2 PLC can be converted to a \\
example
\end{tabular} \\
& \begin{tabular}{l} 
project for an AC500 V3 PLC. Essentially, the conversion is done in Automation Builder, \\
however, some additional actions have to be executed manually. The complete conver- \\
sion procedure of a V2 project to a \(V 3\) project is described in the application note \\
Converting an \(A C 500\) V2 project to an AC500 V3 project.
\end{tabular}
\end{tabular}

\subsection*{6.4 Programming with CODESYS}

\subsection*{6.4.1 CODESYS Development System}

\subsection*{6.4.1.1 Introduction}

CODESYS The CODESYS Development System IEC 61131-3 programming tool forms the core of the System overview CODESYS software platform for tasks in industrial automation technology. With additional, integrated solutions for motion control, visualizations, and fieldbus connections, the usual practical requirements are covered in one system.


Features The free CODESYS Development System is a IEC 61131-3 programming platform for automation devices with control tasks. It provides diverse and comfortable engineering solutions to support you in your developing tasks:
\begin{tabular}{|c|c|}
\hline & For this see in this Online Help: \\
\hline Project configuration through wizards. & ② Chapter 6.4.1.3 "Creating and Configuring a Project" on page 1808 \\
\hline Adaptability of the user interface. & ⓨ Chapter 6.4.1.2.2 "Customizing the user interface" on page 1802 \\
\hline Creation of professional IEC 61131-3 controller applications wit a host of standard features. & \({ }^{\mu} \leftrightharpoons\) Chapter 6.4.1.9 "Programming of Applications" on page 1844 \\
\hline \begin{tabular}{l}
User-friendly programming with mouse and keyboard in all IEC 61131-3 languages. \\
Appropriate editors for FBD, LD, IL, ST, SFC, additionally the variants CFC and Extended CFC.
\end{tabular} & « Chapter 6.4.1.20.1 "Programming Languages and Editors" on page 2047 \\
\hline Input assistance for the input and configuration of data. & y Chapter 6.4.1.9.7 "Using input assistance" on page 1885 \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline & For this see in this Online Help: \\
\hline \begin{tabular}{l}
Support of object-oriented programming. \\
Real object-oriented programming (OOP) fully compliant with the IEC 61131-3 standard in all IEC 61131-3 languages, without any additional tools. \\
Inheritance of POUS to similar application parts to reduce development time and errors. \\
Object-orientation is not a must: Functional and object-oriented programming can be used and mixed as required.
\end{tabular} & ⓨ Chapter 6.4.1.9 "Programming of Applications" on page 1844 \\
\hline Comprehensive project comparison, also for graphic editors. & « Chapter 6.4.1.5 "Comparing projects" on page 1817 \\
\hline Library concept for an easy reutilization of application. & y Chapter 6.4.1.17 "Using Libraries" on page 2034 \\
\hline Debugging and online features for the fast optimization of the application code and to speed up testing and commissioning. & ๕ Chapter 6.4.1.12 "Testing and Debugging" on page 1980 \\
\hline Integrated compilers for many different CPU platforms for optimizing the controller performance. & \begin{tabular}{l}
③ Chapter 6.4.1.21.4.13.3 "Dialog 'Project Environment' - 'Compiler Version'"' on page 2778 \\
\(\stackrel{y}{*}\) Chapter 6.4.1.21.4.12.4 "Dialog Box 'Project Settings' - 'Compileoptions'" on page 2769
\end{tabular} \\
\hline Security features for the protection of the source code and the operation of the controller. & \begin{tabular}{l}
« Chapter 6.4.1.6 "Protecting and Saving Projects" on page 1819 \\
« Chapter 6.4.1.9.18 "Protecting an application" on page 1915 \\
\& \({ }^{4}\) Chapter 6.4.1.11.4 "Handling of Device User Management" on page 1971
\end{tabular} \\
\hline Field bus support and programming of devices from different manufacturers. & «䨿 Chapter 6.4.1.8 "Configuring I/O Links" on page 1835 \\
\hline Extensibility and adaptability without leaving the framework. & \\
\hline
\end{tabular}

Additionally:
Many seamlessly integrated tools for different kinds of automation tasks, for example CODESYS Visualization, CODESYS SoftMotion, CODESYS Application Composer.

Please always note the possibility to extend the functionalities by "AddOn"-Packages, provided in the CODESYS Store.

Customization of the user interface language

In the "Option \(\rightarrow\) International Settings" dialog you can customize the language of the user interface of the development system. This change will take effect the next time you start Automation Builder. You can adjust the help language separately.

If you start Automation Builder from the command line, you can add a parameter to adjust the user interface language.
\(\boldsymbol{y}^{\prime}\) Chapter 6.4.1.21.4.14.14 "Dialog 'Options' - 'International Settings'" on page 2790
Chapter 6.4.1.16 "Using the Command-Line Interface" on page 2028

Copyrights and trademarks

All rights are reserved by the individual copyright holders. Technical specifications are subject to change. Reproduction or further use of this help resp. of parts of it require the express prior authorization of \(A B B A G\).

\subsection*{6.4.1.2 Configuring CODESYS}

\subsection*{6.4.1.2.1 Introduction}

\section*{Setting CODESYS options}

You can configure the behavior and appearance of the CODESYS Development System in the different tabs of the "Options" dialog. The dialog opens by clicking "Tools \(\rightarrow\) Options". Here you can configure the default settings for different editors and functionalities. These settings apply throughout CODESYS.
The settings are stored in your current user profile on your local system. For use on other systems, option settings, either user-specific or machine-specific (computer), can be exported to an XML file.

In V3.5 SP13 and later, CODESYS checks whether an older version is already installed when the development system is started for the first time. If this is the case, then the "Import Assistant" dialog opens for transferring the CODESYS options set with the older version.

See also
- \({ }^{4}\) Chapter 6.4.1.21.3.9.17 "Command 'Options'" on page 2667
- \({ }^{*}\) Chapter 6.4.1.21.3.9.18 "Command 'Import and Export Options'" on page 2668
- \({ }^{\circ}\) Chapter 6.4.1.21.4.2 "Dialog 'Import Assistant"' on page 2745
- \(\quad 4\) Chapter 6.4.1.2.2.2 "Customizing menus" on page 1802
- \({ }^{*}\) Chapter 6.4.1.2.2.5 "Customizing keyboard shortcuts" on page 1805
- \(y^{\prime}\) Chapter 6.4.1.2.2.3 "Customizing toolbars" on page 1804

\subsection*{6.4.1.2 \(\mathbf{2}\) Customizing the user interface}

\section*{Introduction}

In CODESYS, you can customize the user interface by changing the window layout as well as the appearance of menus and commands according to your requirements.

\section*{Customizing menus}

You can customize the menu commands of the CODESYS user interface. In a configuration dialog, you can add or remove menus.

\section*{Removing menus and commands}
1. Choose the command "Tools \(\boldsymbol{\rightarrow}\) Customize".
\(\Rightarrow\) The "Customize" dialog box opens. The "Menu" tab is visible.
2. Select a menu in the menu tree or a command in a menu.
3. Click "Delete".
\(\Rightarrow\) The menu or command is deleted from the menu tree.
4. Click "OK".
\(\Rightarrow\) The dialog box closes and the menu is customized.

\section*{Adding menus}
1. Choose the command "Tools \(\boldsymbol{\rightarrow}\) Customize".
\(\Rightarrow\) The "Customize" dialog box opens. The "Menu" tab is visible.
2. Scroll to the end of the menu tree.
3. Select the blank symbol \(\square)\).
4. Click "Add Popup Menu".
\(\Rightarrow\) The "Add Popup Menu" dialog box opens.
5. Type a name for the new menu in the "Default text" field.

If localization is unnecessary, then skip to step 9.
6. Click "Add Language".
\(\Rightarrow\) A drop-down list opens with available languages.
7. Choose the required language.
\(\Rightarrow\) The language is added to the list of languages.
8. Click into the "Text" field and type the language-specific text.
9. Click "OK".
\(\Rightarrow\) The new menu is added at the bottom of the menu tree.
10. Change the menu order by clicking "Move up" and "Move down". Click "OK" to close the "Customize" dialog box

The new menu is displayed only when it contains a command.

\section*{Adding commands}
1. Choose the command "Tools \(\rightarrow\) Customize".
\(\Rightarrow\) The "Customize" dialog box opens. The "Menu" tab is visible.
2. Expand the branch of the menu where the new command should be added.
3. Select the blank symbol ( \(\square\) ).
4. Click "Add Command".
\(\Rightarrow\) The "Add Command" opens dialog box.
The dialog box lists all commands grouped by category.
5. Select the command to be added. Click "OK".
\(\Rightarrow\) The new command is added to the menu tree.
6. Change the menu order by clicking "Move up" and "Move down". Click "Add separator" to add a border between commands. Click ok to close the "Customize" dialog box.
\(\Rightarrow\) The new command is now available in the menu.

See also
- „y Chapter 6.4.1.21.4.15.2 "Dialog 'Customize' - 'Menu"' on page 2801
- \& Chapter 6.4.1.2.2.3 "Customizing toolbars" on page 1804

\section*{Customizing toolbars}

You can customize the toolbars of the CODESYS user interface. In a configuration dialog, you can add or remove toolbars.

\section*{Removing toolbars and commands}
1. Choose the command "Tools \(\rightarrow\) Customize".
\(\Rightarrow\) The "Customize" dialog box opens.
2. Choose the "Toolbars" tab.
3. Select a toolbar or a command from a toolbar tree.
4. Click "Delete".
\(\Rightarrow\) The toolbar or command is deleted.
5. Click "OK".
\(\Rightarrow\) The dialog box closes and the toolbar or command is removed.

\section*{Adding toolbars}
1. Choose the command "Tools \(\rightarrow\) Customize".
\(\Rightarrow\) The "Customize" dialog box opens.
2. Choose the "Toolbars" tab.
3. Select the blank toolbar.
4. Click "Add Toolbar".
\(\Rightarrow\) The cursor blinks in the new toolbar.
5. Type a name.
6. Change the toolbar order by clicking "Move up" and "Move down". Click "OK" to close the "Customize" dialog box.

CODESYS displays the new toolbar only when it contains a command.

Adding commands
1. Choose the command "Tools \(\rightarrow\) Customize".
\(\Rightarrow\) The "Customize" dialog box opens.
2. Choose the "Toolbars" tab.
3. Expand the tree of the toolbar where the new command should be added.
4. Select the blank symbol
5. Click "Add Command".
\(\Rightarrow\) The "Add Command" dialog box opens.
The dialog box lists all commands grouped by category.
6. Select the command to be added. Click "OK".
\(\Rightarrow\) The new command is added to the toolbar tree.
7. Change the toolbar order by clicking "Move up" and "Move down". Click "Add separator" to add a border between commands. Click "OK" to close the "Customize" dialog box.
\(\Rightarrow\) The new command is available in the toolbar.

See also
- \(\boldsymbol{y y}^{4}\) Chapter 6.4.1.21.4.15.4 "Dialog 'Customize' - 'Toolbars'" on page 2802
- \(\Leftrightarrow\) Chapter 6.4.1.2.2.2 "Customizing menus" on page 1802

\section*{Customize command icon}

CODESYS provides the capability of assigning customized icons to commands.
1. Select the command "Tools \(\rightarrow\) Customize".
\(\Rightarrow\) The "Customize" dialog box opens.
2. Click the "Command icons" tab.
3. Select the category "Help" from the list on the left.
\(\Rightarrow\) All commands in this category are listed on the right.
4. Select the command "Information".
5. Click "Assign".
\(\Rightarrow\) A dialog box opens for selecting the icon file (*.ico).
6. Select an icon file.
7. Click the "Open" button.
\(\Rightarrow\) The icon is assigned to the selected command.
8. Click "OK".

See also
- \(\quad y^{2}\) Chapter 6.4.1.21.4.15.3 "Dialog 'Customize' - 'Command Icons' " on page 2801

\section*{Customizing keyboard shortcuts}

CODESYS provides the capability of executing commands directly via keyboard shortcuts. You can customize or extend predefined keyboard shortcuts.
1. Choose the command "Tools \(\rightarrow\) Customize".
\(\Rightarrow\) The "Customize" dialog box opens.
2. Choose the "Keyboard" tab.
3. Select the category "Help" from the list on the left.
\(\Rightarrow\) All commands in this category are listed on the right.
4. Select the command "Search".
5. Click into the field "Press Shortcut Keys".
6. Press [Ctrl]+[Shift]+[S].
\(\Rightarrow\) CODESYS adds the key combination to the field.
7. Click "Assign".
\(\Rightarrow\) The keyboard shortcut is assigned to the command.
8. Click "OK".
\(\Rightarrow\) You can call the "Search" command by pressing [Ctrr]+[Shift]+[S].

See also
- \# Chapter 6.4.1.21.4.15.5 "Dialog Box 'Customize' - 'Keyboard' " on page 2802

\section*{Changing the window layout}

In CODESYS, you can easily customize the layout of different views to your individual needs.
1. Drag the view by the caption bar or by the tab.
\(\Rightarrow\) Arrows are shown to mark possible destinations. Example:
2. Drag the view to one of the arrows.
\(\Rightarrow\) The destination is displayed as a blue-shaded area.
3. Release the left mouse button.
\(\Rightarrow\) The window is inserted into the selected destination.

The window can also be placed outside of the CODESYS programming interface.

See also
- © Chapter 6.4.1.2.2.7 "Resizing windows" on page 1806
- © Chapter 6.4.1.2.2.8 "Auto-hiding windows" on page 1807
- © Chapter 6.4.1.2.2.9 "Switching between windows" on page 1807

\section*{Resizing windows}
1. Move the mouse pointer over the border between two windows or views.
\(\Rightarrow\) The cursor becomes a left-right arrow.
2. Drag the border to another position.

> You can resize detached views by moving the frame lines.


See also
- \({ }^{*}\) Chapter 6.4.1.2.2.6 "Changing the window layout" on page 1806
- \& Chapter 6.4.1.2.2.8 "Auto-hiding windows" on page 1807
- \(\xi_{\text {- }}\) Chapter 6.4.1.2.2.9 "Switching between windows" on page 1807

\section*{Auto-hiding windows}

Hiding windows When you hide a view, it is minimized to a tab in the frame of the user interface. When you move the pointer over the tab, the window is shown automatically.
1. Click into the window to be hidden.
2. Click "Window \(\rightarrow\) Auto Hide".

Or click the PIN symbol (푸) in the upper right corner of the view.
\(\Rightarrow\) The window is hidden and only visible by a small tab on the edge of the main window.
3. Move the mouse pointer over the tab.
\(\Rightarrow\) The window is shown as long as the mouse pointer hovers over the tab.

\section*{Showing windows}
1. Click the tab of the hidden window.
2. Clear the check box "Window \(\rightarrow\) Auto Hide".

Or click the PIN symbol ( \(-\boldsymbol{\square}\) ) in the upper right corner of the view.
\(\Rightarrow\) The window is permanently shown.

See also
- \({ }^{*}>\) Chapter 6.4.1.2.2.6 "Changing the window layout" on page 1806
- \(\&\) Chapter 6.4.1.2.2.7 "Resizing windows" on page 1806
- \(\sum_{\text {• }}\) Chapter 6.4.1.2.2.9 "Switching between windows" on page 1807

\section*{Switching between windows}

It is possible to switch directly between the currently opened views and the editor windows.
1. Press the keystroke combination [Ctrl]+[Tab]. Continue pressing the [Ctrl] key.
\(\Rightarrow\) An overview opens with all active views and editors.
2. Continue pressing the [Ctrl] key and select a window using the arrow keys.
3. Release the [Ctrl] key.
\(\Rightarrow\) The selected view or editor is activated.

See also
- \({ }^{*}>\) Chapter 6.4.1.2.2.6 "Changing the window layout" on page 1806
- \(\quad\) Chapter 6.4.1.2.2.7 "Resizing windows" on page 1806
- \(\Leftrightarrow\) Chapter 6.4.1.2.2.8 "Auto-hiding windows" on page 1807

\subsection*{6.4.1.3 Creating and Configuring a Project}

\subsection*{6.4.1.3.1 Introduction}

What is a - A project contains the objects which are necessary to create a controller program ("applicaproject? tion"):
- Pure POUs, for example programs, function blocks, functions, and GVLs.
- Objects that are also required to be able to run the application on a PLC. For example, task configuration, Library Manager, symbol configuration, device configuration, visualizations, and external files.
- In a project, you can program multiple applications and connect multiple controller devices.
- CODESYS manages device-specific and application-specific POUs in the "Devices" view ("device tree") and project-wide POUs in the "POUs" view.
- For the creation of projects, there are templates that already contain certain objects.
- Basic configurations and information for the project are defined in the "Project Settings" and "Project Information". For example:
- Compiler settings
- User management
- Author
- Data about the project file

There are settings for the version compatibility of the project in the configuration dialogs in the "Project Environment".
- You save a project as a file in the file system. As an option, you can pack it together with project-relevant files and information into a project archive. It is also possible to save files in a source code management system such as SVN.
- Each project contains the information about the CODESYS version with which it was created. When you open it in another version, CODESYS will notify you about possible or necessary updates regarding file format, library versions, etc.
- You can compare, import/export projects, and create documentation for them.
- You can protect a project from being changed, or even completely protect it from being read. By using user management, you can selectively control the access to the project and even to individual objects in the project.
- \(\Leftrightarrow\) Chapter 6.4.1.21.2.2 "Object 'Application"' on page 2410
- \(\&\) Chapter 6.4.1.21.2 "Objects" on page 2409
- \(\underbrace{*}\) Chapter 6.4.1.21.4 "Dialogs" on page 2745
- « \({ }^{\circ}\) Chapter 6.4.1.21.3.5.13 "Command 'Project information'" on page 2604
- \(\Rightarrow\) Chapter 6.4.1.6 "Protecting and Saving Projects" on page 1819

\subsection*{6.4.1.3.2 Opening a V3 Project}

You can open projects, library projects, or project archives in CODESYS which have been created with different installations. When a project is opened, it is automatically checked whether or not the active installation is appropriate to load the project. At this time, deficiencies can be detected, such as missing add-ons or deprecated installations. You can correct these deficiencies. Then you can load the project with an appropriate installation.

The following actions are possible to correct deficiencies:
- Update existing add-ons and install missing add-ons
- Start another installation which is appropriate for the project
- Install an additional CODESYS version with the appropriate state

Loading a write- Moreover, you can load and read write-protected projects. You have to specify an appropriate protected location where you have the necessary write permissions only when you save the file.

Loading a You can load restricted projects only if you have the access credentials, such as user name and project with access restrictions
\begin{tabular}{ll} 
Loading a & \begin{tabular}{l} 
You have selected a project which is protected by a security key. If the security key is not \\
project with a \\
pecurity key
\end{tabular}
\end{tabular} \begin{tabular}{l} 
project without any information about the protection.
\end{tabular}

Loading a Backups are created when the "Automatically save" project option is selected. When CODESYS backup of a project password. is not ended properly after a change, the project is saved as a backup.

When you have selected a project, the "Auto Save Backup" dialog opens first when loading. There you can handle the backup.

See also
- \(y^{n}\) Chapter 6.4.1.3.3 "Opening a V2.3 project" on page 1809
- \({ }^{4}\) ) Chapter 6.4.1.21.4.14.17 "Dialog 'Options' - 'Load and Save'" on page 2791
- \({ }^{4}\) Chapter 6.4.1.6.2 "Setting up write protection" on page 1823
- \(y^{4}\) Chapter 6.4.1.6.3 "Assigning Passwords" on page 1824
- \({ }^{4}\) Chapter 6.4.1.21.3.2.2 "Command 'Open Project"' on page 2553

See also
- Help on CODESYS Installer

\subsection*{6.4.1.3.3 Opening a V2.3 project}

> A CoDeSys V2.3 project can be converted into a CODESYS V3 project only if the CODESYS V2.3 Converter package is installed in CODESYS V3. The package is available in the CODESYS Store.

Requirement: CODESYS is started (or a project is already open). You should be aware of the restrictions described below the following instructions.
1. Click "File \(\rightarrow\) Open Project".
2. In the "Open Project" dialog, click any CoDeSys V2.3 project or project archive in the file system. For searching, you can set the file filter on the bottom right corner of the dialog.
\(\Rightarrow\) If another project is still open, CODESYS instructs you to close it accordingly. After that the CoDeSys V2.3 converter automatically starts.
3. The V2.3 converter checks that the project can be compiled without errors. If so, then it processes the project automatically.
4. NOTE: If the project contains visualization objects with placeholder variables that the converter cannot resolve, the respective visualizations are shown as a group in place of the visualization references.
5. Device conversion: When a device (target system) is referenced in the project to be opened and no conversion rules are defined for the device, then the "Device Conversion" dialog opens. Specify here whether and how the converter should replace the previous device reference with a current one.
\(\Rightarrow\) For replacement, the converter added the new device in the place of the old one in the device tree of the converted project.
6. Library conversion: if a library, for which no conversion rule has so far been defined, is referenced in the project to be opened, then the "Conversion of Library Reference" dialog opens. Specify here whether and how the converter should replace the existing library reference with a current one. If you select a library for which the project information is missing, then the "Enter Project Information" dialog opens in order to specify this information.
\(\Rightarrow\) The converter loads the adapted project. Note: The redefined library references are to be found in the global Library Manager in the POUs view.

\section*{Restrictions when reusing a CoDeSys V2.3 project in CODESYS}
\begin{tabular}{|c|c|c|c|c|}
\hline Com-pilation: & \multicolumn{4}{|l|}{\begin{tabular}{l}
The project has to be compilable without errors in CoDeSys V2.3. Note: CODESYS stills issues warnings in V3 when compiling. These are caused by implicit conversions, which can lead to a loss of information (for example through sign changes). \\
CODESYS checks "case" statements against the switch variable: CASE USINT OF INT is not checked in CoDeSys V2.3, but it issues an error message when imported into V3.
\end{tabular}} \\
\hline Controller config-uration: & \multicolumn{4}{|l|}{The "Controller Configuration" of a CoDeSys V2.3 project cannot be imported into V3. You have to recreate the device configuration and re-declare the variables used in the controller configuration.} \\
\hline Network variables: & \multicolumn{4}{|l|}{For network variables, CODESYS creates V3 GVL objects and imports the variable declarations. However, the network properties are not imported. See the description of the network variable exchange for this.} \\
\hline Libraries: & \multicolumn{4}{|l|}{All variables and constants that are used in a library also have to be declared in the library. It must be possible to compile the library in CoDeSys V2.3 without errors.} \\
\hline \begin{tabular}{l}
Syn- \\
tactic \\
and \\
seman \\
tic \\
restrict \\
ions \\
since \\
CoDe \\
Sys \\
V2.3:
\end{tabular} & \multicolumn{4}{|l|}{\begin{tabular}{l}
- FUNCTIONBLOCK is not a valid keyword instead of FUNCTION_BLOCK. \\
- TYPE (declaration of a structure) must be followed by a ":". \\
- ARRAY initialization** must have parentheses. \\
- INI is no longer supported (you have to replace this in the code by the Init method). \\
- In function calls it is no longer possible to mix explicit with implicit parameter assignments. Therefore the order of the parameter input assignments can be changed: \\
fun(formal1 := actual1, actual2); // -> error message fun(formal2 := actual2, formal1 := actual1); // same semantics as the following line: fun(formal1 := actual1, formal2 := actual2); \\
- CoDeSys V2.3 pragmas are not converted. They produce an warning in V3. \\
- The TRUNC operator now converts to the data type DINT instead of INT. CODESYS automatically adds a corresponding type conversion for a CoDeSys V2.3 import.
\end{tabular}} \\
\hline Visu-alization: & & & & \\
\hline Placeholder \(s\) and their replac ement & Placeholders & VAR_INPUT & Usage & Replacement \\
\hline & PLC_PRG. \$Local Var\$.aArr[0] & localVar: MyStruct; & \[
\begin{aligned}
& \text { localVar.aArr [0 } \\
& ]
\end{aligned}
\] & \[
\begin{aligned}
& \text { localVar : }= \\
& \text { PLC_PRG.myStruc } \\
& \text { tVar }
\end{aligned}
\] \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|}
\hline & \$Var\$. aArr [0] & Var : MyStr & Var.aArr [0] & Var : \(=\)
PLC_PRG.myStruc
tVar \\
\hline & PLC_PRG.myStr
ctVar.aArr [\$In
dex\$] & Index : INT & PLC_PRG.myStruc
tVar.aArr[Index
\(]\) & Index : \\
\hline Prob-
lemati
c pla-
cehold
ers & \multicolumn{4}{|l|}{\begin{tabular}{l}
- Placeholders within a text: \\
Text: \$axle\$-Axis \\
Correction: \\
localVar : STRING; \\
Text: \%s-Axis \\
Text variable: localVar \\
- Placeholder describes only one part of a variable name: \\
axis\$axis\$spur\$spur\$.fActPosition \\
Correction: \\
Define only one placeholder for the axis\$axis\$spur\$spur\$ placeholder. \\
axis_spur : MyFunctionBlock; \\
Then directly transfer the corresponding instance of the function block. \\
axis_spur := PLC_PRG.axis1spur2; \\
- Placeholder is replaced by an expression: \\
\$Expression\$ -> PLC_PRG.var1 + PLC_PRG.var2 \\
Correction: \\
You must transfer the expression to an auxiliary variable and then transfer this auxiliary variable as an instance. \\
- The placeholder describes a program name: \$Program\$.bToggle -> PLC_PRG.bToggle D \\
The converter cannot transfer this form of setting placeholders in V3. However, you will rarely use it in practice. \\
- Placeholder is replaced by different types: \\
\$Var\$ \\
-> replacement 1: PLC_PRG.n (INT) \\
-> replacement 2 : PLC_PRG.st (STRING) \\
Correction: \\
Define two different placeholders in the interface for this. \\
- The visualization is located in a library. You replace the placeholder later from any desired project when you use the visualization there. \\
Correction: \\
Here you have to replace the TYPE_NONE data types manually. However, there is also the possibility for you to integrate the library in a project and the placeholder is correctly replaced. If you now import this project, the data type is also determined correctly in the library.
\end{tabular}} \\
\hline Nonimport able elements: & \multicolumn{4}{|l|}{Trend, ActiveX - the import is not possible, because the implementation differs a great deal. In V3, a corresponding warning is issued and a corresponding manual reproduction is required.} \\
\hline Pro-
gram-
ming
lan-
guage
s & ST, IL, FBD: & \multicolumn{3}{|l|}{No restrictions} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|}
\hline & LD： & CODESYS imports function blocks with parallel branches in such a way that the part before the branch is repeated for each branch． This corresponds to the generated code that CoDeSys V2．3 cre－ ates for parallel branches． \\
\hline & SFC： & \begin{tabular}{l}
－Step variables explicitly declared by the user must be declared locally in the SFC editor．You may not declare them as VAR＿INPUT，VAR＿OUTPUT or VAR＿INOUT，because CODESYS cannot automatically adapt the calls．Explanation： Steps no longer use Boolean variables for the management of the internal states in \(V 3\) ，but also structures of the type SFCStepType． \\
－Identifier：the following identifiers may not begin with an underscore character： \\
－Names of IEC actions in the tree \\
－Variables that are called in an IEC association list \\
－Names of transitions that have been programmed out \\
Explanation：In V3 the implicit variables that CODESYS creates for actions are given an underscore character as prefix．An invalid identifier with a double underscore character would result．
\end{tabular} \\
\hline & CFC： & \begin{tabular}{l}
－Large boxes：The layout of large boxes can lose quality due to an import．The boxes may overlap one another too much． （Correction planned）． \\
－Macros：Macros cannot be imported．（Correction planned）．
\end{tabular} \\
\hline
\end{tabular}
＊\({ }^{4}\) Chapter 6．4．1．21．2．24＂Object＇Project Information＇＂on page 2515
を \({ }^{4}\) Chapter 6．4．1．21．4．5＂Dialog＇Device Conversion＇＂on page 2747
＊ \(\boldsymbol{y}^{4}\) Chapter 6．4．1．21．4．3＂Dialog＇Library Reference Conversion＇＂on page 2746

\section*{6．4．1．3．4 Configuring a Project}

\section*{General}

You can configure your project using the following dialogs：
－＂Project Settings＂：Basic settings on the behavour of editors and of the compiler，on user management etc．
Chapter 6．4．1．21．4．12＂Dialog＇Project Settings＂＂on page 2766
－＂Project Information＂：Adding of individual and tagging information to the project乡 \({ }^{*}\) Chapter 6．4．1．21．2．24＂Object＇Project Information＇＂on page 2515
－＂Project Environment＂：Defining which versions of the external and internal modules should be used，with the aim of achieving up－to－dateness and compatibility with each other．
凶 Chapter 6．4．1．21．4．13＂Dialog＇Project Environment＇＂on page 2778

\section*{Retrieving and Editing Project Information}

You can use the "Project Information" object to retrieve information about your project and the associated file, and edit certain information.

The object contains information about
- File attributes
- Meta-information, such as manufacturer, title, or author
- Properties with keys
- Statistics
- Licensing
- Signing: This way of signing translated libraries is deprecated, and for security reasons should only be used if compatibility with older versions is required. If this method is used, then later you can use a public key token to verify that the library was last signed by the library vendor. As a library vendor, it is therefore crucial that you make the public key used available to the customer, for example in the documentation.

CODESYS saves the project information as an object within the project. When you transfer a project to another system, the "Project Information" object is transferred with it. There is no need for a project archive.

You can use property keys to access the project information externally via function blocks. For a library project, you can also query information about the licensing.

\section*{Editing metainformation}
1. Click "Project \(\rightarrow\) Project Information".
\(\Rightarrow\) The "Project Information" dialog opens.
2. Click the "Summary" tab.
3. Specify your data in the input fields (example: 0.0.0.1 in the "Version" input field).
\(\Rightarrow\) CODESYS creates a property with a key for each given value and manages them on the "Properties" tab. For a library project, CODESYS still uses the properties and sorts later in the library repository.

If you select the option for CODESYS to create a functions block for these properties, then you can access the properties programmatically.

\section*{Creating func- \\ tions for \\ accessing properties}
1. Click "Project \(\rightarrow\) Project Information".
\(\Rightarrow\) The "Project Information" dialog opens.
2. Select the "Automatically generate 'Project Information' POUs" option.

\section*{Example Requirement: The following property is defined.}
\[
\begin{aligned}
& \text { Key }=\text { nProp1 } \\
& \text { Type= number } \\
& \text { Value }=333
\end{aligned}
\]
1. Select the "Automatically generate 'Project Information' POUs" option.
2. Declare a property of the type DINT, for example showprop : DINT; .
3. Call the function GetNumberProperty: showprop := GetNumberProperty("nProp1");
\(\Rightarrow\) You are granted access to the value in the application.


Note: The functions that are created with the "Automatically generate 'Project Information' POUs" option can be used only if the runtime supports the WSTRING data type. If this is not the case, then instead you can apply the "Automatically generate 'Library Information' POUs" option. You can use the functions created in this way at least in the application to access properties. These functions are not registered in the runtime.

\section*{Licensing library projects}

If your project is a library project, then you can activate the library licensing in use here. The CODESYS Security Key is a dongle.
\(\checkmark\) Requirement: The project is a library project.
1. Click "Project \(\rightarrow\) Project Information".
\(\Rightarrow\) The "Project Information" dialog opens.
2. Click the "Licensing" tab.
3. Select the "Activate dongle licensing" option.
4. Specify the dongle data in "Firm code", "Product code", "Activation URL", and "Activation mail".
\(\Rightarrow\) The library is licensed.

\section*{Creating private \\ key files}
1. Click "Project \(\rightarrow\) Project Information".
2. Click the "Signing" tab.
3. Click the "Create Private Key File" button. \(\Rightarrow\) The "Create Private Key File" dialog opens.
4. Select a safe location, e.g. D: \for lib developers only \(\backslash\) mycomp_libkey.libpk and exit the dialog with "Save".
- \({ }^{\text {• }}\) Chapter 6.4.1.21.2.24 "Object 'Project Information'" on page 2515

\section*{Making project settings}

You can configure settings that affect the behavior of CODESYS and that of certain editors in the "Project Settings" object. The settings are valid throughout the project and are applied immediately for active editors. You can also access the dialog boxes of the object with the command "Project \(\rightarrow\) Project Settings".
CODESYS saves the project settings as an object directly in the project. If you then transfer a project to another system, the "Project Settings" object is also transferred with it, without a project archive being required.

\section*{See also}
- \# Chapter 6.4.1.9.13.3 "Analyzing code statically" on page 1904
- \& Chapter 6.4.1.21.3.5.14 "Command 'Project Settings'" on page 2604
- \& Chapter 6.4.1.21.4.12.2 "Dialog 'Project Settings' - 'SFC"' on page 2767
- © Chapter 6.4.1.21.4.12.3 "Dialog 'Project Settings' - 'Users and Groups"" on page 2768
- « Chapter 6.4.1.21.4.12.4 "Dialog Box 'Project Settings' - 'Compileoptions"' on page 2769
- \#y Chapter 6.4.1.21.4.12.5 "Dialog Box 'Project Settings' - 'Compiler Warnings"' on page 2770
- \& Chapter 6.4.1.21.4.12.6 "Dialog 'Project Settings' - 'Source Download"' on page 2770
- \(\Leftrightarrow\) Chapter 6.4.1.21.4.12.7 "Dialog 'Project Settings' - 'Page Setup"' on page 2772
- \# Chapter 6.4.1.21.4.12.8 "Dialog 'Project Settings' - 'Security"' on page 2772
- \(\Leftrightarrow\) Chapter 6.4.1.21.4.12.9 "Dialog 'Project Settings' - 'Static Analysis Light"' on page 2773
- \({ }^{*}\) Chapter 6.4.1.21.4.12.10 "Dialog 'Project Settings' - 'Visualization"' on page 2776
- \(\xi_{\boldsymbol{y}}\) Chapter 6.4.1.21.4.12.11 "Dialog 'Project Settings' - 'Visualization Profile"' on page 2777

\subsection*{6.4.1.4 Exporting and Transferring Projects}

\subsection*{6.4.1.4.1 General}

Export and import functions are available to you for the exchange of the data from CODESYS projects with other programs.
An exchange of CODESYS projects between CODESYS development systems takes place by way of a copy of the project file (*. project) or project archive (*.projectarchive).

See also
- \(y^{*}\) Chapter 6.4.1.4.2 "Exporting and importing projects" on page 1815
- \(\sum^{4}\) Chapter 6.4.1.4.3 "Transferring Projects" on page 1816

\subsection*{6.4.1.4.2 Exporting and importing projects}

CODESYS offers commands for the export and import of objects to and from a file. Two possibilities are available to you here:
- Export to or import from a CODESYS XML file (*.export) This format is completely compatible with the CODESYS project format. The objects are saved in a machine-readable XML format.
- Export to or import from an XML file in the PLCopen format (*.xml)

You can use this format to exchange information with other programs (for example program editors or documentation tools). PLCopen XML defines a subset of the elements known in CODESYS. 100\% compatibility is thus not guaranteed.

Exporting proj- Requirement: A project is open in CODESYS. ects
1. Select the command "Project \(\boldsymbol{\rightarrow}\) Export..." or "Project \(\boldsymbol{\rightarrow}\) Export PLCopenXML"
2. Select the objects that you wish to export in the dialog box "Export" or "Export PLCopenXML".
3. Click on "OK".
4. Enter the file name and the location and click on "Save".

Importing proj- Requirement: A project is open in CODESYS. ects
1. Select the command "Project \(\rightarrow\) Import..." or "Project \(\rightarrow\) Import PLCopenXML".
2. In the dialog box "Import" or "Import PLCopenXML", select the export file that you wish to import.
\(\Rightarrow\) A dialog box opens and displays the objects in a tree structure, which can be inserted at this point.
3. Select the object in the object tree, under which the objects to be imported are to be inserted.
4. Select the objects and click "OK".
\(\Rightarrow\) The objects are added to the existing object tree.

See also
- \(\quad\) ⓨ Chapter 6.4.1.21.4.14.20 "Dialog 'Options' - 'PLCopenXML" on page 2793
- \({ }^{*}\), Chapter 6.4.1.21.3.5.26 "Command 'Export PLCopenXML"' on page 2612
- \({ }^{*}\) Chapter 6.4.1.21.3.5.27 "Command 'Import PLCopenXML"' on page 2612
- \({ }^{4}\) Chapter 6.4.1.21.3.5.25 "Command 'Import'" on page 2612

\subsection*{6.4.1.4.3 Transferring Projects}

If you wish to transfer a project to another computer and connect from there to the same PLC, without an online change or download being required, observe the following points.
- Make sure that the project requires only fixed versions of libraries (exception: interface libraries), visualization profile and compiler.
- Make sure that the boot application is up to date.

Then create a project archive, which you unpack on the other computer.

Transferring a project to another system

Requirement: A project is open on computer "PC1" that you transfer to another computer "PC2" and reconnect from there to the same controller.
1. Make sure that only libraries with fixed versions are integrated in the project, with the exception of pure interface libraries. To do this, open the "Library Manager" and check all entries that have a "*" instead of a fixed version specification.
2. Make sure that a fixed compiler version is set in the project settings. To check, select "Project \(\rightarrow\) Project Settings" and the "Compiler Options" category.
3. Make sure that a fixed visualization profile is defined in the project settings. To check, select "Project \(\rightarrow\) Project Settings" and the "Visualization Profile" category.
4. Make sure that the application that is presently open is the same as that which is presently in use on the PLC. This means that the "boot application" must be identical to the project in the programming system. To check, look at the project name in the title bar of the programming system window: If an asterisk is displayed behind the name, this means that the project has been modified, but not yet saved. It is then possible that the application and boot application do not correspond!
In this case, first create a (new) boot application. It depends on the PLC and the application properties, whether this takes place automatically during the download of the application. For explicit creation, select the command "Online \(\rightarrow\) Create boot application". Then execute a download with the help of the commands "Online \(\rightarrow\) Login" and "Online \(\rightarrow\) Load".
After that, start the application on the controller with the command "Debug \(\rightarrow\) Start".
\(\Rightarrow\) Now the desired application is running on the PLC, to which you wish to reconnect from the same project later on PC2.
5. Generate a project archive: Select "File \(\rightarrow\) Project Archive \(\rightarrow\) Save/Send Archive". In the "Project Archive" dialog box, also select the following information:
- "Download information files"
- "Library profile"
- "Referenced devices"
- "Referenced libraries"
- "Visualization profile"

Save the project archive in a place that is accessible by PC2.
6. Log out from the controller: To do this, select "Online \(\rightarrow\) Logout". You can stop and restart the PLC without reservations, before you reconnect from PC2.
7. Extract the project archive to PC2: Select "File \(\rightarrow\) Project Archive \(\rightarrow\) Extract Archive" and open the archive saved above. In the "Extract Project Archive" dialog box, activate the same information as described above when generating the archive.
8. Open the project and log in to PLC "xy" again.
\(\Rightarrow\) CODESYS does not demand an online change or download; the project runs.

See also
- \(\quad>\) Chapter 6.4.1.21.4.12.4 "Dialog Box 'Project Settings' - 'Compileoptions"' on page 2769
- Project Settings - Visualization Profile
- \(y^{4}\) Chapter 6.4.1.21.3.7.4 "Command 'Create Boot Application"" on page 2627
- \({ }^{4}\) Chapter 6.4.1.21.3.2.8 "Command 'Save/Send Archive"' on page 2557

\subsection*{6.4.1.5 Comparing projects}

\subsection*{6.4.1.5.1 Introduction}

You can compare the currently open project with another project - a reference project. The differences in contents, properties, or access rights are detected and shown in a comparison view.

\section*{Application example}

How to compare Automation Builder projects for an
AC500 V3 PLC is decribed in the application note
Project Comparison - Compare AC500 Projects and Integrate the Differences.

Clicking "Project \(\rightarrow\) Compare" opens the "Project Compare" dialog for you to configure and run the comparison. Then the result is shown in the comparison view "Project Compare Differences" where the objects are aligned in a tree structure. Objects that indicate differences from the respective reference object are identified by colors and symbols. This is how you detect whether or not the contents, properties, or access rights are different.

For differences in the contents, you can also open the detailed compare view "Project Compare - <object name> Differences" in order to zoom into the object. In the detailed compare view, the contents of the object and reference object are displayed or their source code aligned. The detected differences are marked. Previously opened views are not closed. In this way, you can have any number of comparison views open and read them, in addition to the project compare view.

You can accept the detected differences from the reference project into the current project. This is possible only from the reference project into the open project. To do this, you activate differences (for example in the code) that should be accepted in the current project with the commands \(\approx, \downarrow\), or \({ }^{3}\) in the active comparison view for accepting. These positions are highlighted in yellow. Make sure that any other open compare views are inactive (write-protected, read-only). therefore, you can activate differences to be accepted in exactly one comparison view only. When exiting the active compare view, if you confirm that the differences that are activated for acceptance are actually accepted into the current project, then the current project is modified.
In order to exit the project comparison completely, close the project compare view.

\subsection*{6.4.1.5.2 Creating a comparison view}

Requirement: You have made changes in your current project and wish, for example, to compare it with the last-saved version. In the meantime, for example, you have added further POUs, removed a POU, changed single lines of code or the object properties in function blocks.
1. Select the command "Project \(\rightarrow\) Compare".
\(\Rightarrow\) The "Project Comparison" dialog box opens.
2. Enter the path to the reference project, for example the path to the last-saved version of your current project.
3. Leave the activation of the comparison option "Ignore Spaces" as it is.
4. Click on "OK".
\(\Rightarrow\) The comparison view opens. Title: "Project Comparison - Differences". The Device trees of the current project and the reference project are displayed alongside each other and the changed objects are marked in color.
5. Select an object marked in blue in the tree of the reference project (right). The current project no longer contains this object.
Click on \(\checkmark\) "Accept Single"
\(\Rightarrow\) The object is added to the tree of the current project (left). The line has a yellow background. * \({ }^{\text {a }}\) appears in the middle column.
6. Select an object marked in green in the tree of the current project (left). The reference project does not contain this object.
Click on \(\checkmark\) "Accept Single"
\(\Rightarrow\) The object is removed again from the tree of the current project (left). The line has a yellow background. \({ }^{x_{5}}\) appears in the middle column.
7. If changes are detected in the content of an object that is contained in both the current project and the reference project, this is indicated by red lettering. You can then switch to the detailed comparison view for the object by double-clicking on the object.
8. Close the comparison view and answer the query whether the changes made are to be saved with "Yes".
\(\Rightarrow\) The changes become effective in the project.

\subsection*{6.4.1.5.3 Opening the detailed compare view}

Requirement: For example, a user modified the code in a POU of the current project. You have performed the project comparison by clicking "Project \(\rightarrow\) Compare". The project compare view shows this POU highlighted in red in the aligned in the project tree.
1. Double-click the line of the aligned POU versions.
\(\Rightarrow\) The compare view switches to the detailed compare view of the POU. The modified code lines are highlighted in gray and written in red.
2. Click
\(\Rightarrow\) Code lines with changes (red) are extended by two lines: an line with insert (left, green) and a line with delete (right, blue).
3. Click \(\stackrel{V}{-}\) again.
\(\Rightarrow\) The code line is marked again as modified.
4. Move the mouse pointer to the code line marked as modified and click \(\downarrow\) "Accept Single".
\(\Rightarrow\) The code line from the reference project is activated for acceptance into the current project.
5. Click \(\mathcal{F}\).
\(\Rightarrow\) The project compare view opens for the entire project. It is write-protected (read-only) to prevent you from activating differences for acceptance. The link highlighted in yellow above the tree view also indicates this.
6. Click the link: "Project compare view is read only because there are uncommitted changes in another view. Click here to switch to the modified view."
\(\Rightarrow\) The detailed compare view opens again. The unconfirmed changes are highlighted in yellow.
7. Click \(\times\) in the tab of the view and confirm that the changes should be saved.
\(\Rightarrow\) The detail project view is closed and the POU is overwritten. Now it corresponds to the POU of the reference project. The project view is active again so that you can continue working with project compare.


If you do not click the link, but click \(\times\) instead to close the editor of the project compare view, then you will also confirm the acceptance of changes into the current project. The detail changes are accepted and then the project compare is closed completely.

See also
- \(\boldsymbol{y}^{*}\) Chapter 6.3.1.1.7 "Comparing projects" on page 1405
- «y Chapter 6.4.1.21.3.5.21 "Command 'Compare"' on page 2607
- \(\boldsymbol{y}^{\boldsymbol{y}}\) Chapter 6.4.1.5.2 "Creating a comparison view" on page 1818

\subsection*{6.4.1.6 Protecting and Saving Projects}

\subsection*{6.4.1.6.1 Introduction}

General information about write and access protection

You can protect a project against unintentional changes by means of access and write protection. You can also provide it with read protection (knowledge protection).

Write protection:
The following options are available for providing the entire project with simple write protection:
- Select the "Open Read-Only" option when opening the project.
- You set the "Released" status in the "Project Information".
- You select the "read-only" option in the properties of the project file in the local file system.

In order to protect only certain objects in a project against changes, or to allow access only to certain users, you can use a user and access rights management (see below). Some target devices similarly support user and rights management. The access of CODESYS to objects and files of the target device can thus be restricted.
However, write protection and access protection do not serve as protection of expertise of the POUs. Both CODESYS itself, automation platform plug-ins and persons with knowledge of the project file format can view or modify function blocks created with CODESYS.
Knowledge protection:

Knowledge protection of a project is done by encrypting the project file. Either with a project password, the CODESYS Security Key (dongle), or a certificate. We recommend protection by means of the key or the certificate because in this case no secret needs to be shared between authorized users. The desired type of project encryption is enabled in the project settings.

You can attain knowledge protection of a library by providing it as a target-system-independent "protected library" (*. compiled-library, *.compiled-library-v3). The library file no longer contains source code in this format, but only encrypted precompile context. The compiler is still able to interpret these data. Whether access by other CODESYS components or additional plug-ins is possible depends on their functionality and is to be observed in individual cases. Signing can increase protection even more.

Knowledge protection and copy protection of a boot application can be done by means of a runtime system dongle (simple or licensed) or encryption with a certificate. One of these options is enabled in the object properties of the application.

See also
- «4 "User management and password manager" on page 1821
- \(y^{*}\) Chapter 6.4.1.6.4 "Protecting Projects Using a Dongle" on page 1825
- \({ }^{4}\) Chapter 6.4.1.6.3 "Assigning Passwords" on page 1824
- 乡y Chapter 6.4.1.6.6 "Protecting Objects in the Project by Access Rights" on page 1826
- \({ }^{4}\) Chapter 6.4.1.17.3 "Information for Library Developers" on page 2035
- \({ }^{\bullet}\) Chapter 6.4.1.9.18 "Protecting an application" on page 1915

Encryption with In CODESYS, projects and applications can be encrypted with certificates and signed in order certificates to protect them from unauthorized access.

To do this, you can configure specific security settings for each individual user profile. These settings are always used automatically when the user works with CODESYS projects. Therefore, they do not have to be redone for each project. The general configuration of the security features for a user profile is done in the "Security Screen" view of CODESYS. See the individual instructions below.

You can also encrypt a project file or an application for download or online change directly with a certificate:
- User-independent encryption for the current project is configured in the "Security" category of the "Project Settings".
- User-independent encryption of the application is configured in the "Properties" dialog of the application object.

\section*{NOTICE!}

When you encrypt a project, an application, or online code with a certificate, you will always require the certificate with a private key in order to open the object again.

If the CODESYS Security Agent add-on product is installed, then the "Security Screen" view provides an additional tab: "Devices". This allows for the configuration of certificates for the encrypted communication with controllers.

\section*{Certificates, Windows Certificate Store}

All available certificates are located in the Windows Certificate Store ("certmgr") on your computer. There are two types of keys:
- Certificates with private keys
- for file decryption
- for digital signatures
- Certificates with public keys
- for file encryption
- for verifying digital signatures

The local Windows Certificate Store is usually filled with certificates by the IT administrator of the computer. Certificates are either created using special tools or the creation is requested by a trusted certification authority (CA).
If you receive a certificate file that you need to install yourself in the Windows Certificate Store, then double-click the file in the store directory. Depending on the type (certificate with private or public key only), the appropriate import wizard will appear.
See also
- \(\&\) Chapter 6.4.1.19.2 "General Information" on page 2039
- \(\quad\) Chapter 6.4.1.6.8 "Encrypting Projects with Certificates" on page 1829

\section*{User management and password manager}

User accounts with different rights can be managed in CODESYS. For each account you can define the actions with which the user can access a project object.
The user management is configured in the "Project settings" in the category "Users and Groups".
Before the creation of users and groups, please note the following:
- Rights can only be assigned to user groups. Therefore, you must assign each user to a group.
- There is automatically always a group 'Everyone' and by default every user and every other group is initially a member of this group. Thus each user account is automatically equipped with at least the defined standard rights.
You cannot delete the group 'Everyone', you can only rename it, and you cannot remove members from this group.
Caution: by default "Everyone" does not have the right to change the current user, group and rights configuration!
- There is automatically always a group 'Owner' containing a user 'Owner'. From V3.5 only the 'Owner' initially has the right to change the current user, group and rights configuration in a new project! Hence, only 'Owner' can assign this right to another group.
Initially the 'Owner' can log in with user name 'Owner' and an empty password. You can add further users to the group 'Owner' or remove users from it, but at least one member must be retained. Like 'Everyone', you cannot delete the group 'Owner' and it always possesses all access rights. This prevents a project from being rendered unusable by denying all access rights to all groups.
You can rename both the group 'Owner' and the user 'Owner'.
- If the programming system or a project is restarted, no user is initially logged in to the project. However, the user can then log in via a certain user account with user name and password in order to obtain the access rights defined for the account.
- Each project has its own user management! Therefore, in order to obtain certain access rights to a library integrated into the project, for example, the user must explicitly log in to the library project.
Users and groups defined in different projects are not the same, even if they have the same names.
- A user management in a project only makes sense if it is connected with corresponding rights assignment for the access to project and objects. The project rights are generally managed in the dialog box "Rights" of the "User Management". You can also change the access rights to an individual project object on the "Access control" tab of the "Properties" of the object.
- There are standard menu commands under "Project \(\rightarrow\) User Management" for logging into and out of a project as a defined user. A password manager permits the management of the login data on your computer.

From V3.5 only the 'Owner' initially has the right to change the current user, group and rights configuration in a new project! Hence, only 'Owner' can assign this right to another group.

\section*{NOTICE!}

CODESYS stores the user passwords inaccessibly. If you forget a password, the user account becomes unusable. If you forget the 'Owner' password, the entire project may become unusable!

\section*{Password manager}

The password manager enables you to save login data records that you enter during the login procedures for projects. It is accessible via a button in the login dialog box and offers fast access to the login data currently required. This can be helpful, for example, if you are working in parallel on several library projects that are protected by different passwords.
The password manager itself is protected by an individual master password. If you wish to use the password manager for the first time, CODESYS requests you to define this password in the password manager configuration dialog box. CODESYS notes the master password until you terminate the current CODESYS session. You must always input the password when you wish to \(\log\) in to the password manager for the first time during a new session, or after you have changed it.
See also
- «2 Chapter 6.4.1.6.6 "Protecting Objects in the Project by Access Rights" on page 1826
- *s Chapter 6.4.1.6.7 "Logging in via User Account and Password Manager" on page 1827
- \& Chapter 6.4.1.11.4 "Handling of Device User Management" on page 1971

Rights manage- Rights management for access to a project and objects in a project is necessary in order to ment make a user management meaningful.
The rights for a project are generally managed in the "Rights" editor of the "User Management". You can also change the access rights to an individual project object on the "Access control" tab of the "Properties" dialog box of the object.
Before assigning rights, please observe the following:
- In a new project CODESYS always sets all rights for the execution of actions on objects with the default value 'allowed' (standard right). The only exception to this is the right to change the current user, group and rights configuration. Initially only the 'Owner' group has this right.
- If you are member of a group that is permitted to change rights, you can do this at any time for each right when working further on a project. You change a right by switching between 'allowed' and 'forbidden' or by resetting to the default.
See also
- \& Chapter 6.4.1.6.5 "Setting up a user management" on page 1825
- Chapter 6.4.1.6.6 "Protecting Objects in the Project by Access Rights" on page 1826

Filing, saving Provide the project file with the desired protection before saving it in the file system; see above. For a read-only project file you are given various options so that you can still save the file, depending on the type of write protection.
If the project is to be opened later in an older CODESYS version, it makes sense to save the project for precisely this version (file type), since CODESYS will also inform you immediately about possible losses of data in the course of saving it.
If you wish to save library projects, please observe the rules for the creation of libraries. Also consider the possibility of installing a library directly in a library repository.
If you wish to continue to use a project on another computer, it makes sense not only to save the project file, but also to create a project archive from all relevant auxiliary files.
You can make a setting so that a backup copy of this project is created each time the project is saved. In addition you can configure CODESYS so that projects are generally automatically saved at certain time intervals.
If you wish to keep projects in a source control system, observe the corresponding add-ons for CODESYS. For example, the link to SVN is supported.

See also
- \(\quad \Leftrightarrow\) Chapter 6.4.1.21.4.14.17 "Dialog 'Options' - 'Load and Save"' on page 2791
- \({ }^{\text {• }}\) Chapter 6.4.1.4 "Exporting and Transferring Projects" on page 1815
- \(\quad \Delta\) Chapter 6.4.1.6.9 "Saving the Project" on page 1831
- \({ }^{\bullet}\) Chapter 6.4.1.6.10 "Saving/Sending the project archive" on page 1832
- \(\quad\) © Chapter 6.4.1.17.3 "Information for Library Developers" on page 2035
- \({ }^{\text {• }}\), Chapter 6.4.1.6.11 "Linking a project to the source control system" on page 1833

\subsection*{6.4.1.6.2 Setting up write protection}

A project can be protected against inadvertent changes by means of access and write protection. In addition, however, it can also be provided with read protection (know-how protection). You have the following options:

Open the Requirement: No project is opened. project with write protection
1. Select "File \(\rightarrow\) Open Project".
\(\Rightarrow\) The dialog box "Open project" appears.
2. Select the project.
3. Click on the arrow button * next to the "Open" button and select "Open read-only" from the menu.
\(\Rightarrow\) CODESYS opens the project. At the top right in the main window a line appears "'Project file cannot be saved...'". You must now select one of the offered options if you wish to save the project file.
- \& Chapter 6.4.1.3.2 "Opening a V3 Project" on page 1808

Providing projects with the attribute 'Released'
1. Select "Project \(\rightarrow\) Project Information", then the "Summary" tab.
2. Activate the option "Released", confirm with "OK".
3. Save the project, for example with \([C t r l]+[S]\).
4. Open the project again with the command "File \(\rightarrow\) Open Project".
\(\Rightarrow\) CODESYS opens the project. At the top right in the main window a line appears "Project file cannot be saved...". You can now directly remove the status "Released" again via the offered option if you wish to save the project file.
- \({ }^{4}\) Chapter 6.4.1.21.2.24 "Object 'Project Information"' on page 2515

\section*{Providing a} project in the file system with the property 'Read-only'
\(\triangleright \quad\) Provide the project file in its local file system with the property attribute 'Read-only'.
\(\Rightarrow\) If you had already opened the project and you now attempt to save it under the same name, a dialog box appears informing you about the existent write protection. This dialog box provides you with the following options:

You can save the project under another name or another path using the button "Save As...".

You can deliberately save the project under the same name and path and thus overwrite the existing version in the file system using the button "Overwrite".

You can abort the saving procedure using the "Cancel" button, for example to remove the write protection on the disk.

If you re-open the project, a line appears at the top right in the main window 'The project cannot not be saved...'. You must now select one of the offered options if you wish to save the project file.
- «y "General information about write and access protection" on page 1819

\subsection*{6.4.1.6.3 Assigning Passwords}

Requirement: The project is open
1. Click "Project \(\rightarrow\) Project Settings" and then select the "Security" category.
\(\Rightarrow\) The dialog "Project Settings / Security" opens.
2. Select the "Encryption" option.
\(\Rightarrow\) The option fields "Password", "Dongle", and "Certificates" are selectable.
3. Select the option "Password".
\(\Rightarrow\) The input fields for the encryption password appear.
4. Enter the encryption password in the input field "New Password".
5. Enter the encryption password for confirmation in the input field "Confirm new password".
6. Click "OK".
\(\Rightarrow\) CODESYS saves the encryption password for the project. You must enter this password in order to be able to open the project again, even if it is to be loaded as a library reference.

\section*{CAUTION!}

If you no longer know the encryption password, you can no longer open or restore the project!

\section*{See also}
- \({ }^{4}\) Chapter 6.4.1.21.4.12.8 "Dialog 'Project Settings' - 'Security'" on page 2772
- *) Chapter 6.4.1.6 "Protecting and Saving Projects" on page 1819

\subsection*{6.4.1.6.4 Protecting Projects Using a Dongle}

Requirement: The project is opened and you have connected the CODESYS Security Key (dongle) to your computer.
1. Click "Project \(\rightarrow\) Project Settings" and then select the "Security" category.
\(\Rightarrow\) The dialog "Project Settings / Security" opens.
2. Select the "Encryption" option.
\(\Rightarrow\) The option fields "Password", "Dongle", and "Certificates" are selectable.
3. Select the option "Dongle".
\(\Rightarrow\) The dialog with the drop-down list "Registered Dongles" and the buttons "Add", "Remove", "Comment" and "Flash" opens.
4. Click "Add".
\(\Rightarrow\) The "Add Registered Dongle" dialog opens.
5. Select the CODESYS Security Key (dongle) from the "Dongle" drop-down list and optionally enter a comment.
6. Click "OK".
\(\Rightarrow\) The added dongle is listed in the list "Registered Dongles".
7. Click "OK".
\(\Rightarrow\) The dongle is registered for the project. You must connect the dongle to your computer in order to be able to open the project again, even if it is to be loaded as a library reference.

\section*{NOTICE!}

If the CODESYS Security Key registered for the project is lost, you can no longer open the project or restore it.

See also
- \(\quad\) Chapter 6.4.1.21.4.12.8 "Dialog 'Project Settings' - 'Security'" on page 2772
- \(\Rightarrow\) Chapter 6.4.1.6 "Protecting and Saving Projects" on page 1819

\subsection*{6.4.1.6.5 Setting up a user management}

With the help of the integrated user management, user groups with different access rights and authorizations can be defined. Configuration and handling of the user management in Automation Builder and a AC500 V3 is decribed in the application note User Management with V3 - Configuration and Handling.


This concerns a user management for a CODESYS project file. Visualizations and devices can have their own user management.

The following guide describes how you can adapt the user management for the first time in a project. It deals with the definition of a user and a group to which he belongs.
Requirement: the project for which the user management is to be set up is opened. There is no adapted user configuration yet.
1. Select "Project Settings \(\rightarrow\) Users and Groups" and then the "Users" tab. The user Owner is already created by default.
2. Click on "Add".
\(\Rightarrow\) The dialog box "Add User" appears.
3. Enter a login name, for example 'Dev1', and a password. Leave the option "Activated" activated. Click on "OK".
\(\Rightarrow\) On creating a group for the first time, CODESYS now requests you to authenticate yourself to perform this action.
In this case, enter 'Owner' as the "current user". Do not enter a "password", just click on "OK".
The user Dev1 appears in the list and is automatically a member of the group 'Everyone'.
4. Change to the tab "Groups", in order to add the user to a new group.
\(\Rightarrow\) The groups Everyone and Owner have already been created.
5. Click on "Add" in order to open the dialog box "Add Group".
6. Specify at least one name for the new group, for example 'Developers'. Activate the checkbox next to the entry "User 'Dev1'" in the field "Members". Click on "OK".
\(\Rightarrow\) The group "Developers" now appears with has user member 'Dev1'.
7. Switch to the "Users" tab.
\(\Rightarrow\) The user "Dev1" now appears as a member of the groups 'Everyone' and 'Developers'.

> You can take over the user management configuration from another project by using the "Export/lmport" functions in the dialog box "Project Settings", category "Users and Groups".

See also
- \& Chapter 6.4.1.6 "Protecting and Saving Projects" on page 1819
- « "User management and password manager" on page 1821
- \(\quad\) Chapter 6.4.1.21.4.12.3 "Dialog 'Project Settings' - 'Users and Groups'" on page 2768
- \(\Leftrightarrow\) Chapter 6.4.1.11.4 "Handling of Device User Management" on page 1971

\subsection*{6.4.1.6.6 Protecting Objects in the Project by Access Rights}

Protection of individual objects by setting access rights in the "Rights" editor

\section*{1. Select "Project \(\rightarrow\) User Management \(\rightarrow\) Rights"}
\(\Rightarrow\) The window of the "Rights" editor opens. On the left you can see the action categories, on the right the currently existing user groups.
2. Expand the relevant action category and below it the action for which you wish to change a right.
3. Select the goal of the action in the "Actions" window. In the "Rights" window, select the group for which you would like to change the right. Multiple selection is possible.
\(\Rightarrow\) The buttons in the symbol bar are active.
4. Click on the appropriate button in order to change the right of the group for the action on the target object.
\(\Rightarrow\) CODESYS updates the symbol in front of the group according to the new right. The right is immediately effective.

See also
- \(\xi^{\circ}\) Chapter 6.4.1.21.4.7 "Dialog 'Permissions'" on page 2748

Protection of individual objects by setting access rights in the object properties

Here you can configure whether the members of a group have the right to view, edit or remove the object and to add/remove child objects to/from the object.
1. Select the object in the navigator tree.
2. In the context menu, select the command "Properties" and in the dialog box select the category "Access Control".
3. In the table under "Groups, Actions and Permissions", double-click on the symbol of the right that you wish to change.
\(\Rightarrow\) A selection list of the possible rights appears: "Grant", "Deny", "Clear".
4. Select the desired right and click on "Accept" or "OK".
\(\Rightarrow\) The right is immediately effective for the action and group. The symbol changes accordingly.

See also
- ② Chapter 6.4.1.21.4.11.7 "Dialog 'Properties' - 'Access Control"" on page 2757

\subsection*{6.4.1.6.7 Logging in via User Account and Password Manager}

Logging in to a Requirement: A project is open. You wish to log in as a defined user for this project or for a project without using the password manager functions library integrated in it in order to edit one or the other with certain rights. You have the required login data for the respective project or the library.
1. Select "Project \(\rightarrow\) User Management \(\rightarrow\) User Logon".
\(\Rightarrow\) The dialog box "Logon" opens.
2. Select the project file from "Project/Library" and enter the required access data "User name" and "Password".
3. Log in with "OK".
\(\Rightarrow\) If another user is already logged in, this user will automatically be logged out by the new login.

Setting a master Requirement: A project is open. The dialog box "Login" is open for you to log in as a defined password for the password manager
user for a project or for a library integrated in the project. You wish to use the password manager in order to save login data in it.
1. Select "Project \(\rightarrow\) User Management \(\rightarrow\) User Logon".
2. In the dialog box "Logon", click on the button \(\square\).
\(\Rightarrow\) If you are working for the first time with the password manager, the dialog box "Password Manager Configuration" opens.
3. Enter a character string as the future master password. Confirm it in the second line and click on "OK".
\(\Rightarrow\) CODESYS notes the master password until you terminate the current CODESYS session. You must always input this password when you wish to log in to the password manager for the first time during a new session, or after you have changed it.

\section*{NOTICE!}

If you have forgotten your master password, you no longer have any possibility to access the login data already saved! In this case you can only reset the password manager. After that you must start again to save passwords in the manager!

\section*{Saving login data in the password manager}

Requirement: A project is open. You wish to log in as a defined user for this project or for a library integrated in it in order to edit one or the other with certain rights. You have the required login data for the respective project or the library. These login data have not yet been saved in the password manager.
1. Select "Project \(\rightarrow\) User Management \(\rightarrow\) User Logon", in order to open the "Logon" dialog box.
2. Select the project file from "Project/Library".
3. Enter the user name and password for the project or the library.
4. Click on the button \(\square\).
\(\Rightarrow\) If you are working for the first time with the password manager, you will be requested to define a master password. Refer to the above guide 'Setting a master password for the password manager' for this.
When you call the password manager for the first time in this CODESYS session, you will be requested to enter the master password.
5. Enter the master password when requested to do so.
\(\Rightarrow\) The password manager menu appears.
6. Select the option "Save the credentials locally on this computer".
\(\Rightarrow\) The login takes place. The data are saved in the password manager.

Getting the login data from the password manager

Requirement: A project is open. You wish to log in as a defined user for this project or for a library integrated in it in order to edit one or the other with certain rights. The login data required for this are already saved in the password manager.
1. Select "Project \(\rightarrow\) User Management \(\rightarrow\) User Logon" in order to open the "Logon" dialog box.
2. Click on the button \(\square\).
\(\Rightarrow\) If you are working for the first time with the password manager, you will be requested to define a master password. Refer to the above guide 'Setting a master password for the password manager' for this.
When you call the password manager for the first time in this CODESYS session, you will be requested to enter the master password.
3. Enter the master password when requested to do so.
\(\Rightarrow\) The password manager menu appears.
4. Select the appropriate entry "Use the stored credentials for <user name>".
\(\Rightarrow\) The login takes place automatically with the data read from the password manager.

\section*{Opening the password manager, changing the master password}

Requirement: A project is open. You wish to open the password manager in order to view and/or edit the entries or to change the master password. You have already logged in once with the master password.
1. Select "Project \(\rightarrow\) User Management \(\rightarrow\) User Logon", in order to open the "Logon" dialog box.
2. Click on the button \(\square\).

Select "Open the Password Manager".
\(\Rightarrow\) The password manager window opens.
3. Click on "Change Master Password" and make the change.

\section*{Logging out Requirement: A project is open. A user is logged in, which is recognizable by a name entry in from the project the field "Current User" in the status bar.}
\(\triangleright \quad\) Select "Project \(\rightarrow\) User Management \(\rightarrow\) User Logoff". Alternatively, double-click on the field "Current User" in the status bar.
\(\Rightarrow\) If the user is logged in to only one project, he will now be logged out without further interaction. "(nobody)" appears again in the field "Current User" in the status bar
If the user is logged in to several projects, the dialog box "Logoff" opens. There, select the specific project or library project from which the user is to be logged out.

See also
- « "User management and password manager" on page 1821
- \(y^{2}\) Chapter 6.4.1.21.3.5.28 "Command 'User management' - 'Log in User'"' on page 2613
- \& Chapter 6.4.1.6.5 "Setting up a user management" on page 1825

\subsection*{6.4.1.6.8 Encrypting Projects with Certificates}

\section*{Configuring a} certificate for project file encryption in a user profile

When a project is encrypted with a certificate, this certificate is needed for decryption to open the project. You can assign this certificate to specific user profiles. To do this, select the certificate from the Windows Certificate Store on the "User" tab of the "Security Screen".
1. Double-click in the status bar or click "View \(\rightarrow\) Security Screen".
\(\Rightarrow\) The "Security Screen" view opens.
2. In the "User" tab, select the user profile for which the communication will be encrypted. By default, the specified user profile is the one you have used on your computer to sign into Windows. You can also create a new user profile with \(\dagger\).
3. Click the button in the "Project file decryption" area.
\(\Rightarrow\) The "Certificate Selection" dialog opens.
4. Select a certificate with a private key from the list "Available certificates in the local Windows Certificate Store". Certificates with a private key are identified by the 圆 symbol.
5. Click 圏.
6. The certificate is added to the upper part of the dialog.
7. Click "OK" to confirm your selection.
\(\Rightarrow\) The selected certificate is displayed in the "Security Screen" in the "Project file decryption" area.

Encrypting a A project encrypted with a certificate in connection with a user management allows you to project with a restrict access to the project. certificate
1. Click "Project \(\rightarrow\) Project Settings" and then select the "Security" category. \(\Rightarrow\) The "Project Settings / Security" dialog opens.
2. Select the "Encryption" option.
\(\Rightarrow\) The option fields "Password", "Dongle", and "Certificates" are available.
3. Select the "Encryption" option.
\(\Rightarrow\) The certificates available for project encryption are listed in the lower part of the dialog. If no certificate has been specified yet, then click to select a relevant certificate in the "Certificate Selection" dialog. Then return to the "Project Settings" dialog. Now the certificate is specified for encryption. Now the project can only be edited on computers of users who also have the certificate for file decryption.

Deleting a certif- You delete the certificate in the "Security Screen" view, either directly on the "User" tab or in the icate in the user "Certificate Selection" dialog. The deletion will follow in the other dialog.
- Dialog "Security Screen", tab "User", "Digital Signature", or "Project Data Decryption": Select a certificate and click \(\boxtimes\)
- Dialog "Certificate Selection": in the "Security Screen" dialog, click \({ }^{\circ}\) on the "User" tab. In the upper field of the "Certificate Selection" dialog, select the certificate to be deleted and click \(\boxtimes\).

Configuring a certificate for the digital signature in a user profile

To ensure that the project is not only encrypted with a certificate, but also that its authorship and integrity can be verified, you can add a signature to the project:
1. Double-click in the status bar or click "View \(\rightarrow\) Security Screen".
\(\Rightarrow\) The "Security Screen" view opens.
2. In the "User" tab, select the user profile for which the digital signature will be created. By default, the specified user profile is the one you have used on your computer to sign into Windows. You can also create a new user profile with \(\dagger\).
3. Click the \({ }^{3}\) button in the "Digital signature" area.
\(\Rightarrow\) The "Certificate Selection" dialog opens.
4. Select a certificate with a private key from the list "Available certificates in the local Windows Certificate Store". Certificates with a private key are identified by the 圆 symbol.
5. Click 圏.
\(\Rightarrow\) The certificate is added to the upper part of the dialog.
6. Click "OK" to confirm your selection.
\(\Rightarrow\) The selected certificate is displayed in the "Security Screen" in the "Digital signature" area.

See also
- \(\xi^{*}\) "Encryption with certificates" on page 1820
- \(\quad\) Chapter 6.4.1.21.3.4.18 "Command 'Security Screen'" on page 2592
- \(\quad\) \& Chapter 6.4.1.21.4.19 "Dialog 'Certificate Selection'" on page 2811
- \({ }^{\text {® }}\) Chapter 6.4.1.21.4.11.4 "Dialog: Properties: Security" on page 2754
- \(\quad>\) Chapter 6.4.1.21.4.12.8 "Dialog 'Project Settings' - 'Security"' on page 2772
- \(\Leftrightarrow\) Chapter 6.4.1.9.18 "Protecting an application" on page 1915

\subsection*{6.4.1.6.9 Saving the Project}

Saving a project Requirement: The project is open. The project file is not write-protected. under the same name
\(\triangleright \quad\) Select "File \(\rightarrow\) Save".
\(\Rightarrow\) CODESYS saves the project file with the current project name, which appears in the title bar of the main window. If the project has been changed since it was last saved, then the project name is provided with an asterisk. If this is set in the CODESYS options in the category "Load and Save", then a backup copy will also be made.

\section*{Saving a project Requirement: The project is open. under a different name or format}
1. Select "File \(\rightarrow\) Save Project as".
\(\Rightarrow\) The "Save Project" dialog box opens.
2. Select a storage location in the file system and the desired "File Type" (project file or library file) and the desired storage version. If you want to open the project later in an older version, then it makes sense to save for precisely this version, as you will then be informed immediately in the message window about possible data loss.
\(\Rightarrow\) If the project file is not write protected, then CODESYS saves it in the selected path. Otherwise you will be informed how to proceed.
3. If the current project contains add-ons that are not available in the selected memory format, then the "Extend Profile" dialog box opens.
4. Select the add-ons to extend the memory profile in order for the add-on data to be saved.
5. To save the memory profile permanently, click "Save Profile" and specify a name in the "Enter profile name" dialog box.
6. In the "Extend Profile" dialog box, select the "Use saved profile" option and click "Yes". \(\Rightarrow\) CODESYS saves the project with the saved profile.

Saving a readonly project

Requirement: A read-only project is open.
\(\triangleright \quad\) Select "File \(\rightarrow\) Save".

If the write protection was assigned in CODESYS, then it will be displayed by a line in the top right corner of the main window. Depending on the current situation you will be offered one or more of the following actions so that you can still save the project:
- "Save project under a different file name on the disk": Always appears and continues to the "Save File" dialog box, as for the "Save File as" command
- "Exit read-only mode": Appears if the "Open read-only" option is selected when opening the project.
- "Remove read-only attribute from the project on the disk": Appears if the project file was provided with the 'Read-only' property in the local file system at the time of opening.
- "Remove identification 'Released' in the project information": Appears only if this attribute is currently set.
If the write protection was assigned outside of CODESYS in the properties of the project file in the file system, you will be offered the following options when you attempt to save under the same name and path:
- "Save as": You can save under a different name as with the "Save Project as" command.
- "Overwrite": The write protection is removed from the project file and the file is saved under its existing name.
1. Click on the line in the top right corner of the main window that indicates the write protection.
\(\Rightarrow\) The current options with which you can still save the project appear in a selection menu.
2. Select one of the options offered and perform any necessary actions.
3. Click "File \(\rightarrow\) Save" or "File \(\rightarrow\) Save as".
\(\Rightarrow\) The project can be saved.

\section*{Saving of the project automatically; creating a backup copy \\ Requirement: The project is open.}
1. Click "Tools \(\rightarrow\) Options" (category "Load and Save".
\(\Rightarrow\) The "Load and Save" dialog box opens.
2. Activate the "Create backup files" option.
3. Activate the "Automatically save every ... minutes" option and select a time interval.
4. Click "OK" to close the "Options" dialog box.
\(\Rightarrow\) Each time the project is saved, CODESYS also creates a backup copy <project name>.backup.

CODESYS saves the project automatically at the specified time interval to a file <project name>. autosave in the project directory. If you open the project again after the development system was closed irregularly, then this file will be offered to you as an alternative to the file last saved by the user.

See also
- \(\Rightarrow\) Chapter 6.4.1.6 "Protecting and Saving Projects" on page 1819
- \(\quad\) Chapter 6.4.1.21.4.14.17 "Dialog 'Options' - 'Load and Save"' on page 2791

\subsection*{6.4.1.6.10 Saving/Sending the project archive}

You can configure a project archive and then save it in the file system or send it directly in an e-mail.
To send, follow the guide below as far as point 9. There, click on the button "Send" instead of "Save" in order to directly open the standard e-mail program, in which a new mail will automatically be created with the project archive file as an attachment.
Requirement: A project is opened.
1. Select "File \(\rightarrow\) Project Archive \(\rightarrow\) Save/Send Archive".
\(\Rightarrow\) The dialog box "Project Archive" appears.
2. Activate the checkbox next to each object that is to be saved in the archive.


In order to guarantee know-how protection, CODESYS will not automatically add unprotected libraries, not available as "compiled-library", to a project archive. If you explicitly select such a library in the list of additional files, you will get an appropriate warning.
3. If you want to pack further files in the archive, click on "Additional Files". \(\Rightarrow\) The dialog box "Additional Files" opens.
4. Click on "Add".
5. Select the files and click "Open". \(\Rightarrow\) The files are added to the list of additional files.
6. Click on "OK".
7. Click on "Comment".
\(\Rightarrow\) The dialog box "Comment" opens.
8. Enter a comment and click on "OK".
9. Click on the button "Save".
10. Select a storage location and a file name and click on "Save".
\(\Rightarrow\) The project archive is saved in the file directory.

See also
- \(\sum^{\star}\) Chapter 6.4.1.21.3.2.4 "Command 'Save project"' on page 2554
- \(\boldsymbol{y}^{\boldsymbol{y}}\) Chapter 6.4.1.21.3.2.5 "Command 'Save Project as'" on page 2555

\subsection*{6.4.1.6.11 Linking a project to the source control system}

To link your CODESYS projects to a source control system, check the following option:
The Professional Version Control add-on provides the capability of directly linking to an SVN database. You can get the package at the CODESYS Store and install it with the help of the Package Manager.

Refer to the corresponding help when using Professional Version Control.

\subsection*{6.4.1.7 Localizing projects}

You can display your project in different languages when you create and link localization files. The localization files correspond to those of the GNU gettext system. The format of the localization template files is *.pot (Portable Object Template), from which localization files *.po (Portable Object) are generated after translation.

The project can be localized in different languages. However, editing is possible only in the original version.

You configure which categories of text information are localized in the project. Then you export these texts into a translation template. This template is a file in *.pot format (example: project_1.pot). You produce localization files in the format *.po (example: de.po, en.po, or es.po), either automatically with a corresponding external translation tool or manually with a neutral text editor. You can import the *. po files back into CODESYS and use them for localization.
The commands for using project localization are located in the menu "Project
\(\rightarrow\) Project localization".
See also
- \({ }^{4}\) Chapter 6.4.1.21.3.5.16 "Command 'Project Localization' - 'Create Localization Template"" on page 2604

\section*{Generating
localization templates}

Requirement: A project is open.
1. Click "Project \(\rightarrow\) Project Localization \(\rightarrow\) Create Localization Template".
\(\Rightarrow\) The "Create Localization Template" dialog box opens.
2. Activate the categories of text information that should be included in the localization template.
3. "Position information" can also be included in the template. For each text to be translated, specify its location in the project. Select the positions to be displayed in the translation template: only the first position found, all positions found, or none.
4. Click the "Generate" button.
\(\Rightarrow\) The dialog box opens for saving a * pot file to the file system. Save the localization template. Then you can process the file in a translation tool and generate localization files <language>. po in the required languages.

Format of the In the first line, the text categories are specified that were selected for the translation when localization tem- generating the template:
plate: file *.pot
Example: \#: Content:Comments|Identifiers|Names|Strings: All four categories were selected.

Then each text to be translated is segmented in the form as in the following example:

\section*{Example}
```

\#: D:\Projects\p1.project\Project_Settings:1
msgid "Project Settings"
msgstr ""

```

Line 1: Position information displayed as source code reference. Displayed only if this has been configured when generating the translation file.
Line 2: Untranslated text as entry msgid (example: msgid "Project settings").
Line 3: Placeholder for the translation: msgstr "". Between the single straight quotation marks, the translation in the *. po file must be inserted in the respective language.

\footnotetext{
Format of the You can generate \(\mathrm{a}^{*}\). po file with a translation tool or create one using a neutral text editor localization file: based on the *. pot file. For this purpose, you could change the file extension from *.pot to **. po and edit the according to *. po standard format.
<language>.po
It is imperative to specify the language in the form of the usual culture abbreviation in the metadata of the file (example: "Language: de" for German. Then you insert the translations of the individual texts between the straight quotation marks for the msgstr "" entries.
}

\section*{Example}
```

"Language: de\n"
\#: Content:Names
\#: D:\projects\p1.project\Project_Settings:1
msgid "Project Settings"
msgstr "Projekteinstellungen"

```

Importing locali- Requirement: For your project, localization files (<language>.po) were generated based on zation files / the translation template \({ }^{*}\). pot. The project is open.
1. Click "Project \(\rightarrow\) Project Localization \(\rightarrow\) Manage Localizations".
2. Click on the "Add" button.
\(\Rightarrow\) The "Open Localization File" dialog box appears for selecting a*.po file from the file system.
3. Select one of the localization files (example: <project name>-de.po).
\(\Rightarrow\) The dialog box closes and the affected texts appear in the project in the respective language. For example, if you specify the translation msgstr "Main program" for the POU name "PLC_PRG" in the English localization file, then the object name "Main program" appears in the device tree.
4. In the same way, you import the localization files for other language targets.

Switching locali- Requirement: All required language are stored in the project by importing the corresponding zation, adding and removing localization files
*. po file. The project is open.
1. Click "Project \(\rightarrow\) Project Localization \(\rightarrow\) Manage Localization".
\(\Rightarrow\) The "Manage Localization" dialog box opens. All stored localization files *<language>. po appear in "Files", as well as the entry "<original version>".
2. Select the desired language and click the "Switch Localization" button.
\(\Rightarrow\) The project appears in the selected language. When you select "<original version>", the project is displayed in the original, unlocalized version and it cannot be edited.

Optional: \(\quad\) Select one of the available localizations and activate the "Default Localization" option. Defining a default localization, toggling localizations
\(\Rightarrow\) Click "Project \(\rightarrow\) Project Localization \(\rightarrow\) Toggle Localization" to toggle the localization between the default localization and original version. By default, this command is also available with the

\section*{® Chapter 6.4.1.21.3.5.17 "Command 'Project Localization' - 'Manage Localizations'" on page 2605 \\ Chapter 6.4.1.21.3.5.18 "Command 'Project Localization' - 'Toggle Localization'" on page 2606}

\subsection*{6.4.1.8 Configuring I/O Links}

\subsection*{6.4.1.8.1 General}

With the help of device objects you can map hardware to be controlled in a tree structure in your CODESYS project. This makes the linking of hardware and application easy to handle.

In the configuration editors of the device objects, you can configure the settings for the communication between CODESYS and the controller, and above all for I/O mapping. The I/O mapping is the linking of the inputs and outputs of the controller with the variables of your application.

Access to control objects at runtime can be controlled, depending on the device, via an 'online user management', which you can edit - likewise depending on the device - in the CODESYS Development System. Moreover, communication with the controller depends on the current security settings.

See also
- \(\sum^{4}\) Chapter 6.4.1.8.2 "Configuring Devices and I/O Mapping" on page 1836
- \(y^{\wedge}\) Chapter 6.4.1.11.3 "Encrypting Communication, Changing Security Settings" on page 1967
- \(\Rightarrow\) Chapter 6.4.1.11.4 "Handling of Device User Management" on page 1971

\subsection*{6.4.1.8.2 Configuring Devices and I/O Mapping}

Configuring You can configure the device objects inserted into the device tree in the associated device devices editor. The possibilities depend on the device description. The 'generic device editor' provides tabs that are supplemented as necessary by device-specific tabs.
Requirement: You have opened a standard project in whose device tree a standard PLC and below that a fieldbus device object are inserted.
1. Double-click the device object of the standard PLC in the device tree of your project.
\(\Rightarrow\) The "<device name>" editor opens in the CODESYS main window. The "Communication Settings" tab is in the foreground. Change to the other tabs in order to make configuration settings for the controller. See the help pages for the generic device editor.
2. Double-click the fieldbus device object in the device tree of your project.
\(\Rightarrow\) The "<fieldbus device name>" editor opens in the CODESYS main window. Specific tabs are available depending on the device. For the configuration options, see the help pages for the respective device editor. If the "Show generic device configuration views" option is selected in "Tools \(\rightarrow\) Options", in the "Device Editor" category, then see also the tabs contributed by the generic device editor.

See also
- \({ }^{4}\) ) Chapter 6.4.1.21.2.8.2 "Generic device editor" on page 2427

General infor- Whether or not you can configure an I/O mapping to project variables or even to the entire mation about I/O function blocks depends on the type of device. Configuring an I/O map means linking input and mapping output channels of the device with variables of the project. We also use the term 'mapping' for this.

Pay attention in general to the following for the mapping of inputs and outputs of a device to variables in CODESYS:
- You do not have write access to variables that are mapped to an input.
- You can map an existing variable to one input only.
- You can directly generate new global implicit variables in the I/O map and map them to a device channel.
- The memory layout of structures is specified by the device.
- You can change addresses and fix values in the I/O map.
- For each variable that is assigned to an I/O channel in the "I/O Mapping" dialog, you can cause 'force variables' to be generated during the compilation of the application (see further below). Using these variables you can, for example during the commissioning of a plant, force a value on the input or output via a visualization/HMI.
- Changes in the I/O map can be transferred to the controller with an online change.
- If a pointer to a device input is used, the access is considered to be a write access, for example pTest :=ADR (input); . This leads to a compiler warning when the code is generated: ". . .invalid assignment target". If you require a construct of this kind, you have to first copy the input value input to a variable with write access.
- An I/O address can also be linked with a variable via the 'AT declaration' in the IEC code. Since a device configuration often changes again, however, we recommend that you make the assignments only in the device editor.
If you use the AT declaration, note the following:
- An AT declaration is permissible only with local or global variables, not with input or output variables of function blocks.
- Implicit 'force variables' for I/Os (see below) cannot be generated for AT declarations.
- If you use an AT declaration with structure variables or function block variables, all instances will access the same memory location. This then corresponds to the use of 'static variables' in classic programming languages such as ' C '.

\section*{NOTICE!}

If a pointer to a device input is used, then the access (for example, pTest := ADR (input) ;) applies as write access. This leads to a compiler warning when the code is generated: ". . .invalid assignment target".

If you require a construct of this kind, you have to first copy the input value (input) to a variable with write access.

> As an alternative, you can assign a variable to an address in the programming code using the AT declaration. In view of possible changes of the device configuration, however, we recommend that you make the assignments only in the device editor.

You can export the I/O mapping configuration of a device to a csv file or import it from such a file.

See also
- \({ }^{*}>\) Chapter 6.4.1.21.2.8.12 "Tab '<device name> I/O Mapping'" on page 2444
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.5.37 "Command 'Export Mappings to CSV'" on page 2616
- \#y "Generating implicit variables for the forcing of I/Os" on page 1843

Linking a device Requirement: A device that supports an I/O mapping configuration in CODESYS is inserted in input with an existing project variable ("mapping") the device tree of your project. On the "I/O Mapping" tab in the device editor you thus get a tabular display of the input and output channels of the device with specification of the addresses and data types.

\footnotetext{
0

\section*{NOTICE!}

\section*{Mapping 'too large' data types}

If a variable of a data type that is larger than a byte is mapped to a byte address, the value of the variable will be truncated to byte size there. For monitoring the variable value in the "//O Mapping" dialog, this means that, in the root element of the address, the value is displayed which the variable currently has in the project. The current individual bit values of the byte are displayed in succession in the bit elements below that, but this may not be sufficient for the entire variable value.
}

\begin{abstract}
If a UNION is represented by I/O channels in the mapping dialog, it depends on the device whether mapping to the root element is also possible.
\end{abstract}
1. In a POU, declare, for example, a variable xBool4 of the type BOOL with which you want to access an input of the target device from the application.
2. To open the device editor, double-click the device object in the device tree, and then the "<device name> //O Mapping" tab.
3. Observe the "Variable" column with the display of the device input channels and device output channels \({ }^{\circ}\), which can still be sorted by organizational nodes , depending on the device. We assume that there is a device input of the type BYTE. It is displayed with its individual bit addresses (bit channels) below the BYTE node.
4. Note: When mapping structured variables, the editor prevents you from entering both the structure variable (example: \%QB0) and individual structure elements (example: \%QB0.1 and QBO . 2). Therefore, if there is a main entry with a subtree of bit channel entries in the mapping table, then the following applies: Then you can specify a variable either into the line of the main entry, or into the lines of the subelements (bit channels), but not into both.
You can now occupy either the entire channel with a variable of a suitable type OR its individual bit-channel addresses with suitable variables of the type BOOL or BIT. First of all, double-click a bit input channel in the "Variables" column.
\(\Rightarrow\) An input field opens.
5. In order to place an existing variable on the channel, you have to enter the desired project variable with the complete path. Press \(\ldots .\). to open the Input Assistant. Select, for example, the variable Application.PLC_PRG.xBool4 declared in PLC_PRG.
\(\Rightarrow\) The variable is inserted. The HMI symbol (*) is displayed in the "Mapping" column. The address is now struck through. That does not mean that the address is no longer available, because values of existing variables are managed at another memory space. But: in order to avoid ambiguities when writing the values, you should nevertheless not occupy the address with a further variable, especially in the case of outputs.
Note: For compiler version V3.5 SP11 and higher, the initialization value of the variables is used automatically as the default value when mapping to an existing variable. You can edit the "Default value" field only if you map to a new created variable or if no mapping is specified. In older versions, users had to specify explicitly that the default value and initialization value were identical.
\begin{tabular}{|c|c|c|c|c|c|}
\hline \multicolumn{6}{|l|}{Channels} \\
\hline Variable & Mapping & Channel & Address & Type & Ur \\
\hline \multicolumn{6}{|l|}{- \(\square\) Digital Output} \\
\hline - \({ }^{\text {F }}\) & & Digital Output & \%QED & & \\
\hline - 「 & & Digital Output 0 & \%QEO & BYTE & \\
\hline " & & Bit0 & \%QX0.0 & B00L & \\
\hline " Application.PLC_PRG.xEool_4 & 3 & Bit1 & \%Q40.4 & B00L & \\
\hline 5 & & Bit2 & \%QX0. 2 & B00L & \\
\hline " & & Bit3 & \(\%\) \% \(\times 0.3\) & B00L & \\
\hline \% & & Bit4 & \%Qx0.4 & BOOL & \\
\hline \% & & Rit5 & ocnvo 5 & R R & \\
\hline
\end{tabular}

6．Delete the variable assignment again．Click the root of the channel， the BYTE node．Use the Input Assistant again to select the variable Application．PLC＿PRG．byte＿gotodevice．
\(\Rightarrow\) The variable is inserted，all bit addresses of the main channel are struck through and you should not additionally occupy them．
\begin{tabular}{|c|c|c|c|c|c|}
\hline \multicolumn{6}{|l|}{Channels} \\
\hline Variable & Mapping & Channel & Address & Type & U \\
\hline \multicolumn{6}{|l|}{－．．．Digital Output} \\
\hline －\({ }_{-1}\) & & Digital Output & \％QB0 & & \\
\hline －．\({ }^{\text {a }}\) Application．PLC＿PRG．xBool＿4 & 3 & Digital Output 0 & \％QE0 & BYTE & \\
\hline \％ & & Bit0 & \％Q80．0 & B00L & \\
\hline \％ & & Bit1 & \％ & BOOL & \\
\hline \({ }^{6}\) & & Bit2 & \％0x0．？ & B00L & \\
\hline \({ }^{6}\) & & Bit3 & \％\(\% \times 0.3\) & BOOL & \\
\hline \％ & & Bit4 & \％Qx0．4 & B00L & \\
\hline \％ & & Bit5 & \％ & BOOL & \\
\hline 「 & & Bit6 & \％Qx0．6 & B00L & \\
\hline －\({ }^{\text {侕 }}\) & & Bit7 & \％ & BOOL & \\
\hline \(\pm{ }^{\text {「 }}\) & & Digital Output 1 & \％QB1 & BYTE & \\
\hline －\(\square_{\text {－}}\) Digital Input & & & & & \\
\hline
\end{tabular}

\section*{See also}
－\({ }^{\text {y }}\) Chapter 6．4．1．21．2．8．12＂Tab＇＜device name＞I／O Mapping＇＂on page 2444

Mapping a In the following you will map a device output to a global implicit variable，which you recently device input to a create for this purpose directly in the＂I／O Mapping＂dialog． recently created project variable

Requirement：A device that supports an I／O mapping configuration in CODESYS is inserted in the device tree of your project．On the＂I／O Mapping＂tab in the device editor you will thus see a tabular display of the input and output channels of the device with specification of the addresses and data types．
1．To open the device editor，double－click the device object in the device tree，and then the ＂＜device name＞I／O Mapping＂tab．

2．Click in the mapping table on a channel entry in the＂Variable＂column in order to open an input field．
3. Enter a simple name (without '. ') for a new variable (for example, myBool).
\(\Rightarrow\) CODESYS creates the variable as an implicit global variable in the project and assigns it directly to the channel address. Therefore in this case the address does not appear struck through as in the case of mappings to existing variables
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline Variable & Mapping & Channel & Address & Type & Default Value & Unit & D \\
\hline \multicolumn{8}{|l|}{\(\pm\) Di. Digital Output} \\
\hline \multicolumn{8}{|l|}{--. Digital Input} \\
\hline - \({ }^{*}\) & & Digital Input & \%IB0 & & & & \\
\hline - 4 myBool & 6 & Digital Input 0 & \%IB0 & BYTE & & & \\
\hline \({ }^{4}\) & & Bit0 & \%IX0.0 & B00L & FALSE & & \\
\hline \(\cdots\) & & Bit1 & \%IX0.1 & B00L & FALSE & & \\
\hline 4 & & Bit2 & \%IX0.2 & B00L & FALSE & & \\
\hline
\end{tabular}

Linking a device with a function block instance

If supported by the device, you can map entire function blocks to an input or output channel. This allows you to count the frequency of signal changes or scale a channel value for maintenance purposes, for example.
Here you will map a device output channel to a function block. In this example, the block scales the channel output value.

Requirement: A device with a digital output that supports FB mapping is linked in the project. There is a function block "Scale_Output_Int" with the following implementation. The attributes of the function block itself and before the output parameter with which the channel output is processed are important.
```

{attribute 'io_function_block'}
FUNCTION_BLOCK-Scale_Output_Int
VAR_INPUT
iInput : INT;
iNumerator : INT;
iDenominator : INT :=1;
iOffset : INT := 0;
END_VAR
VAR_OUTPUT
{a\overline{t}tribute 'io_function_block_mapping'}
iOutput : INNT;
END_VAR
VAR
END VAR
IF íDenominator <> 0 THEN
iOutput := TO_INT(TO_DINT(iInput) * TO_DINT(iNumerator) /
TO_DINT(iDenomina\overline{tor)) + iOffset;}

```
1. Open the "<device name> I/O Mapping" tab of the device editor. Double-click the output that should be connected to the function block. Click the button \(«\) "Add FB for IO channel".
\(\Rightarrow\) The "Select Function Block" dialog opens. On the left side, you see at least the function block "Scale_Output_int" below the "Application" node. Libraries linked in the project that contain corresponding function blocks are also displayed for selection.
2. Select the POU myScaleOutputInt.
\(\Rightarrow\) After clicking "OK", the path of the function block parameter iOutput in the "Variable" is entered in the mapping dialog. The path comprises the application name, the device channel name, and the selected FB output (example: App1.Out_4_Int_myScale_Output_Int_1.iOutput).
3. Select the channel and click \(\rightarrow\) "Go to Instance".
\(\Rightarrow\) The focus switches to the "<device name> IEC Objects" tab and the created entry for the new IEC object Out_4_Int_myScale_Output_Int_1. In this view in online mode, you see the current vālue of the \(\bar{p}\) arameter ioūput for the channel Out_4_Int scaled by the FB. You can also write and force the value as in other monitoring views.

See also
- \({ }^{\text {• }}\), Chapter 6.4.1.21.4.4 "Dialog 'Select Function Block'" on page 2746
- «y Chapter 6.4.1.21.2.8.12 "Tab '<device name> I/O Mapping'" on page 2444
- \(\quad\) • Chapter 6.4.1.21.2.8.13 "Tab '<device name> IEC Objects'" on page 2449
- \({ }^{\text {n }}\) Chapter 6.4.1.20.6.3.23 "Attribute 'io_function_block', 'io_function_block_mapping'" on page 2293

\section*{Changing and fixing an address value in the I/O map}

You can change the address value of an entire channel (but not that of an individual subelement of the channel!) in the mapping table of the "<device name> //O Mapping" tab. This allows you to adapt the addressing to a specified machine configuration and to retain the address value even if the layout of the modules changes. By default, a change of the layout leads to an automatic adaptation of the address values.

Requirement: Your project has I/O mapping. See the corresponding sections of the help page above.
1. To open the device editor, double-click the device object in the device tree, and then the "<device name> I/O Mapping" tab.
2. Click in the mapping table on a channel entry in the "Address" column in order to open an input field. This is only possible for the 'root' address of a channel, not for a particular one of its subelements.
Therefore, change the top address entry of a channel in the table, for example from QBO to QB1. Exit the input field.
\(\Rightarrow\) The address value is changed. The symbol is displayed before the address. It indicates that the address is fixed. The addresses of the subelements of the channel are also changed accordingly. If you now change the position of the device object inside other device objects with input/output channels in the device tree, CODESYS does not adapt these addresses to the new order as would be the case without fixing.
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline \multicolumn{7}{|l|}{Channels} \\
\hline Variable & Mapping & Channel & Address & Type & Default Value & Unit \\
\hline \multicolumn{7}{|l|}{-- Digital Output} \\
\hline \(\square{ }_{*}\) & & Digital Output & (10) \%QB1 & & & \\
\hline \(={ }^{*}\) & & Digital Output 0 & \%QB1 & BYTE & & \\
\hline - Application.PLC_PRG.byVar 1 & \({ }^{*}\) & Bit0 & \% \(2 \times 1.0\) & BOOL & FALSE & \\
\hline \% Application.PLC_PRG.byVar2 & * & Bit1 & \%Q*1.4 & BOOL & FALSE & \\
\hline \(\mathrm{K}_{6}\) & & Bit2 & anvi? & R01 & EAISE & \\
\hline
\end{tabular}
3. In order to undo the manual change or fixing, open the input field of the address value again, delete the address entry and press the Enter key.
\(\Rightarrow\) CODESYS resets the address and the subsequent addresses concerned to the values they had before the change and removes the symbol 0 .

See also
- " \(\boldsymbol{y}^{\text {a }}\) Chapter 6.4.1.21.2.8.12 "Tab '<device name> I/O Mapping'" on page 2444
- \(\Leftrightarrow\) Chapter 6.4.1.20.4.12 "Addresses" on page 2231

Configuration of Depending on the device that you link in the project, CODESYS updates the variables applied the I/O variable to its inputs and outputs in different ways. You can explicitly change the settings for this in the update "//O Mapping" dialog.

See also
- \(\quad\) ² Chapter 6.4.1.21.2.8.12 "Tab '<device name> I/O Mapping'" on page 2444

> Monitoring of variables in the I/O map in online mode

Requirement: You have compiled an application with a device configuration containing I/O maps without error. The associated hardware and the bus system are running. You have connected to the controller by means of the "Online \(\rightarrow\) Login" command and have loaded and started the application.
1. Open the "//O Mapping" tab of the PLC in the device editor. To open the editor, doubleclick the device object in the device tree.
\(\Rightarrow\) The mapping table now additionally contains the "Current Value" and "New Value" columns.

If a structure variable is mapped to the 'root' element of an address \({ }^{1}\), CODESYS does not display a value in this line in online mode. If, for example, a DWORD variable is mapped to the address, however, the respective values are monitored both in the 'root' line and in the indented bit-channel lines below it.

As a matter of principle, the field in the 'root' line always remains empty if the value would be composed of several subelements.
1 'root' = top element of this address in the Mapping dialog
2. Enter a certain variable value for an entry in the column "New value" and press [F7] to force or [Ctrl]+[F7] to write the value.
\(\Rightarrow\) As in the case of monitoring in the declaration editor or in watch lists, the forced variable value is displayed in the column "Current Value" with a prefixed red F-symbol or the written value.

(1) Forced values on the controller
(2) Values not used on the controller, value shown in gray

Generating implicit variables for the forcing of I/Os

During the commissioning of a plant or machine it may be necessary to 'force' the values applied at the inputs and outputs. If a device supports this you can cause special 'force variables' to be generated for this purpose and use them, for example, in an HMI visualization.
Requirement: The device supports the functionality. You have a project in which an I/O map is configured for the device and which contains a program object PLC_PRG.
1. Open the device editor, "PLC Settings" tab, by double-clicking the device object in the device tree.
2. Activate the option "Generate force variables for IO mapping".
3. Press [F11] to compile the application.
\(\Rightarrow\) Two variables are created for each I/O channel in accordance with the following syntax, in the process of which spaces in the channel name are replaced by underscores:
<device name>_<channel name>_<IEC address>_Force of type BOOL for the activation and deactivation of forces.
<device name>_<channel name>_<IEC address>_Value of the data type of the channel for defining the value that you want to force on the channel.

These variables are available in the Input Assistant in the category "Variables"/
"IoConfig_Globals_Force_Variables." You can use them in CODESYS in programming objects, in visualizations, in the symbol configuration, etc.
4. Open the function block "PLC_PRG", set the focus in the implementation part and press F2.
\(\Rightarrow\) The Input Assistant opens. The variables are available in the category "Variables"/ "loConfig_Globals_Force_Variables" as described above.
A rising edge at the 'Force variable' input activates the forcing of the respective input or output with the value given by the 'Value variable'. A falling edge deactivates the forcing. Deactivation by resetting the 'Force' variable to FALSE is the requirement for being able to force a new value.

Take note of the following restrictions.
- Forcing via the implicit force variables is only possible for channels that are mapped in the "I/O Mapping" of the device to an existing or recently created variable.
- Forcing via the implicit force variables is not possible for unused inputs and outputs or those that are mapped to a variable via an AT declaration in an application program.
- I/O channels that you want to force via the mechanism have to be used by CODESYS in at least one task.
- CODESYS identifies forced inputs in the monitoring by the red Force symbol, but not forced input/outputs. The forced value is used only implicitly by the I/O driver for writing to the device.

See also
- \({ }^{4}\) Chapter 6.4.1.21.2.8.10 "Tab 'PLC Settings'" on page 2439
- \({ }^{*}\) Chapter 6.4.1.12.5 "Forcing and Writing of Variables" on page 1987 one dialog for multiple devices

I/O mapping in There is a table that displays the I/O map of a device plus the I/O maps of all subelements inserted below it in the device tree. There you can edit the I/O maps in exactly the same way as in the individual mapping tables of the respective device editors.

Requirement: In the device tree of your project there are several PLCs inserted that each enable an I/O mapping configuration.
1. Select the root node of the device tree and click "Edit I/O Mapping" in the context menu .
\(\Rightarrow\) The "Edit I/O Mapping" dialog opens, in which the I/O mapping configurations of all devices inserted in the project are displayed in a table. You can edit the entries in the same way as in the "//O Mapping" dialog of the associated device editor.
2. Now select one of the control objects in the device tree and select the "Edit I/O Mapping" command once again in the context menu.
\(\Rightarrow\) The "Edit I/O Mapping" dialog now shows only the I/O table for the I/O mapping configurations found in and under the selected object.
3. Set a desired "Filter" in the bar above the table or enter a variable name in the "Search for variable" field in order to see the use of this variable in the mapping.
\(\Rightarrow\) The method of working in this window is the same as that described for the "<device name> I/O Mapping" tab.

See also
- ⓨ Chapter 6.4.1.21.3.5.35 "Command 'Edit I/O Mapping'" on page 2615

\subsection*{6.4.1.9 Programming of Applications}

\subsection*{6.4.1.9.1 Introduction}

To create an application program which can be run on the controller, you fill POUs with declarations and implementation code (source code), establish the link from the controller I/Os to application variables, and configure the task assignment. After checking and debugging, the CODESYS compiler creates the application code which can be downloaded to the controller.

The programming of the application POUs is supported by the programming language editors and other features such as text lists, image pools, alarm configurations, pragmas, refactoring, and ready-to-use POUs from CODESYS Development System or libraries.
There are features for syntax checking and code analysis, for achieving data persistence, and for encrypting the application code which is downloaded to the controller.

\subsection*{6.4.1.9.2 Designating identifiers}

Identifiers are names of variables and programming objects (for example programs, function blocks, and methods) and names of other objects of the application and project. There are rules that you must follow when assigning identifiers. Furthermore, there are also recommendations to help you designate uniform and expressive identifiers.

You designate variables identifiers in the variables declaration. These identifiers can e changed in the declaration section of the programming object. You designate identifiers for programming objects and other objects in the dialog box when adding the object. You can change the identifier of an existing object of the application or of the project in the properties dialog of the object. However, you cannot change the identifiers of objects that can be available only one time per application or project (for example, the "Library Manager" and "ImagePool" identifiers).

See also
- \(\boldsymbol{y}^{2}\) Chapter 6.4.1.20.7 "Identifiers" on page 2327

\subsection*{6.4.1.9.3 UTF-8 Encoding}

Basics
Applications in CODESYS can process a wide variety of characters, for example, to output an error message in various languages. Or to display visualizations in a language selected by the user which accepts user input in a wide variety of languages, characters, or symbols. If a comprehensive character set is not necessary, or if a project should not be changed, then strings which are encoded Latin-1 format can still be used.

Table 382: Character set tables
\begin{tabular}{|c|c|c|c|}
\hline Character set & Code page number & Description & Character encoding \\
\hline ASCII & 20127 & \begin{tabular}{l}
- 128 characters \\
- Suitable for english texts
\end{tabular} & 7-bit encoded character \\
\hline DOS-Latin-1 & 819, 850 & \begin{tabular}{l}
- Complies with ISO 8859 \\
- Suitable for western european languages in the Windows command line window.
\end{tabular} & 8-bit encoded character \\
\hline Latin-1 & 28591 & \begin{tabular}{l}
- Complies with ISO-8859-1 \\
- Often used for HTML pages with äöüß but without \(€\) or for example without special french characters.
\end{tabular} & 8-bit encoded character \\
\hline Windows 1252 Encoding & 1252 & \begin{tabular}{l}
- Default Windows character set for western european countries \\
- Windows uses the UTF-16 format internally \\
- Contains all characters from ISO 8859-1 and ISO 8859-15, but partly with different encoding
\end{tabular} & 8-bit encoded character \\
\hline Unicode & & \begin{tabular}{l}
- Universal character set for all possible languages, including historical languages, Braille, music, or emojis. \\
- More than 100,000 characters can be displayed. \\
- Each character has a numeric code. \\
- In contrast to ASCII, a separation is made between the assignment of code points to characters and the encoding of the characters. \\
- Numeric code < 128 are ASCII compatible \\
- Numeric codes < 256 are ISO 8859-1 compatible \\
For more information, see unicode.org
\end{tabular} & \\
\hline Unicode 14.0 & & 144,697 characters & \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|}
\hline Character set & Code page number & Description & Character encoding \\
\hline UTF－16 & 1200 & \begin{tabular}{l}
－Special Unicode \\
－Used in some operating systems（Windows，OS X） and programming languages （Java，．NET）for internal character representation． \\
－It should be noted that different computer architec－ tures encode the 4－byte characters differently． Little endian byte order for UTF－16LE
\end{tabular} & \begin{tabular}{l}
16－bit encoded char－ acters \\
The characters are encoded either in 2 bytes or 4 bytes．
\end{tabular} \\
\hline UTF－8 & 65001 & \begin{tabular}{l}
－Byte－oriented encoding format of Unicode charac－ ters． \\
－Most widespread \\
－Used in GNU／Linux and Unix operating systems，and in various Internet services （email，web，browser）． \\
－Compatible with ASCII char－ acters in the first 128 charac－ ters（0－127）．
\end{tabular} & \begin{tabular}{l}
Tuple of 8－bit words per character \\
The characters are encoded in different length from 1 to 4 bytes．
\end{tabular} \\
\hline
\end{tabular}

\section*{UTF－8 in Codesys}
UTF－8 encoding is the encoding with the most comprehensive character set． Therefore，it is recommended that you enable UTF－8 encoding for new projects as well as for existing projects to be used in a new context．

Table 383：Project－wide encoding in CODESYS
\begin{tabular}{|l|l|l|}
\hline Data type & \begin{tabular}{l} 
Compile option：UTF8 \\
Encoding for STRING
\end{tabular} & Which encoding is used project－wide？ \\
\hline \multirow{3}{*}{ STRING } & Enabled & UTF－8 \\
\cline { 2 - 3 } & Disabled & \begin{tabular}{l} 
Windows 1252 encoding（default Windows \\
encoding） \\
Latin－1
\end{tabular} \\
\hline \multirow{2}{*}{ WSTRING } & Enabled & UTF－16 \\
\cline { 2 - 3 } & Disabled & UTF－16 \\
\hline
\end{tabular}

In CODESYS，the＂STRING＂data type can be encoded in Latin－1 or UTF－8 formats．
The＂WSTRING＂data type always encodes its characters as Unicode in UTF－16．

\section*{Encoding a single string literal in UTF－8 format}

Even if the project－wide encoding format is set to Latin－1，you can encode a single literal in UTF－8 format．To do this，add the＂UTF8\＃＂type prefix to the literal．
```

{attribute 'monitoring_encoding' := 'UTF-8'}
strVarUtf8: STRING := UTF8\#'你好,世界!ÜüÄäÖÖ'';

```

For more information，see：
Constant：UTF8\＃String；« Chapter 6．4．1．20．4．6＂Constant：UTF8\＃String＂on page 2222

Pragma Attribute: \& monitoring_encoding
String conversion for UTF-8 encoding
If you have enabled UTF-8 encoding project-wide, then you can use the string conversion functions as usual.

\section*{String manipulation}

Use library functions to manipulate your strings.
If "STRING" variables should be manipulated, then an index access to a variable in ASCII format often leads to the desired result. It is better not to use this construct. It is not just a bad programming style. To make matters worse, with UTF-8 encoding, index access leads to unwanted string manipulation.

\section*{UTF-8 encoding only for project-wide configuration}

A UTF-8 encoding is used if the project-wide compile option UTF8 encoding for STRING is enabled. Library functions and add-ons are then also oriented according to this setting.

If you use single UTF-8 encoded strings, then you have to make sure that they are interpreted correctly wherever they are used. For example, a string variable in the OPC server will be converted to UTF-8 before being transferred to a client if the setting is not selected. Values such as "UTF8\#'äöü'" would then be misinterpreted. Similar problems can arise when outputting strings in the visualization.

\subsection*{6.4.1.9.4 Declaration of Variables}

\section*{General}
```

Variable decla- You can declare variables at the following locations:
ration: Where
and how?

- Declaration part of a POU
The "Declare Variable" dialog helps you with this.
Hint: If you define a variable in the tabular declaration editor, the correct syntax is automatically produced.
- Declaration part of the GVL or NVL editor
- I/O mapping configuration of an I/O device object

```

\section*{Syntax}
```

( <pragma> )*
<scope> ( <type qualifier> )?
<identifier> (AT <address> ) ? : <data type> ( := <initial
value> ) ? ;
END_VAR

```
\begin{tabular}{|c|c|c|}
\hline & Declaration & \begin{tabular}{l}
See also \\
- ※ Chapter 6.4.1.9.4.2 "Using the declaration editor" on page 1851 \\
- \({ }^{\circ}\) ) Chapter 6.4.1.9.4.3 "Using the 'Declare variable' dialog box" on page 1852 \\
 "Command 'Auto Declare"' on page 2572
\end{tabular} \\
\hline <pragma> & \begin{tabular}{l}
Pragma (none, one, or multiple) \\
Note: By adding a pragma, you can affect the behavior and the properties of one or more variables.
\end{tabular} & \begin{tabular}{l}
See also \\
- « Chapter 6.4.1.9.8 "Using Pragmas" on page 1888 \\
- \(\quad\) H Chapter 6.4.1.20.6 "Pragmas" on page 2269
\end{tabular} \\
\hline <scope> & \begin{tabular}{l}
Scope \\
- VAR \\
- VAR_CONFIG \\
Note: If variables with incomplete address information are declared in function blocks (for example, AT \%I*), then the variables in the variable declaration VAR_CONFIG have to be completely declared. You can access these variables in a local instance only when this is done. \\
- VAR_EXTERNAL \\
- VAR_GLOBAL \\
- VAR_INPUT \\
- VAR_INST \\
- VAR_IN_OUT \\
- VAR_OUTPUT \\
- VAR_STAT \\
- VAR_TEMP
\end{tabular} & \begin{tabular}{l}
See also \\
- ③ Chapter 6.4.1.20.2.2 "Local variables - VAR" on page 2113 \\
 \\
"Configuration variables - \\
VAR_CONFIG" on page 2121 \\
- \(\quad \stackrel{y}{*}\) Chapter 6.4.1.20.2.9 \\
"External variables \\
- VAR_EXTERNAL" \\
on page 2120 \\
- \(\sum_{\text {a }}\) Chapter 6.4.1.20.2.6 "Global variables - VAR_GLOBAL" on page 2118 \\
* Chapter 6.4.1.21.2.12 \\
"Object 'GVL' - Global Variable List" on page 2465 \\
- \& Chapter 6.4.1.20.2.3 "Input variables - VAR_INPUT" on page 2113 \\
- \(\quad\) Chapter 6.4.1.20.2.10 "Instance variables VAR_INST" on page 2120 \\
- \(\Leftrightarrow\) Chapter 6.4.1.20.2.5 "Input/Output Variable (VAR_IN_OUT)" on page 2114 \\
- \& Chapter 6.4.1.20.2.4 "Output variables - VAR_OUTPUT" on page 2114 \\
- \& \({ }^{\circ}\) Chapter 6.4.1.20.2.8 \\
"Static variables - VAR_STAT" on page 2119 \\
- \# Chapter 6.4.1.20.2.7 "Temporary variable - VAR_TEMP" on page 2119
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|}
\hline <type qualifier & \begin{tabular}{l}
Type qualifier \\
- CONST \\
- Retain \\
- Persistent
\end{tabular} & \begin{tabular}{l}
See also \\
- ② Chapter 6.4.1.20.2.12 "Constant Variables - 'CONSTANT'" on page 2121 \\
- ⓢ Chapter 6.4.1.20.2.14 "Retain Variable - RETAIN" on page 2124 \\
- ② Chapter 6.4.1.20.2.13 "Persistent Variable - PERSISTENT" on page 2122
\end{tabular} \\
\hline <identifi er> & \begin{tabular}{l}
Identifier, variable name \\
Note: The rules listed in the chapter "Identifiers" must be followed without exception when assigning an identifier. In addition, you will find recommendations for uniform naming.
\end{tabular} & \begin{tabular}{l}
See also \\
- \(\Leftrightarrow\) Chapter 6.4.1.20.7 "Identifiers" on page 2327
\end{tabular} \\
\hline \begin{tabular}{l}
AT \\
<address>
\end{tabular} & \begin{tabular}{l}
Assignment of an address in the input, output, or flag memory range (I, Q, or M) \\
AT \% <memory area prefix> ( <size prefix> )? <memory position> \\
Example \\
- AT \%I* // Incomplete address \\
- AT \%I7. 5 \\
- AT \%IWO \\
- At \(\%\) QX7. 5 \\
- AT \%MD48
\end{tabular} & \begin{tabular}{l}
See also \\
- \(\Rightarrow\) Chapter 6.4.1.8.2 "Configuring Devices and I/O Mapping" on page 1836 \\
 declaration" on page 1902 \\
- \(\quad 4\) Chapter 6.4.1.20.4.12 "Addresses" on page 2231
\end{tabular} \\
\hline <data type> & \begin{tabular}{l}
Data type \\
- <elementary data type> \\
- <user defined data type> \\
- <function block>
\end{tabular} & \begin{tabular}{l}
See also \\
- ⓢ Chapter 6.4.1.20.5 "Data \\
Types" on page 2234 \\
- \(\Leftrightarrow\) Chapter 6.4.1.21.2.9 "Object 'DUT'' on page 2461 \\
* \({ }^{*}\) Chapter 6.4.1.21.2.21.3 "Object 'Function Block'" on page 2479
\end{tabular} \\
\hline <initial value> & \begin{tabular}{l}
Initial value \\
<literal value> | <identifier> | <expression>
\end{tabular} & \begin{tabular}{l}
See also \\
- \(\boldsymbol{y}^{2}\) Chapter 6.4.1.20.7 "Identifiers" on page 2327 \\
- «y "Constants and literals" on page 2219 \\
- \({ }^{4}\) ) Chapter 6.4.1.20.1.4.3 "ST expressions" on page 2050
\end{tabular} \\
\hline ( ... ) ? & Optional & \\
\hline ( ... ) * & Optional repetition & \\
\hline
\end{tabular}
```

Example
GVL
{attribute 'qualified_only'}
{attribute 'linkalways'}
VAR_GLOBAL CONSTANT
g_CiMAX_A : INT := 100;
g_ciSPEC\overline{CAL : INT := g_ciMAX_A - 10;}
END_VAR
GVL_CONFIG {attribute 'qualified_only'}
VAR_CONFIG
// Generated instance path of variable at incomplete address
PLC_PRG.fbDoItNow.XLOCINPUT AT %I*: BOOL := TRUE;
END_VAR
FB_DoIt (FB) METHOD METH_Last : INT
VAR_INPUT
i\overline{V}ar : INT;
END_VAR
VAR INST
iL̄ast : INT := 0;
END_VAR
METH Last := iLast;
iLast := iVar;
FUNCTION_BLOCK FB_DoIt
VAR_INPUT
wInput AT %IWO : WORD; (* Input variable *)
END VAR
VAR_OUTPUT
wOutput AT %QWO : WORD; (* Output variable *)
END_VAR
VAR_IN_OUT
aData_A : ARRAY[0..1] OF DATA_A; // Formal variable
END VAR
VAR_EXTERNAL
GVL.g_ciMAX_A : INT; // Declared in object GVL
END VAR
VAR_STAT
iNumberFBCalls : INT;
END VAR
VAR
iCounter: INT;
xLocInput AT %I* : BOOL := TRUE; // VAR_CONFIG
END_VAR
iNumberFBCalls := iNumberFBCalls + 1;
PLC_PRG PROGRAM PLC_PRG
(PR\overline{G}) VAR
iLoop: INT;
iTest: INT;
fbDoItNow : FB_DoIt;
iTest_200: INT;
aData_Now : ARRAY[0..1] OF DATA_A := [(iA_1 := 1, iA_2 := 10,
dwA_3 := \overline{1}6\#00FF),(iA_1 := 2, iA_2 := = 20, dwA_3 := 16\#FF\overline{0}0)];
END_VAR
iTest := GVL.g_ciMAX_A;
iTest_200 := \overline{2 * GVL}.g_ciMAX_A;
fbDoItNow(aData_A := aDāta_Now);
FOR iLoop := 0 TO GVL.g_ci\overline{SPECIAL DO}
;
END_FOR

```

Variable initiali- The standard initialization value for all declarations is 0 . In the declaration part you can also zation specify user-defined initialization values for each variable and each data type.

The user-defined initialization starts with the assignment operator : = and consists of any valid expression of the programming language ST (structured text). You thus define the initialization value with the help of constants, other variables or functions. If you use a variable, you must also initialize it.

Examples
```

VAR
varl:INT := 12;
value 12
value defined by an expression of constants
y : INT := x + fun(4);
// initialization
value defined by an expression,
function call; notice the order!
z : POINTER TO INT := ADR(y); // not described in
the standard IEC61131-3:
value defined by an adress function;
case the pointer will not be initialized
Change *)
END_VAR

```

\section*{Notes on the order of initialization}

From compiler version 3.5.3.40, variables in a function block are initialized in the following order: firstly, all constants in accordance with the order of their declarations, then all other variables in accordance with the order of their declarations.

\section*{NOTICE!}

From compiler version 3.3.2.0, variables from global variable lists are always initialized before the local variables of a POU.

See also
- «y Chapter 6.4.1.20.5.15 "Data Type 'ARRAY"" on page 2247
- « "Declaration and initialization of structure variables" on page 2262
- \(\Leftrightarrow\) Chapter 6.4.1.20.5 "Data Types" on page 2234
- « Chapter 6.4.1.20.6.3.16 "Attribute 'global_init_slot'" on page 2285

\section*{Using the declaration editor}

The declaration editor is used for declaring variables in the variable lists and POUs.
The declaration editor offers two possible views: textual 国 and tabular \(\square\).
in the dialog in "Tools \(\rightarrow\) Options \(\rightarrow\) Declaration Editor", you define whether only the textual view or only the tabular view is available, or whether you can switch between both views by means of the buttons on the right side of the editor view.
If the declaration editor is used in conjunction with a programming language editor, it appears as the declaration part at the top of the window of a POU.

Declaring in the textual declaration editor

The behavior and the appearance of the textual editor are configured with the settings in the dialog "Tools \(\rightarrow\) Options \(\rightarrow\) Text Editor". The settings concern colors, line numbers, tab widths, indentations etc. The usual Windows functions are available, plus the IntelliMouse functions if necessary.
Requirement: You have opened a programming object (POU, GVL or NVL) of a project. The textual declaration editor has the focus.
\(\triangleright \quad\) Enter the variable declarations in correct syntax. With [F2] you can open the dialog "Input Assistant" for the selection of the data type or a keyword.

\section*{Declaring in the tabular declaration editor}

In the tabular declaration editor, you add variable declarations to a table with the following columns: "Scope", "Name", "Address", "Data type", "Initialization", "Comment", and "Attributes" (pragmas).

Requirement: A programming object (POU or GVL) of a project is open. The tabular declaration editor has the focus.
1. Click the button in the declaration header or select the command "Insert" in the context menu.
\(\Rightarrow\) CODESYS inserts a new row for a variable declaration and the input field for the variable name opens.
2. Specify a valid variable identifier.
3. Open the other fields of the declaration line as required with a double-click and select the desired specifications from the drop-down lists or with the help of the dialogs which appear.

See also
- \({ }^{4}\) Chapter 6.4.1.20.1.2 "Declaration Editor" on page 2047
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.3.32 "Command 'Auto Declare"' on page 2572
- H Chapter 6.4.1.9.4.3 "Using the 'Declare variable' dialog box" on page 1852
- \(y^{2}\) Chapter 6.4.1.21.3.17.2 "Command 'Edit Declaration Header'" on page 2717
- \#y "Dialog 'Input Assistant' - Tab 'Categories'" on page 2575

\section*{Using the 'Declare variable' dialog box}

Requirement: A programming object (POU or GVL) of a project is open.
1. Select the command "Edit \(\rightarrow\) Auto Declare".
\(\Rightarrow\) The dialog box "Auto Declare" opens.
2. Select the desired scope for the variable from the selection list "Scope".
3. Enter a variable name in the input field "Name".
4. Select the desired data type from the selection list "Type".
5. If the initialization value deviates from the standard initialization value, enter an initialization value for the variable.
6. Complete your entries with a click on "OK".
\(\Rightarrow\) CODESYS lists the newly declared variable in the declaration part of your programming object.

With the help of pragmas in the declaration part you can affect the processing of the declaration by the compiler.

See also
- « Chapter 6.4.1.21.3.3.32 "Command 'Auto Declare"" on page 2572
- « Chapter 6.4.1.20.5 "Data Types" on page 2234
- \(\Leftrightarrow\) Chapter 6.4.1.9.8 "Using Pragmas" on page 1888

\section*{Declaring arrays}

Requirement: A programming object (POU or GVL) of a project is open.
1. Click "Edit \(\rightarrow\) Declare Variable".
\(\Rightarrow\) The "Declare Variable" dialog opens.
2. Select the desired scope for the array from the drop-down list "Scope".
3. Enter an identifier for the array in the "Name" input field.
4. Click the arrow button ( \(\triangle\) ) next to the "Data type" input field and select the "Array Assistant" entry from the selection menu.
5. In the input fields "Dimension 1", type in the lower and upper limit of the first dimension of the array (example: 1 and 3).
\(\Rightarrow\) The field "Result" displays the 1st dimension of the array (example: ARRAY [1..3] OF ?).
6. In the input field "Basic type", type in the data type of the array or use the "Input Assistant" ( \(\triangle\) ) or the "Array Assistant" (example: DINT).
\(\Rightarrow\) The field "Result" displays the data type of the array now (example: ARRAY [1..3] OF DINT).
7. Define the second and third dimensions of the array according to steps 5 and 6 (example: Dimension 2: 1 and 4, Dimension 3: 1 and 2).
\(\Rightarrow\) The "Result" field displays the array with the defined dimensions: ARRAY [1..3, 1..4, 1..2] OF DINT. The array consists of 3 * 4 * \(2=24\) elements.

In an array of variable length, declare the dimension limits with an asterisk placeholder (*). Arrays of variable length are permitted to be used only in VAR_IN_OUT declarations of function blocks, methods, or functions.

Example of a 2-dimensional array of variable length:
aiUnknownLengthData : ARRAY [*,*] OF INT;
8. Click "OK".
\(\Rightarrow\) In the dialog "Declare Variable" the field "Data type" displays the array.
9. To modify the initialization values of the array, click the arrow button ( \(\ldots\) ) next to the "Initialization value" input field.
\(\Rightarrow\) The "Initialization Value" dialog opens.
10. Select the line of the array element whose initialization value you wish to modify. Example: Select array component [1, 1, 1].
11. Enter the desired initialization value in the input field below the list and click button "Use value on selected lines" (example: value 4).
\(\Rightarrow\) CODESYS displays the changed initialization value of the selected line.
12. Click "OK".
\(\Rightarrow\) In the "Initialization value" field of the "Declare Variable" dialog, CODESYS displays the initialization values of the array (example: \{4, \(23(0)]\) ).
13. You can optionally enter a "Comment" in the input field.
14. Click "OK" in order to conclude the declaration of the array.
\(\Rightarrow\) CODESYS adds the declaration of the array to the declaration part of the programming object.

See also
- \({ }^{\wedge}\) Chapter 6.4.1.21.3.3.32 "Command 'Auto Declare"' on page 2572
- \(\sum^{4}\) Chapter 6.4.1.20.5.15 "Data Type 'ARRAY"" on page 2247

\section*{Declaring global variables \\ Declaring global Requirement: A project is open. \\ variables that \\ are available \\ within the appli- \\ cation.}
1. In the Device tree of your project, select the application in which the global variables are to be valid.
2. Select the context menu command "Add Object \(\rightarrow\) Global Variable List".
\(\Rightarrow\) CODESYS inserts the "GVL" in the Device tree under the application and opens it in the editor.
3. Select the menu command "Edit \(\rightarrow\) Auto Declare".
\(\Rightarrow\) The dialog box "Auto Declare" opens.
4. In the selection list "Scope", select the entry "VAR_GLOBAL".
5. In the field "Name", enter a name for the global variable.
6. Select a data type from the selection list "Type".
7. If your variable is to have an initialization value other than the standard initialization value, click on .... next to the field "Initialization".
\(\Rightarrow\) The dialog box "Initialization Value" opens.
8. Double-click on the cell "Init value" of your variable and enter the desired valid value.
9. Click on "OK".
\(\Rightarrow\) The initialization value is displayed in the dialog box "Auto Declare".
10. Activate one of the "Flags" if necessary.
11. Confirm your entries by clicking on the button "OK".
\(\Rightarrow\) CODESYS inserts the declared variable in the GVL.
The global variable is available in the total application of your project.

\section*{Declaring global} variables that are available in the entire project.
1. Select the menu command "View \(\rightarrow\) POUs". \(\Rightarrow\) The "POUs" view opens.
2. In the "POUs" view, select the uppermost node with the project name and select the context menu command "Add Object \(\rightarrow\) Global Variable List".
\(\Rightarrow\) CODESYS inserts the "GVL" in the "POUs" view and opens it in the editor.
3. Select the menu command "Edit \(\rightarrow\) Auto Declare".
\(\Rightarrow\) The dialog box "Auto Declare" opens.
4. In the selection list "Scope", select the entry "VAR GLOBAL".
5. In the field "Name", enter a name for the global variable.
6. Select a data type from the selection list "Type".
7. If your variable is to have an initialization value other than the standard initialization value, enter it in the column "Initialization".
8. Activate one of the "Flags" if necessary.
9. Confirm your entries by clicking on the button "OK".
\(\Rightarrow\) CODESYS inserts the declared variable in the GVL.
The global variable is now available in the entire project.

See also
- \({ }^{4}\) Chapter 6.4.1.21.3.3.32 "Command 'Auto Declare'" on page 2572

\section*{Using Task-Local Variables}

Task-local variables are cycle-consistent. In a task cycle, they are written only by a defined task, while all other tasks have read-only access. It is taken into account that tasks can be interrupted by other tasks or can run simultaneously. The cycle consistency also applies above all if the application is running on a system with a multicore processor.

Therefore, using task local global variable lists is one way to automatically achieve a synchronization (by the compiler) when multiple tasks are processing the same variables. This is not the case when using ordinary GVLs. Multiple tasks can write simultaneously to ordinary GVL variables during a cycle.

However, it is imperative to note: The synchronization of task-local variables requires a relatively large amount of time and memory and is not always the best solution for every application. For this reason, see below for more detailed technical information and best practice guidance to help you make the right decision.

In the CODESYS project, the 嗵 "Variable List (Task-Local)" object is available for defining tasklocal variables. Syntactically, it corresponds to a normal GVL, but also contains the information of the task that has write access to the variables. Then all variables in such a GVL are not changed by another task during a cycle of a task.
The next section contains a simple example that demonstrates the principle and functionality of task-local variables. It includes a writing program and a reading program. The programs run in different tasks, but they access the same data that is stored in a task-local global variable list so that they are processed cycle-consistently.
```

Showing func- See below for Instructions on reprogramming this sample application.
tionality in an
example

```

\section*{Sample appli-}
```

cation

```
```

(* task-local GVL, object name: "Tasklocals" *)

```
(* task-local GVL, object name: "Tasklocals" *)
VAR_GLOBAL
VAR_GLOBAL
    g_diaData : ARRAY [0..99] OF DINT;
    g_diaData : ARRAY [0..99] OF DINT;
END_VA\overline{R}
END_VA\overline{R}
PROGRAM ReadData
PROGRAM ReadData
VAR
VAR
    diIndex : DINT;
    diIndex : DINT;
    bTest : BOOL;
    bTest : BOOL;
    diValue : DINT;
    diValue : DINT;
END_VAR
END_VAR
bTest := TRUE;
bTest := TRUE;
diValue := TaskLocals.g_diaData[0];
diValue := TaskLocals.g_diaData[0];
FOR diIndex := 0 TO 99 D
FOR diIndex := 0 TO 99 D
    bTest := bTest AND (diValue = Tasklocals.g_diaData[diIndex]);
    bTest := bTest AND (diValue = Tasklocals.g_diaData[diIndex]);
END_FOR
END_FOR
PROGRAM WriteData
PROGRAM WriteData
VAR
VAR
    diIndex : DINT;
    diIndex : DINT;
    diCounter : DINT;
    diCounter : DINT;
END_VAR
END_VAR
diCounter := diCounter + 1;
diCounter := diCounter + 1;
FOR diCounter := 0 TO 99 DO
FOR diCounter := 0 TO 99 DO
    Tasklocals.g_diaData[diIndex] := diCounter;
    Tasklocals.g_diaData[diIndex] := diCounter;
END_FOR
```

END_FOR

```

The programs "WriteData" and "ReadData" are called by different tasks.
In the program WriteData, the array g_diaData is populated with values. The program ReadData tests whether or not the values of the array are as expected. If so, then the variable bTest yields the result TRUE.
The array data that is tested is declared via the variable g_diaData in the object Tasklocals of type Global Variable List (Task-Local). This synchronizes the data access in the compiler and guarantees cycle consistency, even when the accessing programs are called from different tasks. In the sample program, this means that the variable test is always TRUE in the program ReadData.

If the variable g_diaData were declared only as a global variable list in this example, then the test (the variable test in the program ReadData) would yield FALSE more often. In this case, this is because one of the two tasks in the FOR loop could be interrupted by the other task, or both tasks could run simultaneously (multicore controllers). And therefore the values could be changed by the writer while the reader reads the list.

\section*{Constraints in the declaration}

\section*{NOTICE!}

An online change of the application is not possible after changes in declarations in the list of task-local variables.

Note the following when declaring a global task-local variable list:
- Do not assign direct addresses by means of an AT declaration.
- Do not map to task-local variables in the controller configuration.
- Do not declare any pointers.
- Do not declare any references.
- Do not instantiate any function blocks.
- Do not declare any task-local variables as PERSISTENT and RETAIN at the same time.

The compiler reports write access in a task without write access as an error. However, not all write-access violations can be detected. The compiler can only assign static calls to a task. However, the call of a function block by means of a pointer or an interface is not assigned to a task, for example. As a result, any write access is not recorded there either. Moreover, pointers can point to task-local variables. Therefore, data can be manipulated in a read task. In this case, a runtime error is not issued. However, values that are modified by means of pointer access are not copied back in the shared reference of variables.

Properties of The variables are located at a different address in the list for each task. For read access, this task-local global means: ADR (variable name) yields a different address in each task.
variables and possible behavior

The synchronization mechanism guarantees the following:
- Cycle consistency
- Freedom from locked states: A task never waits for an action from another task at any time.

With this method, however, no time can be determined when a reading task securely receives a copy of the writing task. Fundamentally, the copies can deviate. In the example above, it cannot be concluded that each written copy is processed one time by the reader. For example, the reading task can edit the same array over multiple cycles, or the contents of the array can skip one or more values between two cycles. Both can occur and have to be considered.

The writing task can be paused for one cycle between two accesses to the shared reference by each reading task. This means that when \(n\) reading tasks exist, the writing task can have \(n\) cycles of delay until the next update of the shared reference.
In each task, the writing task can prevent a reading task from getting a reading copy. As a result, no maximum number of cycles can be specified after which a reading task will definitely receive a copy.

In particular, this can become problematic if very slow running tasks are involved. Assuming a task runs only every hour and cannot access the task-local variables during this time, then the task works with a very old copy of the list. Therefore, it can be useful to insert a time stamp in the task-local variables so that the reading tasks can at least determine whether or not the list is up-to-date. You can set a time stamp as follows: Add a variable of type LTIME to the list of task-local variables and add the following code to the writing task, for example: tasklocal.g_timestamp := LTIME();

\section*{Best practice Task-local variables are designed for the use case "Single writer - multiple readers". When} you implement a code that is called by different tasks, using task-local variables is a significant advantage. For example, this is the case for the sample application appTasklocal as described above when it is extended by multiple reading tasks that all access the same array and use the same functions.

Task-local variables are especially useful on multicore systems. On these systems, you cannot synchronize tasks by priority. Then other synchronization mechanisms become necessary.
Do not use task-local variables when a reading task always has to work on the newest copy of the variable. Task-local variables are not suitable for this purpose.
A similar issue is the "Producer - Consumer" dilemma. This happens when a task produces data and another task processes the data. Choose another type of synchronization for this configuration. For example, the producer could use a flag to notify that a new date exists. Then the consumer can use a second flag to notify that it has processed its data and is waiting for new input. In this way, both can work on the same data. This removes the overhead for cyclic copying of data, and the consumer does not lose any data generated by the producer.

Monitoring At runtime, multiple different copies of the task-local variable list may exist in memory. When monitoring a position, not all values can be displayed. Therefore, the values from the shared reference are displayed for inline monitoring, in the watch list, and in the visualization for a task-local variable.

When you set a breakpoint, the data of the task is displayed that ran to the breakpoint and was halted as a result. Meanwhile, the other tasks continue running. Under certain circumstances, the shared copy can be changed. In the context of the halted task, however, the values remain unchanged and are displayed as they are. You need to be aware of this.

\section*{Background: Technical implementation}

For a list of task-local variables, the compiler creates a copy for each task, as well as a shared reference copy for all tasks. This creates a structure that contains the same variables as the list of task-local variables. Moreover, an array with this structure is created in which an array dimension is created for each task. As a result, an array element is indexed for each task. If a variable in the list is accessed now in the code, then the task-local copy of the list is actually accessed. Furthermore, it is determined in which task the block is currently running and the access is indexed accordingly.
For example, the line of code diValue := TaskLocals.g_diaData[0]; from the above example is replaced by:
diValue := __TaskLocalVarsArray[__CURRENTTASK.TaskIndex].__g_diarr[0];
CURRENTTASK is an operator that is available in CODESYS V3.5 SP13 and later in order to determine the current task index quickly.
At runtime, at the end of the writing task, the contents of the task-local list are written to the global list. For a reading task at the beginning, the contents of the shared reference are copied to the task-local copy. Therefore, for \(n\) tasks, there are \(n+1\) copies of the list: One list serves as a shared reference and every task also has its own copy of the list.

A scheduler controls the time-based execution of multiple tasks and therefore also task switching. The strategy, which is tracked by the scheduler in order to control the allocation of the execution time, has the goal of preventing a task from being blocked. The synchronization mechanism is therefore optimized to the properties of task-local variables to prevent blocking states (lock states) and at no time does a task wait for the action of another task.
Synchronization strategy:
- As long as the writing task writes a copy back to the shared reference, none of the reading tasks gets a copy.
- As long as a reading task gets a copy of the common reference, the writing task does not write back a copy.

Instructions for
creating the sample application as described above

Aim: With a program ReadData, you want to access the same data that is written by a program WriteData. Both programs should run in different tasks. You make the data available in a task-local variable list so that it is processed automatically in a cycle-consistent manner.
\(\checkmark\) Requirement: A brand new standard project is created and open in the editor.
1. Rename the application from Application to appTasklocal.
2. Below appTasklocal, add a program in ST named ReadData.
3. Below appTasklocal, add another program in ST named WriteData.
4. Below the object Task Configuration, rename the default task from MainTask to Read.
5. In the "Configuration" dialog of the task Read, click the "Add Call" button to call the program ReadData.
6. Below the "Task Configuration" object, add another task named Write, and add the call of the program Write to this task.
\(\Rightarrow\) Now there are two tasks Write and Read in the task configuration which call the programs WriteData and ReadData, respectively.
7. Select the application appTasklocal and add an object of type "Global Variable List (Task-Local)".
\(\Rightarrow\) The "Add Global Variable List (Task-Local)" dialog opens.
8. Specify the name Tasklocals.
9. Select the Write task from the "Task with write access" list box.

\(\Rightarrow\) The object structure for using task-local variables within an application is complete. Now you can code the objects as described in the example above.

See also
- \(\Rightarrow\) Chapter 6.4.1.21.3.3.32 "Command 'Auto Declare"" on page 2572
- \(\underbrace{*}\) Chapter 6.4.1.9.4.5 "Declaring global variables" on page 1854
- \& Chapter 6.4.1.21.2.13 "Object 'GVL' - Global Variable List (task-local)" on page 2466
- 乡 \(\boldsymbol{y}^{*}\) Chapter 6.4.1.9.12.3 "AT declaration" on page 1902
- \(\Leftrightarrow\) Chapter 6.4.1.13.2.2 "Calling of monitoring in programming objects" on page 1996

\subsection*{6.4.1.9.5 Creating Source Code in IEC}

\section*{General}

Source code:
"Source code" is a term used for the implementation code, which you insert in the programming modules by using the appropriate programming language editors. The following programming module types are available for this purpose: POU (Program, Function, Function Block), Action, Method, Property, Interface.
Programming Language:
When creating a POU, you define, in which programming language the implementation should be inserted. Besides the IEC languages also CFC is available.

Programming Language:Editors:
You get a programming module editable in the corresponding programming language editor on a double-click on the programming module object. So, the module will appear either in the textual ST editor or in one of the graphical editors for FBD/LD/IL or CFC. Each editor consists of two windows: In the upper window you insert the declarations, in textual or tabular form, depending on the setting. In the lower window you insert the implementation code. The display and behaviour of each editor can be configured in the corresponding tab of the CODESYS "Options" dialog.
Regard the possibility to open a programming module for offline-editing even while the application is in online mode.

\section*{See also}
- « Chapter 6.4.1.20.1 "Programming Languages and Editors" on page 2047
- © Chapter 6.4.1.21.3.5.11 "Command 'Edit Object (Offline)"' on page 2603

\section*{FBD/LD/IL}

\section*{General}

A combined editor enables programming in the languages FBD (function block diagram), LD (ladder diagram) and IL (instruction list).
The basic unit of the FBD and LD programming is a network. Each network contains a structure that can represent the following: a logical or arithmetic expression, the call of a POU (function, function block, program etc.), a jump or a return instruction. IL actually requires no networks. In CODESYS, however, an IL program also consists of at least one network in order to support conversion to FBD or LD. In view of this you should also divide an IL program meaningfully into networks.

See also
- \(\Leftrightarrow\) Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082

Function block diagram (FBD)

The function block diagram is a graphically oriented IEC 61131 programming language. It works with a list of networks, where each network contains a structure that can contain logical and arithmetic expressions, calls of function blocks, a jump or a return instruction.

Boxes familiar from boolean algebra are used here. Boxes and variables are connected by connecting lines. The signal flow in the network runs from left to right. The signal flow in the editor runs from top to bottom, starting with network 1.


CFC is also a programming language based on the same principle as FBD, but with the following differences:
- The CFC editor is not network-oriented.
- You can freely place the elements in the CFC editor.
- Direct insertion of feedbacks is possible.
- The order of execution is determined by a list of currently inserted elements, which you can change.

See also
- \(\quad\) Chapter 6.4.1.9.5.2.2 "Programming function block diagrams (FBD)" on page 1862
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.14 "Menu 'FBD/LD/IL'" on page 2700(commands)
- \({ }^{\wedge}\) Chapter 6.4.1.9.5.3 "Continuous Function Chart (CFC)" on page 1866

Ladder diagram (LD)

The ladder diagram (LD) is a graphically oriented programming language that approximates an electrical circuit diagram. On the one hand the ladder diagram is suitable for designing logical switching units, but on the other you can also create networks just as in FBD. Therefore you can use LD very well for controlling calls of other program blocks.

The ladder diagram consists of a series of networks. A network is bounded on the left side by a vertical line (bus bar). A network contains a circuit diagram of contacts, coils, optional boxes (POUs) and connecting lines. On the left side of a network there is a contact or a series of contacts that relay the ON or OFF state, which corresponds to the boolean values TRUE and FALSE, from left to right. A boolean variable is associated with each contact. If this variable is TRUE, the status is relayed from left to right via the connection line. Otherwise OFF is relayed. Thus the coil(s) in the right part of the network receive(s) the value ON and OFF coming from the left and the value TRUE or FALSE is written accordingly into the boolean variable assigned to them.
If the elements are connected in series, this means an AND operation. If they are connected in parallel, this means an OR operation. A line through an element means a negation of the element. The negation of an input or an output is indicated by a circle symbol.


IEC 61131-3 defines a complete LD command set, consisting of different types of contacts and coils. Contacts conduct the current (according to their type) from left to right. Coils store the incoming value. Contacts and coils are assigned to boolean variables. You can supplement an LD network by jumps, returns, labels and comments.
See also
- "3 Chapter 6.4.1.9.5.2.3 "Programming ladder diagrams (LD)" on page 1864
- \& Chapter 6.4.1.21.3.14 "Menu 'FBD/LD/LL"' on page 2700(Befehle)

Instruction list
(IL)

The instruction list is an assembler-like IEC 61131-compliant programming language. It supports accumulator-based programming.
An instruction list (IL) consists of a series of instructions. Each instruction starts in a new line and contains an operator and, depending on the type of operation, one or more operands separated by commas. A label, followed by a colon, can be placed in front of an instruction. It serves the identification of the instruction and you can use the label as a jump destination. A comment must be the last element in a line. Empty lines can be inserted between instructions.
All IEC 61131-3 operators are supported, as are multiple inputs, multiple outputs, negations, comments, set/reset of outputs and conditional/unconditional jumps.
Each instruction is based primarily on the loading of values into the accumulator (LD instruction). After that the corresponding operation is executed with the parameter from the accumulator. The result of the operation is written again into the accumulator, from where you should store it purposefully with the help of an ST instruction.
The instruction list supports comparison operators (EQ, GT, LT, GE, LE, NE) and jumps for programming of conditional executions or loops. Jumps can be unconditional (JMP) or conditional (JMPC / JMPCN). In the case of conditional jumps, a check is performed as to whether the value in the accumulator is TRUE or FALSE.


See also
- * Chapter 6.4.1.9.5.2.4 "Programming in instruction list (IL)" on page 1865
- * Chapter 6.4.1.20.1.6.3 "Modifiers and operators in IL" on page 2087

\section*{Programming function block diagrams (FBD)}

Creating a POU in the function block diagram (FBD) implementation language
1. Select an application in the device tree.
2. Select the command "Project \(\rightarrow\) Add Object \(\rightarrow\) POU".
\(\Rightarrow\) The dialog box "Add POU" opens.
3. Enter a name and select the implementation language "Function Block Diagram (FBD)".

Click on "Add".
\(\Rightarrow\) The POU is added to the device tree and opened in the editor. It consists of the declaration editor in the top part and the implementation part with an empty network in the lower part. The view "ToolsBox" is also automatically opened, in which the suitable elements, operators and function blocks for FBD programming are available.

See also
- 党 Chapter 6.4.1.21.2.21 "Object 'POU'" on page 2477
- \({ }^{4}\) Chapter 6.4.1.9.5.2 "FBD/LD/IL" on page 1860

\section*{Programming a} network
1. Click inside the automatically inserted empty network in the implementation part.
\(\Rightarrow\) The network is given a yellow background and the area at the left-hand side with the network number is given a red background.
2. Open the context menu with the right mouse button.
\(\Rightarrow\) You obtain amongst other things the insert commands for the elements that can be inserted at this point.
3. Insert the elements required for your programming using the menu commands or by dragging in the elements from the toolbox.
4. For example, select the command "Insert Assignment".
\(\Rightarrow\) An assignment line is inserted. In each case three question marks stand for assignment source and assignment target.
5. Select the question marks and replace them with the desired variable. Input assistance is available for this purpose.
6. Move the cursor over the assignment line.
\(\Rightarrow\) The possible insertion positions for further elements are displayed as grey diamonds. A click on a diamond selects that position and the suitable insert commands are once again available.
7. Alternatively, you can drag an element with the mouse from the toolbox into the network. For example, click in the tool box on the box element, keep the mouse button pressed and move the cursor over the network.
\(\Rightarrow\) Each possible insertion position lights up green.
8. Release the mouse button in order to insert the box.
\(\Rightarrow\) The box is displayed in the network. The type of box on the inside and the instance name above the box, which is required in the case of a function block, are still kept free with three question marks.
9. Select the string ?? ? inside the box and replace it with the name of the box. Input assistance is available for this purpose.
\(\Rightarrow\) The inputs and outputs of the selected box are displayed. They are still kept free with question marks, as is the instance name in the case of a function block.

See also
- \({ }^{\wedge}\) Chapter 6.4.1.21.3.14 "Menu 'FBD/LD/IL"' on page 2700
- \({ }^{4}\) Chapter 6.4.1.9.5.2 "FBD/LD/IL" on page 1860

\section*{Programming}
line branches
(subnetworks)
1. In the implementation part of your POU, insert a new network using the command "FBD/LD/IL \(\rightarrow\) Insert network" or drag it in from the tool box.
2. For example, drag an "ADD" operator into the empty network and replace the characters ? ? ? with two variables of the type INT.
3. Drag the element "Branch" from the tool box into your implementation and release the mouse button at the green insertion position directly at the output of the operator.
\(\Rightarrow\) The line branch splits the processing line at the output of the operator box into 2 subnetworks.
4. Further FBD elements and also further line branches can now be added to each of the two subnetworks.

See also
- \({ }^{*}\), Chapter 6.4.1.20.1.6.4.9 "FBD/LD/IL element 'Branch'" on page 2093
- \({ }^{\bullet} \boldsymbol{y}\) Chapter 6.4.1.21.3.14.33 "Command 'Insert Branch"" on page 2709

See also
- \(\quad 4\) Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082

\section*{Programming ladder diagrams (LD)}

Creating a POU
in the ladder
diagram (LD)
implementation
language
1. Select the application in the Device tree.
2. Select the command "Project \(\rightarrow\) Add Object \(\rightarrow\) POU".
\(\Rightarrow\) The dialog box "Add POU" opens.
3. Enter a name and select the implementation language "Ladder Diagram (LD)".

Click on "Add".
\(\Rightarrow\) CODESYS adds the POU to the Device tree and opens it in the editor. An empty network is inserted in the implementation part. The empty network is bounded on the left by a vertical line, which represents a bus bar. The view "ToolBox" is also automatically opened, in which the suitable elements, operators and function blocks for LD programming are available.

\section*{Adding a contact and a function block (TON)}
\(\checkmark\) Requirement: a POU with the implementation language LD is opened in the editor and an empty network is inserted.
1. Click on the category "Ladder Elements" in the view "ToolBox"
2. Click on the "Contact" element, drag it into your network and release the mouse button at the insertion position "Start here".
\(\Rightarrow\) The contact is added on the left in the network directly against the vertical line.
3. Click on ?? ? and enter the identifier of a boolean variable. The input assistant is also available to you for this.
4. Click on the category "Function Blocks" in the view "ToolBox" and drag the function block "TON" onto an insertion position on the connecting line to the right of the inserted contact.
\(\Rightarrow\) CODESYS inserts the box "TON" to the right of the contact. The contact is connected with the input IN of the TON box.
5. Enter a time constant at the input PT , for example \(\mathrm{T} \# 3 \mathrm{~s}\).
\(\Rightarrow\) If the variable of your contact goes TRUE, then the input IN of the TON box also goes TRUE. The TON box forwards the value TRUE to the output \(Q\) with a switch-on delay of T\#3s, for example.
\(\begin{array}{ll}\text { Inserting a } & \text { Requirement: a POU with the implementation language LD is opened in the editor and an empty } \\ \text { closed line } & \text { network is inserted. }\end{array}\) branch
1. Click inside the empty network and select the command "FBD/LD/IL \(\rightarrow\) Insert Contact".
2. Select the connecting line to the left of the contact and select the command "FBD/LD/IL \(\rightarrow\) Set Branch Start Point".
\(\Rightarrow\) The starting point on the connecting line is marked by a red rectangle. CODESYS marks all possible end points of the branch with a blue rectangle.
3. Click on a blue rectangle in order to set the end point of your closed line branch.
\(\Rightarrow\) CODESYS inserts the line branch between the starting and end points. The program flow will go through both branches up to the end point.

If you insert the line branch at a box instead of at a contact, the box will only be called if none of the other branches is TRUE.

See also
- \(\quad\) - Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082
- \(\quad\) Chapter 6.4.1.20.1.6.4.14 "Closed branch" on page 2096
- \(\quad \stackrel{y}{c}\) Chapter 6.4.1.20.1.6.4.11 "LD element 'Contact'" on page 2094

\section*{Programming in instruction list (IL)}

\section*{\(\square\) \\ If necessary, IL can be activated in the CODESYS options.}

\section*{Creating POUs}
in the instruc-
tion list (IL) implementation language

Programming networks (example: ADD operation)
1. Select the application in the device tree.
2. Click "Project \(\rightarrow\) Add Object \(\rightarrow\) POU".
\(\Rightarrow\) The "Add POU" dialog opens.
3. Enter a name and select the implementation language "Instruction List (IL)".

Click "Add".
\(\Rightarrow\) CODESYS adds the POU to the device tree and opens it in the editor. A network is already inserted in the implementation part.

Requirement: A POU (IL) is opened in the editor and possesses an empty network.
1. Click the line marked in color in the 1st column and enter the operator LD.
2. Press the [Tab] key.
\(\Rightarrow\) The cursor jumps to the 2nd column
3. Enter the first summand of your ADD operation, for example 6.
4. Press [Ctrl]+[Enter] or select the command "FBD/LD/IL \(\rightarrow\) Insert IL Line After".
\(\Rightarrow\) CODESYS inserts a new instruction line. The first column of this line has the focus.
5. Enter ADD and press [Tab].
6. Enter the second summand of your ADD operation, for example 12.
7. Press [Ctrl]+[Enter]
8. Enter the operator ST and press [Tab].
9. Specify a variable of the data type INT, for example iVar.
\(\Rightarrow\) The result - 16 in the example - is stored in the iVar.

\begin{abstract}
Calling function Requirement: A POU (IL) is opened in the editor and possesses an empty network. A variable of blocks the data type <function block> is declared in the declaration part (example: C1:CTU; ).
1. Click the line marked in color in the 1st column and select the command "FBD/LD/IL
\(\rightarrow\) Insert Box".
\(\Rightarrow\) The input assistant opens.
2. Select the desired function block in the category "Function Blocks" or "Boxes", for example the "CTU" counter from the "Standard" library, and click "OK".
\(\Rightarrow\) CODESYS inserts the selected function block "CTU" as follows:
\end{abstract}
\begin{tabular}{|c|c|}
\hline CAL & ??? \\
\hline & CU: \(=\) ? ? ? \({ }^{\text {anmm }}\) \\
\hline & RESET: \(=\) ? ? ? \\
\hline & PV: = ? ? ? \\
\hline & \(\mathrm{Q}=>\) ? \({ }^{\text {a }}\) ? , \\
\hline & CV=> ?? ? \\
\hline
\end{tabular}
3. Replace the strings ? ? ? with the variable name and the values or variables for the inputs/ outputs of the function block.
4. As an alternative to inserting the function block via the input assistant, you can directly enter the call in the editor as shown in the picture at step 4.

See also
- \(\&\) Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082
- \({ }^{\text {• }}\) Chapter 6.4.1.20.1.6.3 "Modifiers and operators in IL" on page 2087

\section*{Continuous Function Chart (CFC)}

\section*{General}

The "Continuous Function Chart (CFC)" implementation language is a graphical programming language which extends the standard languages of IEC 61131-3.
You can graphically program a system by means of a POU in CFC. You insert elements and position them freely. You insert connections and wire the elements to a network so that a well-structured function block diagram is created. You can also insert feedback. You can read function block diagrams like an circuit diagram or a block diagram.

The execution order of a function block diagram is based on data flow. Moreover, a POU can process multiple data flows. Then the data flows do not have any common data. In the editor, multiple networks do not have any connections to each other.

On the other hand, POUs in FBD, LD, or IL have a network-based execution order.

\section*{Unfortunately, it is not possible to switch a POU between the "Continuous} Function Chart (CFC) - page-oriented" and "Continuous Function Chart (CFC)" implementation languages.

The "Continuous Function Chart (CFC) - page-oriented" implementation language is a variant of CFC. The difference from simple CFC is that the code can be distributed on any number of "pages" in the editor. In this way, you can create extensive function block diagrams which are still easy to print.

Each page has edge areas where you arrange the connections to the other pages.
- Creating pages
- Setting the page size
- Copying and inserting pages in the page navigator
- Copying the implementation of a POU in the CFC implementation language and inserting into a page
- Well-structured and space-saving arranging of inputs, outputs, and connection marks in the border areas
- Connection over pages with connection marks
* \({ }^{4}\) Chapter 6.4.1.20.1.7.2 "CFC Editor" on page 2098
* \({ }^{\text { }}\) Chapter 6.4.1.20.1.7.3 "CFC editor, page-oriented" on page 2101
« « Chapter 6.4.1.21.3.13 "Menu 'CFC'" on page 2685
Chapter 6.4.1.21.4.11.14 "Dialog 'Properties' - 'CFC Execution Order'" on page 2761

\section*{Automatic Execution Order by Data Flow}

The execution order in POUs is uniquely determined in text-based and network-based editors. In the CFC editor, however, you can position the elements freely, so the execution order is initially not unique. For this reason, CODESYS determines the execution order by data flow and, in the case of multiple networks, by the topological position of the elements. The elements are sorted from top to bottom and left to right. Now the execution order is unique and makes sure that the POU is processed while optimized by time and by cycle.
You can get information about the chronological order of the elements in the chart and temporarily display the execution order. When you program networks with feedback you can define an element as the starting point in the feedback loop.
You can also explicitly edit the processing order in a CFC object if necessary. To do this, switch the "Auto Data Flow Mode" property of the CFC object to "Explicit Execution Order Mode". In this mode, you have the option of editing the execution order by means of menu commands.
Before CODESYS Development System V3.5 SP15, you had to define the execution order explicitly for each POU. The was no mode switching.

Data flow In general, data flow is described as the chronological order in which data is read or written when and how in which programming objects. A POU can process any number of data flows, which can also be executed independently of each other.

\footnotetext{
Displaying the By default, the execution order of a CFC object is determined automatically. The "Auto Data execution order Flow Mode" property is selected for this. You can temporarily display the automatically determined execution order in the CFC editor.
}
1. Create a new project using the "Standard project" template and specify the name Minimal for example.
2. Extend the application with the function block FB_DOIt in the "ST" implementation language with inputs and outputs.
```

=>
FUNCTION_BLOCK FB_DoIt
VAR_INPUT
iAlfa : INT;
iBravo: INT;
sCharlie : STRING := 'Charlie';
xItem : BOOL;
END VAR
VAR -OUTPUT
iResult : INT;
sResult : STRING;
xResult : BOOL;
END_VAR
VAR
END_VAR
iResult := iAlfa + iBravo;
IF xItem = TRUE THEN
xResult := TRUE;
END_IF

```
3. Create the function block ExecuteCFC in the "CFC" implementation language.
```

=>
PROGRAM ExecuteCFC
VAR
fb_DoIt_0: FB_DoIt;
fb_DoIt_1: FB_DoIt;
iFinal_1: INT;
iFinal 0: INT;
xFinal: BOOL;
END_VAR

```


Recently created programming objects in CFC have the Auto Data Flow Mode selected. The execution order of the programming object is optimally defined internally.
4. Click "CFC \(\rightarrow\) Execution Order \(\rightarrow\) Display Execution Order".
\(\Rightarrow\) The execution order of the object is shown. The boxes and inputs are numbered accordingly and reflect the chronological processing order. The numbering is hidden as soon as you click again in the CFC editor.


\section*{Determining the execution order in feedback networks}
1. Create a CFC program with feedback.
\(\Rightarrow\) The POU PrgPositiveFeedback counts.

PROGRAM PrgPositiveFeedback
VAR
iResult: INT;
END VAR

2. Select an element within the feedback.
\(\Rightarrow\) The selected element is highlighted in red.
3. Click "CFC \(\rightarrow\) Execution Order \(\rightarrow\) Set Start of Feedback".

\(\Rightarrow\) At run time, this POU is processed first. The start POU of the feedback is defined and decorated with the symbol. The execution order is resorted and the selected element gets the number 0 . (This is the lowest number of the feedback.)
4. Select the start POU again.

\section*{5. Click "CFC \(\rightarrow\) Execution Order \(\boldsymbol{\rightarrow}\) Set Start of Feedback".}
\(\Rightarrow\) The POU is not selected as the start POU.


The execution order is defined internally.
6. Click "CFC \(\rightarrow\) Execution Order \(\rightarrow\) Display Execution Order".
\(\Rightarrow\) The execution order by data flow is displayed.


\section*{Defining the execution order explicitly}

The automatically defined execution order by data flow results in time- and cycle-optimized execution of the POU. You do not need any information about the internally managed execution order during the development process.
In "Explicit Execution Order Mode", it is your responsibility to adapt the execution order and to assess the consequences and impacts. This is another reason why the execution order is always displayed.

You can change the automatically defined execution order of a CFC object explicitly when you select the "Explicit Execution Order Mode" option for the object.
1. In the "Devices" or "POUs" view, select a CFC object.
2. In he context menu, click "Properties".
3. Click the "CFC Execution Order" tab.
\(\Rightarrow\) The "Execution order" list box displays the currently selected mode.
4. In the "Execution order" list box, select "Explicit Execution Order Mode".
5. Click "OK" to confirm the dialog.
\(\Rightarrow\) The Explicit Execution Order Mode property is selected. The networks are numbered in the CFC editor, and the following commands are provided in the "CFC \(\rightarrow\) Execution Order" menu for editing the execution order.
6. Open a CFC object.
7. Select a numbered element and click "CFC \(\rightarrow\) Execution Order \(\rightarrow\) Send to Front".
\(\Rightarrow\) The execution order is resorted and the selected element has the number 0 .

See also
- \({ }^{4}\) ) Chapter 6.4.1.20.1.7.2 "CFC Editor" on page 2098
- \(y^{4}\) Chapter 6.4.1.20.1.7.3 "CFC editor, page-oriented" on page 2101
- \(y^{4}\) Chapter 6.4.1.21.3.13 "Menu 'CFC'" on page 2685
- 染 Chapter 6.4.1.21.4.11.14 "Dialog 'Properties' - 'CFC Execution Order'" on page 2761

\section*{Programming in the CFC editor}

In the CFC editor, you can wire POUs to each other and create well-structured block diagrams.
The editor supports you in the following ways:
- Programming with elements and connecting lines
- Dragging instances and variables to the editing area
- Auto-routing the connecting lines
- Automatic linking
- Fixing of connecting lines by control points
- Collision detection
- Input assistance for connection marks
- Forcing and writing of values in online mode
- Movement of selection using arrow keys
- Reduced display of a POU without disconnected pins

\section*{Inserting elements and wiring with connecting lines}
1. Drag a "Box" element and an "Output" element into the editor.
2. Click the output of the "Box" element.
\(\Rightarrow\) The output is marked with a red box.
3. Drag a connecting line from the box output of the "Box" element to the box input of the "Output" element.
\(\Rightarrow\) The cursor symbol changes when it reaches the box input.
4. Release the left mouse button.
\(\Rightarrow\) The output pin of the box is wired to the input pin of the output.


You can also hold down the [Ctrl] key, select each pin, and then right-click "Connected Selected Pins".

\section*{Calling of instances}
1. Create a new project using the standard template and specify the name First for example.
\(\Rightarrow\) The project First.project is created.
2. Extend the application with the function block FB_DOIt in the "ST" implementation language with inputs and outputs.
```

=>
FUNCTION_BLOCK FB_DoIt
VAR_INPUT
iAlfa : INT;
iBravo: INT;
sCharlie : STRING := 'Charlie';
xItem : BOOL;
END VAR
VAR_OUTPUT
iResult : INT;
sResult : STRING;
xResult : BOOL;
END VAR
VAR
END_VAR
iResult := iAlfa + iBravo;
IF xItem = TRUE THEN
xResult := TRUE;
END_IF

```
3. Select the application and click "Add Object \(\rightarrow P O U\) " in the context menu. Select the "Continuous Function Chart (CFC)" implementation language and the type Program. Specify the name PrgFirst for example.

Click "OK" to confirm the dialog.
\(\Rightarrow\) The program PrgFirst is created and it opens in the editor. It is still empty.
4. Instantiate function blocks and declare variables.
```

=>
PROGRAM PrgFirst
VAR
iCounter: INT;
fbDoIt_1 : FB_DoIt;
fbDoIt_2 : FB_DoIt;
iOut : INT;
sOut: STRING;
xOut: BOOL;
END_VAR

```
5. Drag a "Box" element from the "ToolBox" view into the editor.
6. Click the ??? field and type in ADD.
\(\Rightarrow\) The box type is ADD. The box acts as an adder.
7. Click line 3 in the declaration editor.
\(\Rightarrow\) The declaration line of iCounter is selected.
8. Click in the selection and drag the selected variable into the implementation. Focus there on an input of the ADD box.
\(\Rightarrow\) An input has been created, declared, and connected to the box.
9. Click again in the selection and drag the variable to the output of the ADD box.
\(\Rightarrow\) An output has been created, declared, and connected to the box.
10. Drag an "Input" element from the "ToolBox" view to the implementation. Click its ? ? ? field and type in 1.
11. Connect the 1 input to an input of the ADD box.
\(\Rightarrow\) A network is programmed. At runtime, the network counts the bus cycles and stores the result in iCounter.

12. Click line 5 in the declaration editor.
\(\Rightarrow\) The line is selected.
13. Click in the selection and drag the selected instance into the implementation.
\(\Rightarrow\) The instance appears as a POU in the editor. The type, name, and POU pins are displayed accordingly.


\section*{fbDolt_1}

14. Drag the fbDoIt 2 instance to the editor. Interconnect the instances to each other and to inputs and outputs.
\(\Rightarrow\) Example:


A program in ST with the same functionality might look like this:
```

PROGRAM PrgFirstInSt
VAR
iCounter: INT;
fbDoIt_1 : FB_DoIt;
fbDoIt_2 : FB_DoIt;
iOut : INT;
sOut: STRING;
xOut: BOOL;
END VAR
iCounter := iCounter + 1;
fbDoIt_1(iAlfa := 16, iBravo := 32, sCharlie := 'First',
xItem := TRUE, iDelta := 2, iResult => fbDoIt_2.iAlfa, xResult
=> fbDoIt 2.xItem);
fbDoIt_2(iBravo := fbDoIt_1.iResult, sCharlie := 'Second',
iDelta := 2, iResult => iOut , sResult=> sOut, xResult =>
xOut);

```

\section*{Creating con-} nection marksRequirement: A CFC POU has connected elements.
1. Select a connecting line between two elements.
\(\Rightarrow\) The connecting line is displayed as selected. The ends of the connecting line are marked with red boxes (릭).
2. Click "CFC \(\rightarrow\) Connection Mark".
\(\Rightarrow\) The connection is separated into a "Connection Mark - Source" and a "Connection Mark - Sink". The name of the mark is generated automatically.

3. Click in the source connection marks.
\(\Rightarrow\) You can edit the name.
4. Specify a name SimpleMark for the source connection mark.
\(\Rightarrow\) The source connection mark and sink connection mark have the same name.


Resolving collisions and fixing connecting lines by means of control points
1. Position the "Input" and "Output" elements. Connect the elements.

The following example shows how to use the "Route All Connections" command with control points.

xInput xOutput
2. Position two "Box" elements on the line.
\(\Rightarrow\) The connecting line and the boxes are marked red because of the collision.

3. Click "CFC \(\rightarrow\) Routing \(\rightarrow\) Route All Connections".
\(\Rightarrow\) The collision is resolved.

4. Change the connecting lines gradually.

\(\Rightarrow\) The connecting line has been changed manually and is now blocked for auto-routing. This is shown by a lock symbol at the end of the connection.

5. Select the connecting line and click "CFC \(\rightarrow\) Routing \(\rightarrow\) Create Control Point".
\(\Rightarrow A\) control point is created on the connecting line. The connecting line is fixed to the control point.


You can also drag a control point from the "ToolBox" view to a line.
6. Change the connecting line as seen in the following example.

\(\Rightarrow\) Use the control point for changing the connecting line according to your needs. You can set any number of control points.
7. In the context menu, click "CFC \(\rightarrow\) Routing \(\rightarrow\) Remove Control Point" to remove the control point.
8. Unlock the connection by clicking "Unlock Connection" or by clicking the lock symbol.
9. Select the connecting line and click "Route All Connections".
\(\Rightarrow\) The connecting line is routed automatically as seen in Step 3.


\section*{NOTICE!}

Connections in a group are not auto-routed.

Reducing the display of a POU
\(\checkmark\) Requirement: A CFC POU is open. In the editor, its POUs with all declared pins are displayed.
1. Select a POU whose pins are partially disconnected.
\(\Rightarrow\) Example: fb_DoIt_1


The POU needs space for all of the pins.
2. Click "CFC \(\rightarrow\) Pins \(\rightarrow\) Remove Unused Pins".
\(\Rightarrow\) Now the POU needs less space and is displayed only with the functionally relevant pins.


See also
- \({ }^{4}\) Chapter 6.4.1.20.1.3 "Common functions in graphical editors" on page 2048
- \({ }^{*}\) Chapter 6.4.1.20.1.7.2 "CFC Editor" on page 2098
- \({ }^{\bullet}\) Chapter 6.4.1.20.1.7.3 "CFC editor, page-oriented" on page 2101
- \(\Rightarrow\) Chapter 6.4.1.20.1.7.6 "Elements" on page 2109

\section*{Structured Text (ST), Extended Structured Text (ExST)}

\section*{General}

The ST editor is used for the programming of POUs in the IEC-61131-3 programming language Structured Text (ST) and Extended Structured Text. The Extended Structured Text offers some additional functions with regard to the IEC 61131-3 standard.

Structured Text is a programming language, comparable with other high-level languages such as C or PASCAL, which permits the development of complex algorithms. The program code consists of a combination of expressions and instructions, which can also be executed conditionally (IF... THEN... ELSE) or in loops (WHILE... DO).
An expression is a construct that returns a value following its evaluation. Expressions are also operators and operands together. You can also use assignments as expressions. An operand can be a constant, a variable, a function call or a further expression.
Instructions control how the expressions are to be processed.
For this text editor you can make various settings with regard to behavior, appearance and menus in the dialog boxes "Options" and "Adapt" in the "Tools" menu. The familiar Windows functions (for example IntelliMouse) are also available for this editor.
See also
- \(\Leftrightarrow\) Chapter 6.4.1.9.5.4.2 "Programming structured text (ST)" on page 1879
- \(\Leftrightarrow\) Chapter 6.4.1.21.4.14.26 "Dialog 'Options' - 'Text Editor"' on page 2798
- \(y^{\boldsymbol{y}} \boldsymbol{y}\) Chapter 6.4.1.21.4.15.2 "Dialog 'Customize' - 'Menu"' on page 2801

ExST - Extended Extended Structured Text (ExST) is a CODESYS-specific extension of the IEC 61131-3 structured text standard for Structured Text (ST).
See also
- \({ }^{\wedge}\) Chapter 6.4.1.20.1.4.4.4 "ExST assignment ' \(R=\) "' on page 2053
- \(\Rightarrow\) Chapter 6.4.1.20.1.4.4.3 "ExST assignment 'S="' on page 2052
- \({ }^{4}\), Chapter 6.4.1.20.1.4.4.5 "ExST - Assignment as expression" on page 2054

\section*{Programming structured text (ST)}

The programming languages 'Structured Text' and 'Extended Structured Text' are programmed in the ST editor. The program code consists of a combination of expressions and instructions, which can also be executed conditionally or in loops. You must conclude each instruction with a semicolon;
The variables are declared in the declaration editor.

\section*{Creating a POU in the structured text (ST) implementation language}
1. Select an application in the device tree.
2. Select the command "Project \(\rightarrow\) Add Object \(\rightarrow\) POU".
\(\Rightarrow\) The dialog box "Add POU" opens.
3. Enter a name and select the "Implementation language""Structured Text (ST)". Click on "Add".
\(\Rightarrow\) The POU is added to the device tree and opened in the editor.
Now insert the variable declarations in the upper part of the POU and enter the ST program code in the lower part of the POU.

See also
- \(y^{\text {E }}\) Chapter 6.4.1.20.1.4.3 "ST expressions" on page 2050
- \(y^{2}\) Chapter 6.4.1.20.1.4 "Structured Text and Extended Structured Text (ExST)" on page 2049
- \({ }^{4}\) Chapter 6.4.1.20.1.4.5.10 "ST function block call" on page 2060
- \#y Chapter 6.4.1.20.1.4.5.11 "ST - Comments" on page 2061
- \(\Leftrightarrow\) Chapter 6.4.1.9.4.2 "Using the declaration editor" on page 1851

\section*{Sequential Function Chart (SFC)}

\section*{General}

Use the SFC editor for programming POUs in the IEC 61131-3 compliant SFC implementation language. SFC is a graphical programming language for describing the chronological sequence of individual actions in a program. For this purpose, actions (discrete programming objects) are assigned to step elements. Transition elements control the processing order of steps.
See also
- \({ }^{〔}>\) Chapter 6.4.1.20.1.5.1 "SFC editor" on page 2062

\section*{Programming in SFC}

Creating a POU
in SFC
1. Select an application in the device tree.
2. Click "Project \(\rightarrow\) Add Object \(\rightarrow\) POU".
\(\Rightarrow\) The "Add POU" dialog opens.
3. Specify a name and select the "Sequential Function Chart (SFC)" implementation language.

Click "Add".
\(\Rightarrow\) CODESYS adds the POU to the device tree and opens it in the editor.

\section*{Adding a step-} transition
1. Select the transition after the initial step. \(\Rightarrow\) The transition is marked in red.
2. Click "SFC \(\rightarrow\) Insert Step-Transition After".
\(\Rightarrow\) CODESYS inserts the "Step0" step and the "Trans0" transition.
3. Select the "Trans0" transition and click "SFC \(\rightarrow\) Insert Step-Transition".
\(\Rightarrow\) CODESYS inserts the "Trans1" transition and the "Step1" step before the "Trans0" transition.

You can also drag the "Step" and "Transition" elements into the diagram from the "Toolbox" view. See also
- \(\Longleftrightarrow\) Chapter 6.4.1.20.1.5.8.1 "SFC elements 'Step' and 'Transition"' on page 2073
- \({ }^{*}>\) Chapter 6.4.1.21.3.12.6 "Command 'Insert Step-Transition"' on page 2677
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.12.7 "Command 'Insert Step-Transition After'" on page 2677
1. Select the "StepO" step.
2. Click "SFC \(\rightarrow\) Add Entry Action".
\(\Rightarrow\) By default, you are prompted to select the duplication mode for the step actions. You decide whether the reference information about the existing step action objects is copied when the step is copied, or the objects are embedded. Embedding results in new step action objects being created when the step is copied. The duplication mode is defined in the "Duplicate when copying" step property. When this property is deactivated, the copied steps call the same actions as the current step.

You can deactivate the prompt completely in the SFC properties.
The display of embedded objects in the "Devices" and "POUs" views can be deactivated by means of a menu command.
3. For this example, accept the "Copy reference" default setting and click "OK" to confirm. \(\Rightarrow\) The "Add Entry Action" dialog opens.
4. Enter the name "Step0_entry" and select the "Structured Text (ST)" implementation language. Click "Add".
\(\Rightarrow\) CODESYS inserts the "Step0_entry" action below the POU in the device tree and opens the action in the editor.
In the Step0_entry entry action, you program statements to be executed one time when the "Step0" step becomes active.
5. Close the editor of Step0_entry.
\(\Rightarrow\) The "Step0" step is now marked with an " \(E\) " in the lower left corner. Double-click this marker to open the editor.


The entry action Step0_entry is now available in the properties of the step in "Entry action". Other actions cān also be selected there as needed.
6. Select the "Step0" step. Press [Ctrl]+[V] to copy the step.
\(\Rightarrow\) The same entry actions inserted above are available In the inserted copy of the step. The new step then calls the same exact action.

See also
- \(\xi^{*}\) Chapter 6.4.1.21.3.12.8 "Command 'Add Entry Action"" on page 2678
- \& "2. Step actions" on page 2076
- \({ }^{\mu}\), Chapter 6.4.1.21.4.14.23 " Dialog 'Options' - 'SFC Editor'" on page 2795

\section*{Adding an exit} action
1. Select the "Step0" step.
2. Click "SFC \(\rightarrow\) Insert Exit Action".
\(\Rightarrow\) By default, you are prompted to select the duplication mode for the step actions of the step. See above for adding an entry action. Then the "Insert Exit Action" dialog opens.
3. Enter the name "Step0_exit" and select the "Structured Text (ST)" implementation language. Click "Add".
\(\Rightarrow\) CODESYS inserts the "Step0_exit" action below the POU in the device tree and opens the action in the editor.

In the Step0_exit exit action, you program statements to be executed one time before the "Step0" step becomes inactive.
4. Close the editor of Step0_exit.
\(\Rightarrow\) The "Step0" step is now marked with an " \(X\) " in the lower right corner. Double-click this marker to open the editor.


You can define the exit action in the properties of the step in "Exit action". Other actions can also be selected there.

See also
- \({ }^{\text {• }}\) Chapter 6.4.1.21.3.12.9 "Command 'Add Exit Action'" on page 2678
- «2 "2. Step actions" on page 2076

\section*{Adding an action}
1. Double-click the "Step0" step.
\(\Rightarrow\) By default, you are prompted to select the duplication mode for the step actions of the step. See above for adding an entry action. The "Add Action" dialog opens.
2. Type in the name "Step0_active" and select the "Structured Text (ST)" implementation language. Click "Add".
\(\Rightarrow\) CODESYS inserts the "Step0_active" action below the POU in the device tree and opens the action in the editor.
In the Step0_active step action, you program statements to be executed as long as the step is active.
3. Close the editor of Step0_active.
\(\Rightarrow\) The "Step0" step is now marked with a black triangle in the upper right corner.


You can define the action in the properties of the step in "Step action". Other actions can also be selected there.
See also
- 良 "2. Step actions" on page 2076
1. Select the "Step1" step.
2. Click "SFC \(\rightarrow\) Insert Branch Right".
\(\Rightarrow\) CODESYS inserts the "Step2" step to the right of "Step1". The steps are connected as a parallel branch signified by two pairs of double lines.
3. Select one of the double lines.
\(\Rightarrow\) The double line is marked red.
4. Click "SFC \(\rightarrow\) Alternative"
\(\Rightarrow\) CODESYS converts the branch into an alternative branch. The double lines change into a single line.

You can click "SFC \(\rightarrow\) Parallel" to convert an alternative branch into a parallel branch.
See also
- \({ }^{〔}\) Chapter 6.4.1.20.1.5.8.3 "SFC element 'Branch"" on page 2078
- \({ }^{*}\) Chapter 6.4.1.21.3.12.10 "Command 'Parallel"' on page 2678
- \(y^{*}>\) Chapter 6.4.1.21.3.12.12 "Command 'Insert Branch"" on page 2679
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.12.13 "Command 'Insert Branch Right"' on page 2679

\section*{Adding a jump}
1. Select the "Step2" step.
2. Click "SFC \(\rightarrow\) Insert Jump After".
\(\Rightarrow\) CODESYS inserts the "Step" jump after the "Step2" step.
3. Select the "Step" jump destination.
\(\Rightarrow\) You can type the jump destination manually or select it by using the Input Assistant ...). Select Step0.


See also
- \({ }^{\text {n }}\) Chapter 6.4.1.20.1.5.8.4 "SFC element 'Jump'" on page 2079
- \({ }^{\bullet}\) Chapter 6.4.1.21.3.12.16 "Command 'Insert Jump'" on page 2681
- \(\boldsymbol{\mu}^{\bullet}\) Chapter 6.4.1.21.3.12.17 "Command 'Insert Jump After'" on page 2681

\section*{Adding a macro}
1. Select the "Step1" step.
2. Click "SFC \(\rightarrow\) Insert Macro After".
\(\Rightarrow\) CODESYS inserts the "Macro0" macro after the "Step1" step.
3. Double-click the "MacroO" element.
\(\Rightarrow\) The macro opens in the implementation section of the editor. The name "Macro0" is displayed in the caption.
4. Click "SFC \(\rightarrow\) Insert Step-Transition".
\(\Rightarrow\) CODESYS inserts a step-transition combination.
5. Click "SFC \(\rightarrow\) Zoom out of Macro".
\(\Rightarrow\) The implementation section returns to the main diagram.

See also
- * Chapter 6.4.1.20.1.5.8.5 "SFC element 'Macro'" on page 2079
- * \(\#\) Chapter 6.4.1.21.3.12.18 "Command 'Insert Macro"' on page 2682
- \(\&\) Chapter 6.4.1.21.3.12.19 "Command 'Insert Macro After"' on page 2682

\section*{Adding an association}
1. Select the "Step2" step.
2. Click "SFC \(\rightarrow\) Insert Action Association".
\(\Rightarrow\) CODESYS inserts an association to the right of the "Step2" step.
3. Click in the left field of the association to select the qualifier.
\(\Rightarrow\) You can enter the qualifier manually or use the Input Assistant.... Select "P".
4. Click in the right field of the association to select the action.
\(\Rightarrow\) You can type the action or select it by using the Input Assistant \(\ldots\).


See also
- "1. IEC actions" on page 2075
- \# Chapter 6.4.1.20.1.5.4 "Qualifiers for Actions in SFC" on page 2065
- \(\#\) Chapter 6.4.1.21.3.12.14 "Command 'Insert Action Association"' on page 2680
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.12.15 "Command Insert Action Association After'" on page 2681

Using the library The library analyzation.library allows for the analyzation of expressions. It can be used,
'analyzation.library' for the analyzation of expressions
for example, in the SFC diagram to examine the result of the flag SFCError. This flag is used to monitor timeouts in the SFC diagram.
See also
- * Chapter 6.4.1.20.1.5.7 "Library "Analyzation"" on page 2072

See also
- \& Chapter 6.4.1.20.1.5.1 "SFC editor" on page 2062

\subsection*{6.4.1.9.6 Function block - Calling functions or methods with external implementation}

A runtime system can include the implementation of a function block, function, or method (for example, from a library). If you create a POU in your application with the same name by using the "External implementation" without an implementation, then you can execute the existing implementation. Please make sure that you declare local variables only in an external function block. External functions and methods must not contain local variables.

When the application is downloading, CODESYS searches for and links the associated implementation in the runtime system for each external POU.

Objects with the property "External implementation" are postfixed with (EXT) after the object name in the "Devices" or "POUs" view.

See also
- \# Chapter 6.4.1.21.4.11.5 "Dialog 'Properties' - 'Build"' on page 2755

\section*{Creating POUs \\ with external implementation}
1. Click "Project \(\rightarrow\) Add Object \(\rightarrow\) POU".
2. Activate "Function block", Method, or "Function" and specify the name of the associated implementation of the runtime system. Close the dialog box by clicking "Add".
\(\Rightarrow\) The runtime system POU is created in the "POUs" view. The name is postfixed with (EXT).
3. Right-click the POU and select "Properties".
\(\Rightarrow\) The dialog box opens.
4. Click the "Build" tab.
5. Select the "External implementation (Late link in the runtime system)" check box. \(\Rightarrow\) The POU is declared and you can implement a POU call.

\section*{Creating} methods with external implementation
1. Select a function block in the device tree or in the POUs view.
2. Select "Add Object \(\rightarrow\) Method" and type the name of the associated implementation of the runtime system. Click "Add" to close the dialog box.
\(\Rightarrow\) The method is created.
3. Right-click the method and select "Properties".
\(\Rightarrow\) The dialog box opens.
4. Click the "Build" tab.
5. Select the "External implementation (Late link in the runtime system)" check box.
\(\Rightarrow\) The method is declared and you can implement a method call. The method name is postfixed with (EXT) in the "Devices" or "POUs" view.

\subsection*{6.4.1.9.7 Using input assistance} CODESYS provides tools and features to help you code when creating programs.

Input assistant The input assistant provides all program elements that you can insert at the current cursor position. Open the "Input Assistant" dialog by clicking "Edit \(\rightarrow\) Input Assistant" or by pressing [F2].
See also
- «» "Dialog 'Input Assistant' - Tab 'Categories'" on page 2575
\begin{tabular}{ll} 
Dialog 'Auto & This dialog supports the declaration of variables. \\
Declare' & See also
\end{tabular}
- \({ }^{〔}\) Chapter 6.4.1.21.3.3.32 "Command 'Auto Declare'" on page 2572
"List components"

The "List components" function is an input tool in textual editors to help you input valid identifiers. Activate this function by clicking "Tools \(\rightarrow\) Options" and then the "SmartCoding" category. Select the "List components after typing a dot (.)" check box.
- If you type a dot (.) instead of a global variable, then a drop-down list opens with all available global variables. You insert the selected variable after the dot by double-clicking a variable in the drop-down list or by pressing [Enter].
- If you type a dot (.) instead of a global variable after a function block instance variable or a structure variable, then CODESYS opens a drop-down list with all global variables, all input and output variables for the function block, or all structure members.
You insert the selected variable after the dot by double-clicking a variable in the drop-down list or by pressing [Enter].
Note: When you also want to choose from the local variables of function block instances, select the "Show all instance variables in input assistant" option in the CODESYS options (SmartCoding category).
- If a component access (with a dot) for a drop-down list has already happened, then the last selected entry is preselected at the next component access.
- When you type any sequence of characters and then press [Ctrl]+[Space], a drop-down list opens with all available POUs and global variables. The first element in this list that starts with the sequence of characters is selected by default and you can insert it by double-clicking it or by pressing [Enter].
Matches with the entered character string are highlighted in yellow in the drop-down list. If the entered character string is changed, then the displayed drop-down list is refreshed.
- In the ST editor, you can filter the displayed drop-down list by scopes:

Depending on the displayed drop-down list, you can use the [Arrow right] and [Arrow left] keys to toggle between the following drop-down lists:
- "All items"
- "Keywords"
- "Global declarations"
- "Local declarations"
- CODESYS displays a tooltip if you type an opening parenthesis for a POU parameter when calling a function block, a method, or a function. This tooltip includes information about the parameters as they are declared in the POU. The tooltip remains open until you click to close it or you change the focus away from the current view. If you accidentally close the tooltip, then you can reopen it by pressing [Ctrl]+[Shift]+[Space].


You can use the pragma attribute 'hide' for excluding variables from the "List components" feature.
Examples Typing structure variables:

list
Calling a function block:
\[
\begin{aligned}
\text { erg: }:= & \text { fbinst ( } \\
& \left\lvert\, \begin{array}{lll}
\text { FUNCTION_BLOCK FB } \\
\text { VAR_INPUT } & \text { iVarin } & \text { INT } \\
\text { VAR_OUTPUT } & \text { iVarout } & \text { INT }
\end{array}\right.
\end{aligned}
\]

\section*{See also}
- \({ }^{4}\) Chapter 6.4.1.21.4.14.24 "Dialog 'Options' - 'SmartCoding'" on page 2796
- ⓨ Chapter 6.4.1.20.6.3.17 "Attribute 'hide'" on page 2286

Short form fea- The short form feature allows you to type abbreviated forms for variable declarations in the ture declaration editor and in textual editors where variables declarations are possible. Use this feature by pressing [Ctrl]+[Enter] to end a declaration line.
CODESYS supports the following short forms:
- All identifiers become variable identifiers except the last identifier of a line.
- The data type of the declaration is determined by the last identifier of the line. The following applies:
- B or BOOL yields BOOL
- I or INT yields INT
- R or BOOL yields BOOL
- \(\quad\) S or STRING yields STRING
- If a data type is not defined using this rule, then the data type is automatically BOOL, and the last identifier is not used as the data type (see Example 1).
- Depending on the type of declaration, every defined constant becomes an initialization or string length definition (see Example 2 and 3).
- An address, such as \%MD12, is automatically extended with the AT attribute (see Example 4).
- Any text after a semicolon (; ) is converted into a comment (see Example 3).
- All other characters in the line are ignored (see exclamation mark in Example 5).

\section*{Examples}
\begin{tabular}{|l|l|l|}
\hline Example & Short Form & Resulting declaration \\
\hline 1 & A & A: BOOL \\
\hline 2 & A B I 2 & A, B: INT \(:=2 ;\) \\
\hline 3 & ST S 2; A string & ST:STRING(2); (* A string *) \\
\hline 4 & X \%MD12 R 5 Real Number & \begin{tabular}{l} 
X AT \%MD12: REAL \(:=5.0 ;(* ~ R e a l ~\) \\
Number *)
\end{tabular} \\
\hline 5 & B \(!\) & BOOL \\
\hline
\end{tabular}

See also
- * Chapter 6.4.1.9.4 "Declaration of Variables" on page 1847

Smart tag func- Smart tags make it easier to write program code by suggest appropriate commands directly at tions the programming element. When you place the cursor over a programming element that has a smart tag function, the symbol appears. When you click the symbol, the commands that you can choose from are shown. Available smart tags:
- The smart tag function provides the "Declare Variable" command for undeclared variables in the implementation part of the ST editor.
See also
- \({ }^{\#}\) Chapter 6.4.1.21.3.3.32 "Command 'Auto Declare"' on page 2572

\subsection*{6.4.1.9.8 Using Pragmas}

Pragma in CODESYS

A pragma is a text in the source code of the application that is enclosed in curly brackets. Pragmas are used to insert special statements in the code, which the compiler can evaluate. This allows a pragma to influence the properties of one or more variables with respect to precompilation or compilation (code generation). Pragmas that the compiler does not recognize are passed over as a comment.
The statement string of a pragma can also extend over multiple lines. For more details about the syntax, see the descriptions of the individual CODESYS pragmas.
There are different pragmas for different purposes (example: initialization of a variable, monitoring of a variable, adding a variable to the symbol configuration, forcing the display of messages during the compilation process, and behavior of a variable under certain conditions).

\section*{NOTICE!}

Uppercase and lowercase characters have to be respected.

\section*{Examples}
```

{warning 'This is not allowed'}
{attribute 'obsolete' := 'datatype fb1 not valid!'}
{attribute 'Test':='TestValue1;
TestValue2;
TestValue3'}

```

\section*{Possible insertion positions}

\section*{NOTICE!}

Pragmas in CODESYS are not one-to-one implementations of C preprocessor directives. You have to position a pragma like an ordinary statement. You must not use a pragma within an expression.

A pragma that the CODESYS compiler should evaluate can be inserted at the following positions:
- In the declaration part of a POU:
- In the textual declaration editor, specify pragmas directly as line(s), either at the beginning of the POU or before a variable declaration.
- In the tabular editor, you specify pragmas that should be located before the first declaration line in the "Edit Declaration Part"/ "Attributes" dialog.
- In a global variable list
- In the implementation part of a POU:
- The pragma has to be at a "statement position", meaning at the beginning of a POU on a separate line, or after a "; " or END_IF, END_WHILE, etc.
- FBD/LD/IL editor: In networks of the FBD/LD/IL editor, you insert pragmas like a label by means of the "FBD/LD/IL \(\rightarrow\) Insert Label" command. Then, in the text field of the label with the corresponding pragma statement, replace the default text "Label:". To use a pragma in addition to a label, you specify the pragma first and then the label.
```

Incorrect and
correct posi-
tions for a con-
ditional
pragma
INCORRECT: {IF defined (abc)}
IF x = abc THEN
{ELSE}
IF x = 12 THEN
{END_IF}
y :=- {IF defined(cde)} 12; {ELSE} 13; {END_IF}
END_IF
CORRECT: {IF defined(abc)}
IF x = abc THEN
{IF defined(cde)}
y := 12;
{ELSE}
y := 13;
{END IF}
END \overline{IF}
{EL\overline{SE }}
IF x = 12 THEN
{IF defined(cde)}
y := 12;
{ELSE}
y := 13;
{END_IF}
END \overline{IF}
{END_IF}

```

In the "Properties" dialog ("Compile" category), you can specify "defines" that can be queried in pragmas.

Scope:
Depending on the type and contents of a pragma, it may influence the following:
- Subsequent declarations
- Exactly the next statement
- All subsequent statements until it is canceled by a corresponding pragma
- All subsequent statements until the same pragma is executed with other parameters or the end of the code is reached. In this context, "code" means the declaration part, implementation part, global variable list, and type declaration. Therefore, a pragma influences the entire object when the pragma is alone on the first line of the declaration part and is not superseded or canceled by another pragma.

Pragma catego- The CODESYS pragmas are divided into the following categories:
ries in
CODESYS
- Attribute pragmas (influence compiling and precompiling)
- Message pragmas (print user-defined messages when compiling)
- Conditional pragmas (influence code generation)
- User-defined pragmas

See also
- \({ }^{*}\), Chapter 6.4.1.9.4.3 "Using the 'Declare variable' dialog box" on page 1852
- \({ }^{*} \curvearrowright\) Chapter 6.4.1.20.6.3 "Attribute Pragmas" on page 2271
- \(\Leftrightarrow\) Chapter 6.4.1.20.6.2 "Message Pragmas" on page 2269
- \& Chapter 6.4.1.20.6.4 "Conditional Pragmas" on page 2318

\subsection*{6.4.1.9.9 Using Library POUs}

Libraries are collections of objects that you can link to your application. You can use the objects contained in libraries in exactly the same way as objects that you have defined in the project.
Libraries can contain the following objects:
- POUs (for example function blocks, or functions)
- Interfaces and their methods and attributes
- Data types (for example enumerations, structures, aliases, and unions)
- Global variables, constants, and parameter lists
- Text lists, image pools, visualizations, and visual elements
- External files (for example, documentation)
- Cam plate tables

Libraries in a project are managed in the Library Manager. You use the dialog of the library repository to perform the previous installation of the library on the system.


For "visibility" of library POUs and namespaces of libraries, see also the help page for the library properties.

\section*{See also}
- \({ }^{\circ}\) Chapter 6.4.1.21.3.15.3 "Command 'Properties'" on page 2714
- ※ Chapter 6.4.1.17 "Using Libraries" on page 2034
- \(y^{4}\) Chapter 6.4.1.21.3.15.3 "Command 'Properties"" on page 2714

Using library The following instructions describe the example of how to insert the counter POU CTUD from the POUs library Standard into your program.
1. Open a POU in the editor and place the cursor in the declaration part.
2. Specify the name for the function block instance, followed by a colon (example:
iCounter1:).
3. Press [F2] to open the Input Assistant.
4. In the category "Structured Types", select the CTUD function block from the Standard library (subfolder "Counter").
Select the "Insert with namespace prefix" option.
5. Click "OK" to exit the dialog.
\(\Rightarrow\) The function block is inserted with a namespace prefix into the declaration part: iCounter1:Standard.CTUD.

See also
- \(\Rightarrow\) Chapter 6.4.1.17 "Using Libraries" on page 2034

\subsection*{6.4.1.9.10}

\section*{Managing text in text lists}

\section*{General}

Text lists are used for preparing visualization texts is multiple languages. You can specify the texts in Unicode format so that all languages and characters are possible. You can export text lists and then translate the texts outside of the current project.

CODESYS differentiates between static text (managed in the "GlobalTextList" object) and dynamic text (managed in objects of type "TextList". Static texts exist in the visualization and can change only the displayed language while in runtime mode. The text ID stays the same. Dynamic texts can be controlled by means of an IEC variable that contains the text ID. In this way, you can display varying text in a visualization element in runtime mode. For example, you can configure a text field so that is shows an error text for an error number.

Both text list types include a table with text entries. An entry consists of an ID for identification, the output text, and its translation. In a text list or global text list, you can translate an output text in any number of languages. The translations are the basis for the language selection and the language switch in visualizations.

\section*{Adding a language and translating text}

Requirement: A project is open with a text list or global text.
1. Double-click an object of type "TextList" or "GlobalTextList" in the device tree or POUs view.
\(\Rightarrow\) The "Textlist" menu is shown in the menu bar and the text list opens in the editor.
2. Click "Textlist \(\rightarrow\) Add Language".
3. Specify a name for the language (example: en-US). Click "OK" to close the dialog.
\(\Rightarrow\) A column is displayed with the heading en-US.
4. Type in the translation of the source text into the column.

You can correct the name of a language in the table by means of the command
"Rename Language" in the context menu of the text list.

\section*{Exporting a text list}
\(\checkmark\) Requirement: A project is open with a text list or global text.
1. Double-click the object "GlobalTextList" or an object of type "TextList".
\(\Rightarrow\) The object opens.
2. Click "Textlist \(\rightarrow\) Import/Export Text Lists".
\(\Rightarrow\) The "Import/Export" dialog opens.
3. At "Choose export file", click for more (...) and select the directory and file name (example: Text_lists_exported).
4. Select the "Export" option.
5. Click "OK" to close the "Import/Export" dialog.
\(\Rightarrow\) CODESYS exports to a file the text list entries of all text lists of the project. The table contains a column with the text list names.

\section*{Example}

Contents of the
filext_lists_e
xpor̄̄ed
```

TextList Id Default en_US
Text_list_A A Information A Infrrmation A_en
Text_list_A B Information B: OK Informatiōn B_en: OK
Text_list_A C Information C Information C_en
Text_list_A D Information D Information D_en
Text_list_A E Information E Information E_en
Text_list_A F Information F Information F_en
AlarmGroup 2 Warning 2
AlarmGroup 1 Warning 1
GlobalTextList Information B Information B_en
GlobalTextList Information A Information A_en
GlobalTextList Switch Switch
GlobalTextList Counter: %i Counter : %i

```

\section*{Preparing the exported file for input assistance}
\(\checkmark \quad\) Requirement: A file is created (example : Text_lists_exported) by means of the command "Import/Export Text Lists". It contains the texts of the text lists of the project.
1. Click "Tools \(\rightarrow\) Options", "Visualization" category, "File Options" tab.
2. Click ... in "Text file for textual "List components" " and select a file (example: Text_lists_exported). Click "OK" to close the dialog.
\(\Rightarrow\) When you specify a static text in the "Texts" property for an element in a visualization, CODESYS offers the source text of the file as input assistance when typing in the first letter.

(1): "Texts", "Text"
«""List components"" on page 1886

\section*{Importing files with text list entries}

A file to be imported has the . csv format. The first line is a header (example: TextList Id Default en US). The other lines contain text list entries. You get this kind of file by exporting the text lists of the project to a file. There you can edit the text list entries and then import the file outside of CODESYS. When importing, CODESYS handles the text list entries differently for the GlobalTextList and for dynamic text lists.

GlobalTextList
- CODESYS does not create new text list entries for an unknown ID.
- CODESYS ignores changes that affect the ID or the source text.
- CODESYS accepts changes in the translations.

TextList
- For a new ID, CODESYS supplements the corresponding text list with a text list entry.
- For an existing ID that does not agree in the source text, the source text of the text list is overwritten with the source text of the file.
- CODESYS accepts changes in the translations.

Importing a file \(\checkmark\) Requirement: A project is open with a text list or global text.
1. Double-click the object "GlobalTextList" or an object of type "TextList". \(\Rightarrow\) The object opens.
2. Click "Textlist \(\rightarrow\) Import/Export Text Lists".
\(\Rightarrow\) The "Import/Export" dialog opens.
3. In the "Choose file to compare or to import" input field, click for more ( \(\ldots\) ) and select the directory and file (example: Text_lists_corrected.csv).
4. Select the "Import" option.
5. Click "OK" to close the dialog.
\(\Rightarrow\) CODESYS imports the text list entries of the file into the respective text lists.

\section*{Example}

Contents of the
file \(\begin{aligned} & \text { Text lists_c }\end{aligned}\)


These contents are applied to the text lists with the same name in the project.
\# Chapter 6.4.1.21.3.21.6 "Command 'Import/Export Text Lists"' on page 2729

Comparing text lists with a file and exporting differences
\(\checkmark\) Requirement: A project is open with a text list or global text.
1. Double-click the object "GlobalTextList" or an object of type "TextList".
\(\Rightarrow\) The object opens.
2. Click "Textlist \(\rightarrow\) Import/Export Text Lists" in the context menu. \(\Rightarrow\) The "Import/Export" dialog opens.
3. In the "Choose file to compare or to import" input field, click for more ( \(\ldots\) ) and select the directory and file name of the comparison file (example: Text_lists_corrected.csv).
4. For "Choose export file", click ... and select the directory and file that contains the comparison result.
5. Select the "Export only text differences" option.
6. Click "OK" to close the dialog.
\(\Rightarrow\) CODESYS reads the import file and compares the text list entries that have the same ID. If they do not agree, then CODESYS writes the text list entries of the text list to the export file.

For the global text list, CODESYS compares the translations of the same source texts. If they do not agree, then CODESYS writes the text list entries to the export file.

ヶ Chapter 6.4.1.21.3.21.1 "Command 'Add Language"' on page 2728
\({ }^{4}\) Chapter 6.4.1.21.3.21.6 "Command 'Import/Export Text Lists'" on page 2729
« Chapter 6.4.1.21.3.21.7 "Command 'Remove Language'" on page 2730
« Chapter 6.4.1.21.2.28 "Object 'Text List"' on page 2532

\section*{Managing static text in global text lists}

The global text list is the central location for texts that are displayed in the visualization.
When you configure a text for the first time in visualization element, CODESYS creates the global text list. CODESYS fills in the table as you create more texts. Therefore, the table includes all texts automatically that you create in the project visualizations. CODESYS assigns incremental IDs as integers, beginning at 0.
You can check, update, and compare the global text list with the static texts of the visualization. You cannot edit the source text or the ID directly in the table. However, you can replace a source text with another source text by creating and importing a replacement file. Menu commands are provided for this purpose.

Configuring vis- A text in a "GlobalTextList" can contain a format definition. ualization elements with static text
\(\checkmark\) Requirement: A project is open with a visualization. The "GlobalTextList" object contains the texts that are defined in the project visualizations.
1. Double-click the visualization.
\(\Rightarrow\) The editor opens.
2. Select an element with the "Text" property (example: "Text field").
3. Type in some text in the "Text" property (example: Static Information A).
\(\Rightarrow\) CODESYS adds the text to the global text list in the POU view.

\section*{Checking the global text list}
\(\checkmark\) Requirement: A project is open with a visualization. The "GlobalTextList" object contains the texts that are defined in the project visualizations.
1. Double-click the "GlobalTextList" object in the POUs tree.
\(\Rightarrow\) The table opens with the static texts.

\section*{2. Click "Text List \(\rightarrow\) Check Visualization Text IDs".}
\(\Rightarrow\) CODESYS reports when a source text of the text list does not match the static text that is identified by the ID. The source text in the global text list and the text in the visualization with the same ID do not match.

\section*{Updating IDs of the global text list}

\section*{Removing the global text list and creating current IDs again}

\section*{Removing IDs from the global text list}

\footnotetext{
Updating the global text list with a replacement file

Example (tab as separator character)
}
\(\checkmark\) Requirement: A project is open with a visualization. The "GlobalTextList" object contains the texts that are defined in the project visualizations.
1. Double-click the "GlobalTextList" object in the POUs tree.
\(\Rightarrow\) The list opens with the text list entries.
2. Click "Text List \(\rightarrow\) Update Visualization Text IDs".
\(\Rightarrow\) CODESYS adds text to the global text list when a text in the "Static Text" property does not match the source text in the project visualizations.
\(\checkmark\) Requirement: A project is open with a visualization. The "GlobalTextList" object contains the texts that are defined in the project visualizations.
1. Right-click the "GlobalTextList" object in the POUs tree and select the "Delete" command. \(\Rightarrow\) The object is removed.
2. Open a visualization.
3. Click "Visualization \(\rightarrow\) Create Global Text List".
\(\Rightarrow\) In the POU view, a new "GlobalTextList" object is created. The global text list contains the static text from the existing project visualizations.

Requirement: A project is open with a visualization
the texts that were defined in the project visualizations.
1. Double-click the "GlobalTextList" object in the POUs tree.
\(\Rightarrow\) The table opens with the texts.
2. Click "Text List \(\rightarrow\) Remove Unused Text List Entries".
\(\Rightarrow\) CODESYS removes the text list entries with IDs not referenced in the project visualizations.

A replacement file has the CSV format. The first row is a header: defaultold defaultnew REPLACE. The following rows contain the old source texts, the new source texts, and then the REPLACE command. Tabs, commas, and semicolons are permitted separators. A combination of separator characters in a file is not permitted.
```

defaultold defaultnew REPLACE
Information A Information A1 REPLACE

```

When you import a replacement file, CODESYS processes the replacement file row by row and performs the specified replacements in the "GlobalTextList". In addition, CODESYS replaces the previous text with the replacement text in the visualizations. If the replacement text already exists as static text, then CODESYS recognizes this and harmonizes the static text and leaves only one text list entry.
\(\checkmark \quad\) Requirement: A project is open with a text list or global text.
1. Double-click the "GlobalTextList" object.
\(\Rightarrow\) The object opens.
2. Click "Text List \(\rightarrow\) Import/Export Text Lists".
\(\Rightarrow\) The "Import/Export" dialog opens.
3. At the "Choose file to compare or to import" input field, click for more ( \(\ldots\) ) and select the directory and file (example: ReplaceGlobalTextList.csv).
4. Select the "Import replacement file" check box.
5. Click "OK" to close the dialog.
\(\Rightarrow\) The texts in the text lists and the visualizations are replaced.
\begin{tabular}{|c|c|c|c|}
\hline \multirow[t]{15}{*}{Example} & \multicolumn{3}{|l|}{The global text list contains the following source texts:} \\
\hline & GlobalTextList & Counter: \% & \\
\hline & GlobalTextList & Counter: \% & \\
\hline & GlobalTextList & Informatio & \\
\hline & GlobalTextList & Informatio & a \\
\hline & GlobalTextList & Informatio & \\
\hline & GlobalTextList & Switch & \\
\hline & \multicolumn{3}{|l|}{The replacement file contains the following replacements:} \\
\hline & defaultold & defaultnew & REPLACE \\
\hline & Counter: \%i & Counter2: \%i & REPLACE \\
\hline & Counter: \%i & Counter2: \%i & REPLACE \\
\hline & Information A & Information A2 & REPLACE \\
\hline & Information a & Information A2 & REPLACE \\
\hline & Information Aa & Information A2 & REPLACE \\
\hline & Switch & ch2 & EPLACE \\
\hline
\end{tabular}

CODESYS detects duplicate text list entries and removes them. Afterwards, the global text list contains the following source texts:
\begin{tabular}{|l|l|}
\hline 5 & Switch2 \\
\hline 4 & Counter2: \%i \\
\hline 3 & Information A2 \\
\hline
\end{tabular}

The texts in the visualization have been replaced.


See also
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.21.2 "Command 'Create Global Text List"' on page 2728
- \({ }^{\circ}\) ) Chapter 6.4.1.21.3.21.10 "Command 'Check Visualization Text IDs'" on page 2731
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.21.11 "Command 'Update Visualization Text IDs'" on page 2731
- \(\boldsymbol{y}^{\boldsymbol{y}}\) Chapter 6.4.1.21.2.11 "Object 'GlobalTextList'" on page 2465

\section*{Managing dynamic text in text lists}

You can create and translate texts in a text list for dynamic texts in order to show them dynamically in a visualization or in the alarm management. The object of type "Text list" can be located globally in the POUs view or below an application in the device tree. It contains a table with text list entries that you can edit and extend. A text list entry consists of an ID for identification, the output text, and its translation. You can add new text list entries to a text list. Menu commands are provided for this purpose.

\section*{Creating text lists for dynamic text output}

Requirement: A project is open with a visualization.
1. Select an application in the POUs view or device tree and click "Project \(\rightarrow\) Add Object".
2. Select "Text list".
3. Type a name (example: Textliste_A). Click "Add" to close the dialog.
\(\Rightarrow\) An object of type "Text list" is created.
4. Click below the "Default" column and open the input field. Type a text (example:

Information).
\(\Rightarrow\) The source text is created. It is used as a key in the table and as a source text for translations.
5. Type any string in the "ID" column (example: A).
\(\Rightarrow\) A text list entry is defined with source text and ID. If you configure the "Dynamic texts" property of an element in a visualization, then you can select the text list Textliste_A and assign the ID A.
6. Double-click in the blank line at the end of the table below "Default" and type in more text list entries.
\begin{tabular}{l|l|l|l}
\hline & 践 Textliste_A \(\mathbf{x}\) & & \\
\hline ID & Standard & en \\
\hline A & Information A & \\
\hline B & Information B & \\
\hline C & Information: OK & \\
& & &
\end{tabular}

\section*{Displaying text dynamically}

In a visualization, you can configure the dynamic output of texts that were created in a text list by configuring the "Dynamic texts" property of an element. You can directly assign a text list and an ID, as well as IEC variables, where you set the values programmatically.
\(\checkmark\) Requirement: A project with visualization is open and a text list is in the device tree.
1. Open the text list (example: Text_list_A).
2. Double-click the visualization.
\(\Rightarrow\) The editor opens.
3. Drag an element to the visualization (example: a "Text field").
4. Configure its "Dynamic texts" property by selecting one in the "Text list" property (example: 'Text_list_A') and add an ID from the text list into the "Text index" (example: 'A').
Pay attention to the single straight quotation marks. You can also assign an IEC variable of type STRING for the text list name and ID.
\(\Rightarrow\) The IEC variables allow for programmatic access to the texts of the text lists.
5. Build the application, download it to the controller, and start it.
\(\Rightarrow\) The visualization shows the text from the text list in the text field : Information A.

See also
- \({ }^{\text {• }} \boldsymbol{y}\) Chapter 6.4.1.21.2.28 "Object 'Text List"' on page 2532

\subsection*{6.4.1.9.11 Using image pools}

An image pool is a table of image files. CODESYS references image files for use in the project (for example, in a visualization) uniquely by the ID and name of the image pool. A project can include several image pools. You can create Image pools in the device tree below the application or in the POU pool. In a library project, you can use the object properties of an image pool to turn it into a symbol library for the visualization.

We recommend that you reduce the size of image files as much as possible integrating them. This will optimize the loading time of the visualization in every visualization type: TargetVisu, WebVisu and development system.

If you insert an image element into a visualization and enter an ID ("Static ID") in the element properties, then CODESYS automatically creates a global image pool. CODESYS uses the default name "GloballmagePool" for this.

Please note the following when the ID of an image file appears in several image pools.
- Search order: If you selected an image managed in "GloballmagePool", then you do not have to enter the name of the image pool. The search order for image files is as follows:
- 1. GloballmagePool
- 2. Image pools assigned to the currently active application
- 3. Image pools next to the GloballmagePool in the POU window
- 4. Image pools in libraries
- Unique access: You can reference a selected image directly and uniquely by appending the image ID to the name of the image pool in the following syntax "<pool name>. <image ID>.

See also
- \({ }^{4}\) Chapter 6.4.1.21.2.15 "Object 'Image Pool'" on page 2468
- 乡y Chapter 6.4.1.21.4.11.18 "Dialog 'Properties' - 'Image Pool"" on page 2764

\section*{Creating image pools}
1. Select the "Application" object in the device tree.

Click "Project \(\rightarrow\) Add Object \(\rightarrow\) Image Pool". \(\Rightarrow\) The "Add Image Pool" dialog box opens.
2. Type a name for the image pool (for example, "Images1") and click "Add".
\(\Rightarrow\) The image pool is added to the device tree.
3. Select the image pool object and open by choosing the command "Project \(\rightarrow\) Edit Object".
4. Double-click the field in the "ID" column and assign an appropriate ID (for example, "Icon1").
You can also add new images to the list by clicking "Imagepool \(\rightarrow\) Add Image File".
5. Double-click the field in the "File name" column. Click for more settings (... ).
\(\Rightarrow\) The "Select Image" dialog box opens.
6. Click for more settings (...) and select the image file.
\(\Rightarrow\) A thumbnail of the image file is displayed in the field of the column "Image". The name of the file is displayed in the field of the column "File name".

The image file can be references only by the name Images1.Icon1.
See also
- \({ }^{4}\) Chapter 6.4.1.21.3.16.1 "Command 'Insert Image"" on page 2717

Using image files in the 'Image' visualization element

When you insert an image element into a visualization, you can define the image type.
- Static image: Enter the image ID of the image file or the name of the image pool plus the image ID into the element configuration (property "Static ID"). Please note the comments for the search order and access.
- Dynamic image: Type the variable for defining the image file ID (for example, PLC_PRG.imagevar) in the element configuration ("Bitmap ID variable" property). You can exchange a dynamic element in online mode depending on a variable.
See also
- \({ }^{2} \gg\) Chapter 6.4.5.20.1.5 "Visualization Element 'Image"" on page 3038

\section*{Using image \\ files for the visualization background}

You can set an image in the background definition of a visualization. You can define the image by the name of the image pool plus the filename, as described above for a visualization element.
See also
- \(\&\) Chapter 6.4.5.21.2.10 "Command 'Background'" on page 3349

\subsection*{6.4.1.9.12 Programmatic Access to I/Os}

\section*{General}

CODESYS provides the following features for mapping project variables to input, output and memory addresses:
- Assignment of project variables to input, output and memory addresses in the "//O Mapping" tab of the device editor
- Programmed access to I/Os
- Variables configuration
- AT declaration

\section*{NOTICE!}

We recommend that you define the mapping of project variables to input, output and memory addresses in the "//O Mapping" of the editor of the respective device.

\section*{See also}
- \(y^{2}\) Chapter 6.4.1.8.2 "Configuring Devices and I/O Mapping" on page 1836

\section*{Variables configuration - VAR_CONFIG}

Use the variables configuration for mapping variables of functions blocks to the process map. For declarations in the function block, assign the variables to the device inputs/outputs without providing the full address. Later, the exact address is provided centrally for all function block instances of the application in a global variable list including VAR_CONFIG declarations. This global variables list with the VAR_CONFIG declarations is termed the "variables configuration".

\section*{NOTICE!}

For changes to variables that are assigned to I/O addresses, CODESYS displays them immediately In the process map. For changes to variables that are mapped by a variables configuration, CODESYS displays them not until the end of the responsible task.

\section*{Declaration of variables in functions blocks}

When declaring variables in a function block, declare the variables between the keywords VAR and END VAR and assign incomplete addresses to the variables. Mark these incomplete addresses with an asterisk (*).
Syntax:
<identifier> AT \%<I|Q>*:<data type>;
Example Define two local I/O variables: the input variable xLocIn and the output variable xLocOut.
```

FUNCTION_BLOCK locio
VAR
xLOcIn AT %I*: BOOL := TRUE;
xLocOut AT %Q*: BOOL;
END_VAR

```

Final definition of addresses in the variables configuration of the global variables list
In the global variables list that you use as the variables configuration, define the variable declarations with the absolute addresses between the keywords VAR_CONFIG and END_VAR.
You must declare the VAR_CONFIG variables with the complete instance path, separating the individual POU and instance name by a dot (.). The declaration must include an address whose class (input/output) agrees with the class of the incomplete address (\%|*, \%Q*) in the function block. The data type must also agree.
Syntax:
<instance variable path> AT \%<I|Q><location>: <data type>;
If the path instance does not exist, then an error is reported. CODESYS prints an error also if there is not an address configuration available for a variable that you declared with an incomplete address.

Example The locio function block in the example above is used in a program as follows:
```

PROGRAM PLC_PRG
VAR
locioVar1: locio;
locioVar2: locio;
END_VAR

```

A correct variables configuration in a global variable list could then look like this:
```

VAR_CONFIG
PLCC PRG.locioVar1.xLocIn AT %IX1.0 : BOOL;
PLC_PRG.locioVar1.xLocOut AT %QX0.0 : BOOL;
PLC_PRG.locioVar2.xLocIn AT %IX1.0 : BOOL;
PLC PRG.locioVar2.xLocOut AT %QX0.3 : BOOL;
END_VA\overline{R}

```

\section*{See also}
- \(乡\) Chapter 6.4.1.20.2.11 "Configuration variables - VAR_CONFIG" on page 2121
- \({ }^{\circ}>\) Chapter 6.4.1.9.12.3 "AT declaration" on page 1902
- \(\quad\) © Chapter 6.4.1.20.4.12 "Addresses" on page 2231

Creating a varia-
bles configuration

In the function block, assign variables to device I/Os with incomplete addresses and then create a variables configuration.

Requirement: You have a project open that includes a controller configuration with a field device. The project contains a program (e.g. PLC_PRG) and a function block (e.g. func1). The field device has inputs and outputs. The textual view is selected in the options for the declaration editor.
1. Double-click a function block in the device tree (e.g. func1).
\(\Rightarrow\) The function block editor opens.
2. Type the following between the keywords VAR and END_VAR: xLocIn AT \%I*: BOOL := TRUE; and XLocOut AT \%Q*:BOOL; in the next line.
\(\Rightarrow\) You have declared an input variable xLocIn and assigned it to the incomplete input address \(\% I^{*}\) of a field device. You have assigned the declared output variables have to the incomplete output address \(\% Q^{*}\).
3. Click the PLC_PRG object in the device tree and add the following to the declaration section of the program between VAR and END_VAR:
```

locioVar1: func;
locioVar2: func;

```
4. Right-click the "Application" object in the device tree and click "Add Object \(\rightarrow\) Global Variable List" and then click "Add" in the "Add Global Variable List" dialog box. \(\Rightarrow\) The global variables list is added to the device tree and opens in the editor.
5. Change the keyword VAR_GLOBAL to VAR_CONFIG.
6. Click "Declarations \(\rightarrow\) Add All Instance Paths".
\(\Rightarrow\) The following instance paths are added:
```

PLC_PRG.logioVar1.xLocIn AT %I*;
PLC_PRG.logioVar2.xLocIn AT %I*;
PLC_PRG.logioVar1.xLocOut AT %Q*;
PLC_PRG.logioVar2.xLocOut AT %Q*;

```
7. Now, replace the incomplete addresses \(\% I^{*}\) and \(\% Q^{*}\) with the absolute, complete addresses.

See also
- \({ }^{\text {® }}\) Chapter 6.4.1.21.3.18.4 "Command 'Add all instance paths'" on page 2720

\section*{AT declaration}

In the variables declaration, the code AT assigns a project variable to a specific input address, output address, or memory address of the PLC that is configured in the device tree. You can also define the assignment of variables to an address in the "I/O Mapping" dialog of the device in the PLC configuration.

\section*{Syntax}
```

<variable name> AT <address> : <data type>;
<address> : %<memory area prefix> ( <size prefix> )? <memory position>

```

The AT declaration allows you to give the address a meaningful name. You can make any necessary changes for the input or output signals at just one location, for example in the declaration.

\section*{Examples}
\begin{tabular}{|l|l|}
\hline VAR wInput AT \%IW0 : WORD; END_VAR & \begin{tabular}{l} 
Variable declaration with address information of an \\
input word
\end{tabular} \\
\hline VAR xActuator AT \%QW0 : BOOL; END_VAR & \begin{tabular}{l} 
Boolean variable declaration \\
Note: For Boolean variables, one byte is allocated \\
internally if a single bit address is not specified. A \\
change in the value of xActuator affects the range \\
from QX0.0 to QX0. 7.
\end{tabular} \\
\hline VAR xSensor AT IX7.5 : BOOL; END_VAR & \begin{tabular}{l} 
Boolean variable declaration with explicit specification \\
of a single bit address. On access, only the input bit \\
7.5 is read.
\end{tabular} \\
\hline VAR xSensor AT IX* : BOOL; END_VAR & \begin{tabular}{l} 
For the address specification, the placeholder * is \\
given instead of the memory position. The final \\
address specification is done in the variables configu- \\
ration. \\
Note: This is possible in function blocks.
\end{tabular} \\
\hline
\end{tabular}

If you assign a variable to an address, please note the following:
- You cannot write to variables that are placed at inputs. This will cause a compiler error.
- You can perform AT declarations only for local and global variables, not for input/output variables of POUs.
- Furthermore, AT declarations cannot be used in persistent variable lists.
- If you use AT declarations for structure components or function block variables, then all instances use the same memory. This is just like using static variables in classic programming languages, such as \(C\).
- The memory layout of structures also depends on the target system.

\section*{NOTICE!}

If you do not specify a single bit address explicitly, then Boolean variables are allocated byte-by-byte.

\section*{Example}
```

PROGRAM PLC_PRG
VAR
xVar AT %QWO : BOOL;
END_VAR
xVar := TRUE;

```

When the variable \(x\) Var is written, the output memory range from \(Q \times 0.0\) to \(Q \times 0.7\) is affected.

See also
- \({ }^{*} \boldsymbol{y}\) Chapter 6.4.1.9.12.2 "Variables configuration - VAR_CONFIG" on page 1901
- \({ }^{4}\) Chapter 6.4.1.20.2.11 "Configuration variables - VAR_CONFIG" on page 2121
- \# Chapter 6.4.1.20.4.12 "Addresses" on page 2231

\subsection*{6.4.1.9.13 \\ Checking Syntax and Analyzing Code}

General
CODESYS provides useful functions for detecting errors and assisting you while you create programs. The syntax check flags errors and prints them to the message view as early as the programming phase.
The static code analysis in CODESYS also assists you in complying with defined coding guidelines and detecting weak constructs.
See also
- «4 Chapter 6.4.1.9.13.2 "Checking Syntax" on page 1904
- \({ }^{\text {• }} \boldsymbol{y}\) Chapter 6.4.1.9.13.3 "Analyzing code statically" on page 1904

\section*{Checking Syntax}

When you input code, the precompile in CODESYS already runs some basic checks. Then, wavy underlines appear under buggy code in the editor and an error message is printed to the messages view.
CODESYS automatically generates the application code from the source code that was written in the development system. This is done automatically before downloading the application to the PLC. Before the application code is generated, a test is performed for checking the allocations, the data types, and the availability of libraries. Moreover, the memory addresses are allocated when the application code is generated. You can click "Build \(\rightarrow\) Generate Code" to execute this command explicitly, or press the [F11] key. This is useful for detecting any errors in your source code, even when the PLC is not connected yet.
CODESYS prints all errors and warnings to the "Build" category of the messages view. Doubleclicking the error message opens the respective POU in the editor with the buggy code marked. As an alternative, you can also jump to the buggy code by right-clicking the error message.
Note the settings for this in the CODESYS options.
See also
- \({ }^{4}\), Chapter 6.4.1.21.4.14.24 "Dialog 'Options' - 'SmartCoding'" on page 2796

\section*{Analyzing code statically}

You can subject your source code also to static analysis (lint) during the code generation. This determines whether or not your source code complies with the coding guidelines that you defined - according to the idea behind the lint analysis tool.
- You activate the rules to the checked in the "Project Settings" dialog in the "Static Analysis Light" category. The check itself is performed automatically each time code is generated, for example when you click "Build \(\rightarrow\) Generate Code". If divergence from the rules is determined, then it is reported as an error message in the "Build" category of the message view. The reported errors have the prefix \(S A<n u m b e r>\).

\section*{NOTICE!}

For static code analysis with "Static Analysis Light", only the application code of the project is checked. Libraries are excluded from the check.

GVL variables in the "POUs" view are not necessarily checked: If you have a project with several applications, then only the objects in the active application are checked. If you have only one application, then the objects in the common POU pool are also checked.

\begin{abstract}
"Static Analysis Light" includes only a reduced set of rules in the default development system. A larger set of rules, additional naming conventions, and metrics are available when you install the CODESYS Static Analysis add-on.
\end{abstract}

Deactivating lines of code in the implementations with pragmas from the static analysis
Excluding
implementation
code

By means of the pragma \{analysis . . . \}, you can mark code so that the specified rules are not checked. As a result, the marked lines of code are not subjected to static analysis. The marked code is ignored during the check.
Syntax:
\{analysis <sign><rule number>|, <other combinations of signs and rules, comma-separated>\}
-<rule number>: Deactivate the rule SA <rule number>.
-<rule number>: Activate the rule SA <rule number>.
\(\square\) Requirement: Rules are activated in the "Project Settings" dialog.
1. Add the pragma \{analysis -<number>\} above the line of code that contains code not to be checked first of all. For example, for the rule SA0024
\(\Rightarrow\) The line of code is the first line of the code snippet that is not checked with rule 24.
2. Add the pragma \{analysis -<number>\} below the line of code that contains code not to be checked first of all. For example, for the rule SA0024
\(\Rightarrow\) The line of code above is the last line of the code snippet that is not checked with rule 24.

\section*{Example: \\ Ignore untyped \\ literal}
```

{analysis -24}
nTest := 99;
iVar := INT\#2;
{analysis +24}

```

The rule "SA0024: Untyped literals only" is deactivated for two lines. An error is not issued although the code does not correct to: nTest := DINT\#99;

\section*{Example: Ignore several rules}
```

{analysis -10, -24, -18}
{analysis +10, +24, +18}

```
"SA0010: Arrays with only one component"
"SA0018: Unusual bit access"
"SA0024: Untyped literals only"

\footnotetext{
\(\stackrel{O}{\square}\)
However, you cannot deactivate the rule SA0004: "Multiple Write Access on Output" with a pragma.
}
```

Excluding pro- Syntax:
gramming
objects with
pragmas from
the static anal-
ysis
{attribute 'analysis' := '-<rule number>[,<other negative rule
numbers, comma-separated>]'}
When you insert the attribute pragma in the declaration part of a programming object, the specified rules are excluded for the entire programming object. If multiple rules are excluded, then the rules are each comma-separated with a dash and a number. A pragma statement for activation is not required.

```

\section*{Example}
```

{attribute 'analysis' := '-33, -31'}
TYPE LocalData :
STRUCT
iLocal : INT;
uiLocal : UINT;
udiLocal : UDINT;
END_STRUCT
END_TYPE

```

The rules SA0033 and SA0031 are ignored for the structure LocalData.
```

{attribute 'analysis' := '-100'}
big: ARRAY[1..10000] OF DWORD;

```

The rule SA0100 is ignored for the array big.

See also
- \(\boldsymbol{y}^{2}\) Chapter 6.4.1.21.4.12.9 "Dialog 'Project Settings' - 'Static Analysis Light'" on page 2773

\subsection*{6.4.1.9.14 Orientation and Navigation}
6.4.1.9.14.1 Using the cross-reference list to find occurrences..................... 1906
6.4.1.9.14.2 Finding declarations.................................................................... 1908
6.4.1.9.14.3 Setting and using bookmarks.................................................... 1908

Using the cross-reference list to find occurrences
The occurrences of symbols of a variable, a POU (program, function block, function), or a DUT can be displayed in a cross-reference list. Then you can jump from the list directly to the corresponding locations in the project.
There are two ways to search for occurrence locations of a symbol:
- Plain text search: You manually specify a text (symbol name, placeholder) in the "CrossReference list" view.
- Search for a specific declaration:
- In the "Cross-Reference List" view, you select the declaration from the input assistant.
- The focus is on a symbol name in the POU editor and you start the cross-reference search from the context menu.
- The focus is on a symbol name in the POU editor, the "Cross-Reference List" view is open, and the cross-reference search executes automatically.
- In the "Cross-Reference List" view which already lists occurrence locations for several declarations, you limit these results to a specific declaration.

\section*{Cross references with text search by symbol name}

Requirement: The "Cross-Reference List" view is open.
1. Specify a string in the field next to the name, for example the identifier of the variable for which you want to find the occurrence location in the project. Example: "iCounter".

For the text search, you can use the asterisk " \(\star\) " (for any number of characters) or the question mark "?" (for an exact number of characters) combined with a substring of a variable identifier.
Use the percent sign "\%" to search for IEC addresses. Examples: " \(\% \mathrm{MW} 8\) ", " \(\% \mathrm{M}^{*}\) ".
2. Click the \(\Rightarrow\) button to start a text search in the project.
\(\Rightarrow\) The view "Cross-Reference List" opens and displays the occurrence locations for the iCounter variable. The declaration parts are always displayed in the project with the occurrence location indented.
3. Double-click an occurrence location in the cross-reference list.
\(\Rightarrow\) The respective object opens in the editor with the marked occurrence location.

Requirement: A POU is open in the editor.
ences for a specific symbol declaration
From the POU editor, with a menu command
1. Set the cursor at the identifier of the symbol (variable, POU) in the declaration part or implementation part.
2. Click "Browse for Symbol \(\rightarrow\) Browse Cross-References" in the context menu or "Edit" menu.
\(\Rightarrow\) The "Cross-Reference List" view opens and shows the occurrence locations of the variables or POU.

If the "Cross-Reference List" view is already open, then you can also search the occurrence locations for a specific result as follows:
\(\triangleright \quad\) Select the "Automatically list selection in cross reference view" check box in "Tools \(\rightarrow\) Options" ("SmartCoding" category). Select the name of the symbol in the POU, or set the cursor in the name.
\(\Rightarrow\) Depending on the position of the selection or cursor, the cross-reference list automatically shows the occurrence locations for the respective symbol.

In the cross-reference list view, with input assistance

In the cross-reference list view, limited to a specific declaration
\(\triangleright \quad\) If multiple declarations for a symbol are listed in the "Cross-Reference List" view, for example after a text search, then you can reduce the display to one result: Select the line with the desired declaration and click the \(\square\) button or click "Limit Results to Current Declaration" in the context menu.

\footnotetext{
\(\Rightarrow\) the cross-reference list includes only the occurrence locations for the selected declaration.
}

See also
- «y Chapter 6.4.1.21.3.4.13 "Command 'Cross Reference List"' on page 2587
- \(\quad \geqslant\) Chapter 6.4.1.21.3.3.29 "Command 'Browse Cross References'" on page 2571
- \(\Leftrightarrow\) Chapter 6.4.1.21.4.14.24 "Dialog 'Options' - 'SmartCoding'" on page 2796

\section*{Finding declarations}

CODESYS provides the capability of searching the entire project for the definition location of a variable or function. The block that includes the definition opens in the editor with the marked declaration.

Finding the dec- Requirement: You have opened a POU in the editor. laration of a variable
1. Set the cursor at an identifier in the implementation section.
2. Click "Edit \(\rightarrow\) Browse \(\rightarrow\) Go to Definition".
\(\Rightarrow\) The POU with the declaration opens in the editor with the variable definition marked. If the definition is located in a compiled library, then the respective block opens in the library manager.

You can execute this command in both online and offline mode.

Examples The following block includes a function block definition (fbinst), a program call (prog_y()), and a function block call (fbinst. out):
```

VAR fbinst:fb1; ivar:INT; END_VAR prog_y(); ivar:=prog_y.y;
res1:=fbinst.out;

```

If the cursor is located at prog_y, then the command opens the program prog_y in the editor.
If the cursor is located at fbinst, then this command focuses in the declaration section at line fbinst:fb1;

If you set the cursor at out, then this command opens the function block fb1 in its editor.

\section*{See also}
- \(\boldsymbol{y}^{\text {• }}\) Chapter 6.4.1.21.3.3.37 "Command 'Go to Definition'" on page 2576

\section*{Setting and using bookmarks}

Bookmarks are used for easy navigation through long programs. You can use bookmarks in all implementation language editors, except SFC (sequential function chart). Commands help to navigate directly to the marked position in the program.

\footnotetext{
Setting and Requirement: The POU is open in the editor. deleting bookmarks
}
1. Set the cursor at any program line.
2. Click "Edit \(\rightarrow\) Bookmarks \(\rightarrow\) Toggle Bookmark".
\(\Rightarrow\) A bookmark is set at this position in the program. This is marked by the bookmark symbol \({ }^{\text {D }}\).
3. Set several bookmarks at different places in the program.
4. Set the cursor at a bookmarked program line.
5. Click "Edit \(\rightarrow\) Bookmarks \(\rightarrow\) Toggle Bookmark".
\(\Rightarrow\) The bookmark is removed. The bookmark symbol \(\boldsymbol{\square}\) is deleted.
As an alternative to this, you can delete one or more bookmarks in the "Bookmarks" view by clicking the \(\times\) button. For this purpose, the corresponding bookmarks have to be selected in the "Bookmarks" view.

Click "Edit \(\rightarrow\) Bookmarks \(\rightarrow\) Clear All Bookmarks (Active Editor)" to remove all bookmarks from the active POU.
n order to delete all bookmarks in a project, click "Clear All Bookmarks". However, for this command to be available, you first have to add it to a menu by means of the command "Tools \(\rightarrow\) Customize".

See also
- \({ }^{4}\) Chapter 6.4.1.21.3.3.22 "Command 'Toggle Bookmark"' on page 2569
- \(\quad\) y Chapter 6.4.1.21.3.3.27 "Command 'Clear All Bookmarks (Active Editor)"' on page 2571
- \({ }^{4}\), Chapter 6.4.1.21.3.3.28 "Command 'Clear All Bookmarks"" on page 2571

Jumping to bookmarks within a POU

Requirement: The POU is open in the editor. Multiple bookmarks are set.
1. Click "Edit \(\rightarrow\) Bookmarks \(\rightarrow\) Next Bookmark (Active Editor)".
\(\Rightarrow\) Depending on the current cursor position, the cursor jumps to the next bookmark (see below).
2. Click "Edit \(\rightarrow\) Bookmarks \(\rightarrow\) Previous Bookmark (Active Editor)".
\(\Rightarrow\) Depending on the current cursor position, the cursor jumps to the previous bookmark (see above).

See also
- \({ }^{4}\) Chapter 6.4.1.21.3.3.23 "Command 'Next Bookmark (Active Editor)"' on page 2570
- \({ }^{\Perp}\) Chapter 6.4.1.21.3.3.25 "Command 'Previous Bookmark (Active Editor)"' on page 2570

Jumping to bookmarks of different POUs in a project

A project is open with multiple POUs. Multiple bookmarks are set in different POUs.
1. Click "View \(\rightarrow\) Bookmarks".
\(\Rightarrow\) The "Bookmarks" view opens.
All bookmarks in the project are listed in a table in the view.
2. Click the \(\rightarrow\) "Next Bookmark" button.
\(\Rightarrow\) In the "Bookmarks" view, the bookmark in the row below the selected bookmark is selected.

The POU with the recently selected bookmark in the table opens in the editor and the row with the bookmark is selected in the POU.
3. As in step 2, you can click the "Previous Bookmark" button to jump to the bookmark in the project that is displayed in the row above it in the "Bookmarks" view.

See also
- \(\&\) Chapter 6.4.1.21.3.4.11 "Command 'Bookmarks'" on page 2585
- \({ }^{\text {• }}\), Chapter 6.4.1.21.3.3.26 "Command 'Previous Bookmark"' on page 2570
- \(\Rightarrow\) Chapter 6.4.1.21.3.3.24 "Command 'Next Bookmark"' on page 2570

\subsection*{6.4.1.9.15 Searching and replacing in the entire project}

In CODESYS you can search for strings in single objects or project-wide. If required, you can replace the string found.
1. Choose the command "Search" in the main menu "Edit \(\rightarrow\) Search Replace".
\(\Rightarrow\) The dialog "Find" opens.
2. Enter the string to be found in the field "Find what".
3. Activate the search options
4. Define the objects to be searched by choosing an entry from the combobox "Search".
5. Click on the button "Find Next".
\(\Rightarrow\) The first hit is displayed.
6. Click on the button "Replace" to replace the string found by a different one.
7. Click on the button "Find All" to get a list of all hits.

See also
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.3.2 "Command 'Find', 'Find in Project"' on page 2563
- \({ }^{\text {• }}\) Chapter 6.4.1.21.3.3.3 "Command 'Replace', 'Replace in Project'" on page 2564

\subsection*{6.4.1.9.16 Refactoring}

In general, refactoring is a technique for improving the design of existing software code without changing the way it functions.
In CODESYS, refactoring provides functions for renaming objects and variables and updating referenced pins. You can display all occurrences of renamed objects and variables and then rename them all at once or individually. In "Tools \(\rightarrow\) Options", you can also configure where CODESYS will prompt you for refactoring.

Renaming Requirement: A project is open that includes at least a function block "FB" and a global variable
global variables list. The global variable list "GVL" is open in the editor and contains a variable declaration (example: iGlobal). "FB" uses iGlobal.
Renaming
global variables
throughout the
project
1. Select the global variable name iGlobal.
2. Right-click the variable and click "Refactoring \(\rightarrow\) Rename iGlobal".
3. In the "Rename" dialog, type a name in the "New name" input field, for example iGlobaloK, and click "OK".
\(\Rightarrow\) The "Refactoring" dialog opens. In the device tree view on the left, the "GVL" and "FB" objects are highlighted in red and yellow. In the view on the right, "FB" in is open in its editor and iGlobal has already been renamed as iGlobaloK.
4. Click "OK".
\(\Rightarrow\) No global variable iGlobal is in your project. Now iGlobaloK is everywhere.
1. Select the global variable name iGlobal.
2. Right-click the variable and click "Refactoring \(\rightarrow\) Rename iGlobal".
3. In the "Rename" dialog, type a name in the "New name" input field, for example iGlobalTest, and click "OK".
\(\Rightarrow\) The "Refactoring" dialog opens. In the device tree view on the left, the "GVL" and "FB" objects are highlighted in red and yellow. In the window on the right, the function block "FB" is open in its editor. iGlobalTest is listed instead of iGlobal.
4. Right-click in the view on the right.
5. Click "Reject this Object" and click "OK".
\(\Rightarrow\) The global variable iGlobal is available in "FB" in your project. The variable iGlobalTest is now specified in the objects where the previous variable occurred.
The error message in the message view reports that the iGlobal identifier is not defined.

Adding and removing input variables

In the declaration part of blocks, you can add and delete input and output variables by using the refactoring commands. CODESYS performs updates at the occurrence locations and calling locations of the blocks. You can accept or reject these updates individually. The "Refactoring" dialog also opens for this purpose.
Requirement: The FCT (function type) POU is open in the editor. The function already contains the input variables input1, input2, and inputx. They are called in the PLC_PRG and POU programs.
1. Set the focus in the declaration part of the FCT function.
2. Click "Refactoring \(\rightarrow\) Add Variable".
\(\Rightarrow\) The default dialog opens for declaring variables.
3. Declare the variable input_3 with the scope of VAR_INPUT and data type INT. Click "OK" to close the dialog.
\(\Rightarrow\) The "Refactoring" dialog opens (see figure below). The affected locations are marked in yellow. (1)+(2)
4. In the upper right corner, select "Add inputs with placeholder text" from the drop-down list. (3).
5. In the left side of the window, click one of the highlighted objects (for example, PLC_PRG). Right-click and choose the "Accept Whole Project" command to add the new variable at the new location of use in FCT for the entire project.
\(\Rightarrow\) You see the change in the implementation part of PLC_PRG in the view on the right: The placeholder _REFACTOR_ appears at the location where the new variable was added.
6. Click "OK" to close the "Refactoring" dialog.
7. Click "Edit \(\rightarrow\) Find". Search the project for "_REFACTOR_" to check and edit the affected locations.
8. Note: As an alternative, you can insert the new variable with another initialization value without working with a placeholder first. In this case, in Step 4 you select "Add inputs with the following value" and type the value in the field on the right side of the drop-down list.


Example of a new variable with placeholder text in a CFC block:


Please note that you can also remove variables with refactoring.

Reordering vari- In the declaration part of function blocks, you can change the order of declarations by refacables in the dec- toring This is possible for declarations with scope VAR_INPUT, VAR_OUTPUT, or VAR_IN_OUT laration

Requirement: The declaration part of a POU is open and includes declarations, for example:
```

VAR_INPUT
invar2 : INT;
invar1 : INT;
in : DUT;
bvar : BOOL;
invar3 : INT;
END_VAR

```
1. Right-click in this declaration block to access the context menu.
2. Click "Refactoring \(\rightarrow\) Reorder Variables".
\(\Rightarrow\) The "Reorder" dialog opens with a list of VAR_INPUT variables.
3. Drag the "invar1 : INT;" entry to the position before the "invar2." entry.
\(\Rightarrow\) The invar1 declaration is now at the top position.
4. Click "OK" to close the dialog.
\(\Rightarrow\) The "Refactoring" dialog opens. The affected locations are marked in yellow (see figure above).
5. Click "OK" to accept the new order for the function block.

\section*{Changing a variable declaration and applying refactoring automatically}
\(\checkmark\) Requirement: Function block fb_A.
1. Click "Tools \(\rightarrow\) Options".
\(\Rightarrow\) The "Options" dialog opens.
2. Select the "Refactoring" category.
3. In "Auto-Declare", activate the options "On renaming variables" and "On adding or removing variables, or for changing the namespace".
4. Double-click the function block fb_A.
5. Select a variable in the declaration of \(f b\) _A, for example iA. As an alternative, you can set the cursor before or in the variable.
6. Specify "Edit \(\rightarrow\) Declare variable" ([Shift]+[F2]).
\(\Rightarrow\) The "Declare Variable" dialog opens. The dialog includes the settings of iA.
7. Change the name of iA to iCounter_A.
8. The option "Changes by means of refactoring" appears and is activated.
9. Click "OK".
\(\Rightarrow\) The dialog "Refactoring" "Renaming from iA to iCounterA" opens. All locations affected by the variable renaming are marked there.
10. Click "OK" to close the dialog.
\(\Rightarrow\) The changes are applied.

See also
- \({ }^{\text {n }}\) Chapter 6.4.1.21.3.3.40 "Command 'Refactoring' - 'Rename <...>"' on page 2577
- \({ }^{\text {® }}\) Chapter 6.4.1.21.3.3.41 "Command 'Refactoring' - 'Update Referenced Pins'" on page 2578
- \(\quad y^{4}\) Chapter 6.4.1.21.3.3.42 "Command 'Refactoring' - 'Add Variable"' on page 2578
- \(y^{\prime y}\) Chapter 6.4.1.21.3.3.43 "Command 'Refactoring' - 'Remove <variable>"' on page 2580

\subsection*{6.4.1.9.17 \\ Task Configuration}

General
In the task configuration, you define one or more tasks for controlling and executing the application program in the controller. Each application must include a "Task Configuration" object.

A task is a time-based flow unit of an IEC program. You define a task with a name, a priority, and a type, which determines which condition triggers the start of the task. You can define this condition either by time (cyclic-interval, freewheeling) or by the occurrence of an internal or external event to process the task. Examples of an event are the rising edge of a global project variable or an interrupt event of the controller.

A task calls one or more program blocks (POUs). These programs can be application-specific (objects below the application in the device tree) or project-specific (objects available in the POU window). In the case of a project-specific program, the application instances the projectglobal program. If CODESYS processes the task in the current cycle, then the programs are executed for the duration of a cycle.

With the combination of priority and condition, you define the order in which the tasks are processed. You can configure a watchdog for each task, and you can link a start, stop, and reset directly to the execution of the project block.

Rules for the processing order of the defined tasks:
- If the task condition is satisfied, then CODESYS processes the task.
- If several tasks satisfy the condition for processing at the same time, then CODESYS processes the tasks with the highest priority first.
- If several tasks with the same priority level satisfy the condition for processing at the same time, then CODESYS processes the longest waiting task first.
- The program calls are processed in the order they appear in the configuration dialog of the task.
- If a called program has the same name in the device tree of the application and in a library or project-global in the POU window, then the application program is used.

\section*{Attention}

All tasks share one process map. The reason is as follows: When each task has its own individual process map, performance is compromised. However, the process map can be consistent only with one task. When you create a project, you must ensure that the application copies the input and output data to a safe location in case of conflicts. Modules, such as the library SysSem, provide the capability of solving consistency and synchronization problems.

Consistency problems can also occur when accessing other global objects, such as global variables or blocks. Consistency problems always occur if several tasks read and write to one variable. Modules, such as the library sysSem, are available as a solution.

\section*{Creating a task configuration}

Requirement: The open project includes a program-type POU and a "Task Configuration" with a
"Task" object has been inserted below "Application" in the device tree.
1. Double-click the task object below "Task Configuration" in the device tree.
\(\Rightarrow\) The "Configuration" tab of the task object opens.
2. In the "Type" dropdown list., click "Cyclic". \(\Rightarrow\) The "Interval (e.g. t\#200ms)" input field appears.
3. Enter t\# 300 ms in the "Interval (e.g. \(\# \# 200 \mathrm{~ms}\) )" input field.
4. Click "Add Call".
\(\Rightarrow\) The Input Assistant opens.
5. In "Input Assistant \(\rightarrow\) Categories" -> "Programs", click the desired POU and then click "OK".
\(\Rightarrow\) CODESYS inserts the selected POU into the POU list of the "Configuration" tab and below the task object in the device tree.

When the application is executed from the controller, CODESYS executes the selected POU in cyclical intervals of 300 ms .
- « Chapter 6.4.1.21.2.30.2 "Tab 'Configuration"' on page 2538

\section*{Definitions of Jitter and Latency}

In the "Task Configuration" object, on the "Watchdog" tab, you can monitor the periodic jitter values of the individual tasks at runtime. The periodic jitter is differentiated from latency-based release jitter. See the following definitions:

\section*{Periodic jitter}

Latency Latency is the delay between the invocation of a task and the actual start of its release.

Release jitter The release jitter \(J_{r}\) is the difference between the maximum and the minimum latency ( L ) that has ever occurred.
\(J_{r}=L_{\text {max }}-L_{\text {min }}\)
In the case that \(L_{\text {max }}=L_{\text {min }}\), a release jitter \(J_{r}\) of 0 . results. This corresponds to a plain offset shift.

See also
- ② Chapter 6.4.1.9.17 "Task Configuration" on page 1914
- \(\xi^{*}\) Chapter 6.4.1.21.2.30.2 "Tab 'Configuration"" on page 2538
- \({ }^{4}\) Chapter 6.4.1.21.2.29.4 "Tab 'Monitor'" on page 2536

\subsection*{6.4.1.9.18 Protecting an application}

You achieve the know-how protection and copy protection of a boot application with the help of PLC -specific license management and its settings in the object properties of the application. The download code and the boot application can be encrypted and signed.

\section*{Encryption with Requirements: You have a project with an application that you want to download to the cona dongle troller as an encrypted boot application. A security key for license management is connected to your computer.}
1. Select the application in the device tree.
2. Select the "Properties" command in the context menu.
\[
\Rightarrow \text { The "Properties - <application name>" dialog opens. }
\]
3. Click the "Security" tab.
4. For "Encryption Technology", select the "Simple Encryption" option and type the "Product Code" that you received from the hardware manufacturer for the controller. Depending on the controller, it is protected either by a security key (firmcode is shown automatically) or by an integrated Wibu memory card for example.
5. Click "Online \(\rightarrow\) Login" and download the application.
\(\Rightarrow\) If the matching security key and/or valid license is available, then you can download the application to the controller. By default, a boot application is automatically created at this time in the controller directory. The default setting is defined in the application "Properties", in the "Boot Application" category.
6. Logout, change the application, and login again.
\(\Rightarrow\) You are prompted to perform an online change. The dialog provides the option of updating the boot application on the PLC. If the security key and license match, then you can log in. If not, then you receive a corresponding message.

Encrypting with Requirements: You have a project with an application that you want to download to the concertificates troller as an encrypted boot application. In the Windows Certificate Store of your computer, you have a certificate of this controller for encrypting the application. Note: In case you want to download the application to different controllers, you will need the appropriate certificate for each controller.
1. Select the application in the device tree.
2. Select the "Properties" command in the context menu.
\(\Rightarrow\) The "Properties - <application name>" dialog opens.
3. Click the "Security" tab.
4. On "Encryption Technology", select the "Encryption with certificates" option.
\(\Rightarrow\) The "Certificates" group is enabled.
5. If there are not any certificates listed in the table, then click the \({ }^{-3}\) button.
\(\Rightarrow\) The "Certificate Selection" dialog opens for selecting a certificate from the local Windows Certificate Store.
6. In the lower area, select a certificate and add it to the upper area by clicking the button, Click "OK" to confirm.
\(\Rightarrow\) The certificate is shown in the "Certificates" group of the "Encryption" dialog.
7. Select the certificate and click "Apply" or "OK".
\(\Rightarrow\) The certificate is now used to encrypt the application. It can only be transferred to the controller on computers that have an corresponding key installed in the Windows Certificate Store.

See also
- ๕ Chapter 6.4.1.19.4 "Security for the Runtime/PLC" on page 2041
- 4 Chapter 6.4.1.6 "Protecting and Saving Projects" on page 1819
- \({ }^{\text {• }}\) Chapter 6.4.1.6.8 "Encrypting Projects with Certificates" on page 1829
- \(\&\) Chapter 6.4.1.21.4.11.4 "Dialog: Properties: Security" on page 2754
1. Click in the status bar of CODESYS to open the "Security Screen" view. Then select a certificate with a private key for a user profile for the "Digital signature". The procedure is described in the instructions "Configuring a certificate for the digital signature in a user profile".
2. Double-click the certificate for the "Digital signature" in the "User" tab.
\(\Rightarrow\) The "Certificate" dialog opens.
3. On the "Details" tab, click "Copy to file".
\(\Rightarrow\) The "Certificate Export Wizard" starts.
4. In the "Export Private Key" prompt, select the "No, do not export the private key" option.
5. For "Export File Format", select the "DER encoded binary X. 509 (.CER)" option.
6. In the next step, select a file name and the location for the certificate.
7. After the last step "Finish", a message appears that the export was successful.
8. After successful export to CODESYS, open the device editor by double-clicking the controller in the device tree and selecting the "Files" tab for the file transfer.
9. Select the "Path" cert/import in the right side of the "Runtime" dialog.
10. On the left side of the dialog for "Host", select the path in the file system where you saved the exported certificate and selected the certificate.
11. Click \(\gg\).
\(\Rightarrow\) The certificate is copied to the cert/import folder.
12. Click the "PLC Shell" tab.
13. Type the command cert-import trusted <file name of the
certificate.cer> in the input line of the tab and press the [Enter] key. Note that the file name is specified with the extension.cer; otherwise the certificate is not imported successfully.
\(\Rightarrow\) The certificate is created on the controller under trusted. With this certificate, the controller can test the integrity of the boot application.
14. Open the "Security Screen" by double-clicking in the status bar.
15. If you want that downloads, online changes, and boot applications of your project are always encrypted, then select the "Enforce signing of downloads, online changes and boot applications" option in the "Security level" group on the "User" tab. To do this, the "Enforce encryption of downloads, online changes and boot applications" option also has to be selected.

See also
- \({ }^{4}\) Chapter 6.4.1.6.8 "Encrypting Projects with Certificates" on page 1829
- ※"Encryption, signature" on page 2039
- \({ }^{*}\) Chapter 6.4.1.21.3.4.18 "Command 'Security Screen"" on page 2592
- \({ }^{4}\) Chapter 6.4.1.21.4.11.4 "Dialog: Properties: Security" on page 2754

Encrypting the download, online change, and boot application

Requirement: The CODESYS Security Agent add-on product is installed.
The "Security Screen" view provides an additional tab: "Devices". This allows for the configuration of certificates for the encrypted communication with controllers. In this case, see the help for CODESYS Security Agent.

Alternatives:

If you the CODESYS Security Agent is not available to you, then you can proceed as follows by means of the PLC shell of the device editor:
In order to use certificates on the controller for the encryption of downloads, online changes, and boot applications, these certificates first have to be generated on the controller and loaded from the controller and installed in the Windows Certificate Store.
\(\square\) Requirement: You are connected to the controller.
1. Open the device editor by double-clicking the controller in the device tree, and select the "PLC Shell" tab.
\(\Rightarrow\) The tab appears with a blank display window. Below that is a command line.
2. Type ? in the command line and press the [Enter] key.
\(\Rightarrow\) All commands are listed in the display window.
3. Type the following command in the command line: cert-getapplist.
\(\Rightarrow\) All used certificates are listed with information about components and availability with certificates.
4. If no certificate is available for the CmpApp component, then type the command certgenselfsigned <Number of the Component in the applist>.
5. Click the "Log" tab and then the refresh button (*).
\(\Rightarrow\) The display shows whether or not the certificate was generated successfully.
6. Type in cert-getcertlist and press the [Enter] key.
\(\Rightarrow\) Your own certificates are listed that can be used for encryption. The information Number and Key usage (s) are useful in the next step.
Number: The number is specified as a parameter in the next step.
Key usage (s) : Data encryption means that this is a certificate of the controller for a download, online change, and boot application.
7. Export the required certificate by typing in the command cert-export own 0 and press the [Enter] key. 0 is the Number of the certificate with Key usage (s): Data encryption.
\(\Rightarrow\) The display shows that the certificate has been exported to a cert directory.
8. Click the "Files" tab of the device editor.
9. Click the refresh button (\%) in the right part of the dialog in "Runtime".
\(\Rightarrow\) The list of files and directories is refreshed.
10. Open the "cert" folder in the list and then the "export" subfolder.
11. In the left part of the dialog in "Host", open the directory where the certificate of the controller will be loaded.
12. In the right part of the dialog, select the certificate that you have exported and click \(\qquad\) \(\Rightarrow\) The certificate is copied to the selected directory.
13. In the file explorer, go to the directory where the certificate was copied and double-click the certificate.
\(\Rightarrow\) The "Certificate" dialog opens and shows the information about this certificate.
14. On the "General" tab, click "Install Certificate".
\(\Rightarrow\) The "Certificate Import Wizard" starts.
15. In the "Certificate Storage" dialog, for "Certificate Import Wizard", select the "Store all certificates in the following store" option and then select the "Controller Certificates" folder.
\(\Rightarrow\) The controller certificate is imported into the Windows Certificate Store in the "Controller Certificates" folder. Now the certificate is available for the encryption of boot applications, downloads, and online changes.
16. Open the "Security Screen" by double-clicking in the status bar.
17. If you want that downloads, online changes, and boot applications of your project are always encrypted, then select the "Enforce encryption of downloads, online changes and boot applications" option in the "Security level" group on the "User" tab.
18. Open the "Project" tab and double-click the application in the "Encryption of boot application, download and online change" area.
\(\Rightarrow\) The properties dialog of the application opens.
19. Click the "Encryption" tab, select "Encryption with certificates" in the "Encryption technology" list box, and click 葡.
If the "Enforce encryption of downloads, online changes and boot applications" option is selected in the "Security Screen", then "Encryption with certificates" is already selected.
20. In the "Certificate Selection" dialog, select the respective certificate from the "Controller Certificates" folder and click 諫.
21. Click "OK" to confirm the dialog.
\(\Rightarrow\) The certificate is displayed in the properties dialog.
22. Confirm the properties dialog of the application.
\(\Rightarrow\) The certificate is shown on the "Project" tab of the "Security Screen" in the "Encryption of boot application, download and online change" group.

The boot application, download, and online change are encrypted.

See also
- Help for the CODESYS Security Agent add-on product
- \({ }^{\bullet}\) ) Chapter 6.4.1.21.2.8.11 "Tab 'PLC Shell"' on page 2441
- \(\quad\) Chapter 6.4.1.21.3.4.18 "Command 'Security Screen'" on page 2592

Deleting a certif- Requirement: The CODESYS Security Agent add-on product is installed. A certificate with the icate for the encryption of boot application, download and, online change
information "Encrypted Application" is already installed on your computer.
1. In the "Security Screen" view, on the "Project" tab, in the bottom view, click the entry for the application.
\(\Rightarrow\) The "Properties" dialog for the application opens with the "Encryption" tab.
2. For "Encryption Technology", select "Encryption with certificates". In the "Certificates" group, click
3. In the "Certificate Selection" dialog, delete the certificate as described above.
4. Click "OK" to close the "Certificate Selection" dialog.
\(\Rightarrow\) The certificate is no longer displayed in the "Properties" dialog.

See also
- Help for the CODESYS Security Agent add-on product
- \(\quad>\) Chapter 6.4.1.21.3.4.18 "Command 'Security Screen'" on page 2592

\subsection*{6.4.1.9.19}

Data Persistence

\section*{General}

The lifespan of a variable and its data begins at the time when the variable is created and ends at the time when the variable is deleted and its memory is freed. The time when the variable is created, initialized, or instantiated depends on the declared scope. The time when the memory is freed usually depends on the scope as well. For example, the memory of global variables is freed by exiting the application.
They can retain data longer than usual. The following mechanisms are provided for this purpose.

Mechanisms for data retention
- (A): Persistent global variable list \(\mathrm{T}^{\text {- }}\) with the keyword PERSISTENT RETAIN Persistent variables retain their values when the application is reloaded. Moreover, the values are restored after a download, warm start, or cold start.
- (B): Retain variables with the keyword RETAIN

Retain variables retain their values after a warm start, but not after reloading the application, a download, or a cold start.
- (C): Variables of the Persistence Manager of the CODESYS Application Composer Variables of the Persistence Manager are stored in an external file.
- (D): Recipe variables

Recipe variables and their values are stored in a recipe file.
See also
- \(\quad \Rightarrow\) Chapter 6.4.1.9.19.4 "Retaining data with variables of the persistence manager" on page 1926
- \(\boldsymbol{y y}^{4}\) Chapter 6.4.1.9.19.3 "Preserving data with retain variables" on page 1925
- \(>\) Chapter 6.4.1.9.19.5 "Preserving data with recipes" on page 1926
- «y Chapter 6.4.1.20.2.13 "Persistent Variable - PERSISTENT" on page 2122
- \(\sum^{*}\) Chapter 6.4.1.20.2.14 "Retain Variable - RETAIN" on page 2124
- \(\Leftrightarrow\) Chapter 6.4.1.21.2.20 "Object 'Persistent variable list"' on page 2476

Which mechanism is suitable for which application? Some common use cases are considered in comparison the table. The specific examples refer to a building control system.

Table 384: Comparison of mechanisms and use cases
\begin{tabular}{|c|c|c|c|c|c|}
\hline & Uses case & (A) Persistent variables & (B) Retain variables & (C) Variables of the Persistence Manager & (D) Recipe variables \\
\hline 1 & \begin{tabular}{l}
The application must maintain device settings. \\
Example: After a power failure, the building control has to have information available about how long a window blind needs to be raised.
\end{tabular} & \begin{tabular}{l}
Suitable \({ }^{1}\) \\
Preferred use case \\
In this case, you can also use retain variables instead of persistent variables. This is advantageous for variables whose declaration is often changed.
\end{tabular} & \begin{tabular}{l}
Suitable \\
Preferred use case \\
Retain variables are an advantage when their declarations are changed often.
\end{tabular} & \begin{tabular}{l}
Suitable \({ }^{2}\) \\
This is advantageous for controllers that do not have any hardware support. Special functionalities make this possible, such as double file buffering.
\end{tabular} & Possible, but very complicated and therefore not recommended. \\
\hline 2 & The application must maintain values also after program changes or extensions. & & & & \\
\hline & \begin{tabular}{l}
2a: Rare extensions \\
Example: An application programmer extends the program with a new switch and installs a new light. The building control must still have saved values available until then.
\end{tabular} & \begin{tabular}{l}
Suitable \({ }^{1}\) \\
Preferred use case
\end{tabular} & Suitable & Suitable \({ }^{2}\) & Possible, but complicated. \\
\hline & \begin{tabular}{l}
2b: Unrestricted changes, including deleting or changing the data type of variables \\
The building control is running and is persistent. When an application programmer adds a new functionality to the controller and therefore adds another persistent variable to a function block, the values saved up to that point must be retained. For example, the program in an FB is extended with a variable that controls the automatic switching off of a previously uncontrolled lamp after a certain time. The building control must have the times of all controlled lamps available after the extension.
\end{tabular} & Not suitable & \begin{tabular}{l}
Suitable \\
Data from retain variables are preserved as far as possible after an online change.
\end{tabular} & \begin{tabular}{l}
Suitable as far as possible \({ }^{2}\) \\
Preferred use case
\end{tabular} & Possible if textual, but complicated \\
\hline & 2c: The application must maintain values after a download. & Suitable & Not suitable & Suitable & Suitable \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|}
\hline & Uses case & (A) Persistent variables & (B) Retain variables & (C) Variables of the Persistence Manager & (D) Recipe variables \\
\hline 3 & \begin{tabular}{l}
The application must be able to use different value sets. \\
Example: The operating settings for summer, winter, and holidays must be saved and imported when needed.
\end{tabular} & Not suitable & Not suitable & Not suitable & \begin{tabular}{l}
Suitable \\
Preferred use case
\end{tabular} \\
\hline 4 & \begin{tabular}{l}
The application must be able to use settings from another system. \\
It must be possible to transfer settings to another plant using similar variables.
\end{tabular} & Not suitable & Not suitable & Suitable \({ }^{2}\) & Suitable \({ }^{3}\) \\
\hline 5 & \begin{tabular}{l}
The application must provide human readable data. \\
The user must be able to read, compare, and edit the data.
\end{tabular} & Not suitable & Not suitable & Suitable \({ }^{2}\) & Suitable \({ }^{3}\) \\
\hline
\end{tabular}
\({ }^{1}\) Disadvantage: Only possible if the runtime system supports this mechanism and an NVRAM memory or UPS is available. Advantage: Speed; recommended application: 1 and 2a
\({ }^{2}\) Disadvantage: In the case of large variable sets (> 10000), long delays during initialization and shutdown are to be expected. Advantage: No special memory is required; value retention exists even in case of changes, extensions, or deletions.
\({ }^{3}\) Advantage: Editable remotely, transferable. Disadvantage: Complicated

Lifespan of variables when calling online commands
\(\left.\begin{array}{|l|c|c|c|}\hline \text { User input in the "Online" menu } & \begin{array}{c}\text { Variable with usual lifespan } \\ \text { Neither RETAIN nor PERSISTENT }\end{array} & \text { RETAIN } & \text { PERSISTENT } \\ \text { RETAIN PERSISTENT } \\ \text { PERSISTENT RETAIN }\end{array}\right]\)
\(x\) : The variable retains its value.
i : The variable is initialized.
\({ }^{1}\) Note: For the structure of persistent data, see the information in "Mechanism for downloading".
See also
- « "Mechanism for downloading" on page 1923
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.7.6 "Command 'Online Change"' on page 2629
- \({ }^{*}\) Chapter 6.4.1.21.3.7.12 "Command 'Reset Origin"" on page 2635
- «> Chapter 6.4.1.21.3.7.10 "Command 'Reset Cold'" on page 2633
- \(\quad\) Chapter 6.4.1.21.3.7.5 "Command 'Load'" on page 2628

Lifespan of vari- The values of ordinary variables lose their value and are reinitialized.
ables when
downloading a
boot project

The values of persistent variables are protected when:
- The structure of the persistent variable in memory matches the structure in the persistent data list.

The values of retain variables are protected when:
- The structure of the persistent variable in memory matches the structure in the persistent data list.
- The persistent variables match the application (GUID has to agree).

A "Retain mismatch" occurs when the requirements for restoring the values of retain variables and persistent variables are not met when the application is booted. The response to this discrepancy is described in the documentation of the hardware manufacturer.
Note: For the structure of persistent data, refer to the information in "Mechanism for downloading".
See also
- ② "Mechanism for downloading" on page 1923

\section*{Preserving data with persistent variables}

Persistent variables retain their values after reloading the application, and after a download, warm start, or cold start.
A special non-volatile memory area on the controller, for example as NVRAM or UPS, is required to extend the lifespan. Securing the data in such a memory does not require any additional time, which is an advantage over data retention with the Persistence Manager. If the controller does not provide hardware support, then the data is usually stored in a file. Then the data will be retained if you shut down the controller correctly. In the event of a power failure or a pulled plug, however, data will be lost.

Behavior Value retained for
- Uncontrolled exit
- Warm start by calling the "Reset Warm" command
- Cold start by calling the "Reset Cold" command
- Repeated download of the application

Reinitialization for

\section*{- Call of the "Reset Origin" command}

Therefore, persistent variables are reinitialized only if you reset the controller to the factory settings (for example, when you click "Online \(\rightarrow\) Reset Origin").
If, on the other hand, you download the application again, the persisted data is retained if possible. That depends on how profound the changes that led to the download were. Changing the application name always leads to a full reinitialization. Changes to the implementations never lead to a reinitialization: the data persistence is completely preserved. Changes to the declarations lead to an initialization of the new variables only if the existing variables are persistent, when you change the declarations so that the persistent variable list remains consistent. This is the case when you add a new variable or delete an existing one. Inconsistencies can occur if you edit and change the identifiers or data types of previously declared persistent variables.

\footnotetext{
Mechanism for downloading
}

Editing the variable list in the persistence editor causes the variable list to be edited automatically before it is saved, not to be saved as it is shown in the editor.

During post-processing, a variable that you have removed is replaced by a placeholder variable with the same memory requirement. As a result, the subsequent variables retain their addresses in the process image. Moreover, a variable you add is moved to the end of the list. Post-processing can neutralize changes that would lead to a loss of persistence. But you create gaps that use additional memory.
When downloading, the CRC value of the variable list and the length of the list (number of variables) are stored on the controller. When downloading again, the new test value is compared with the test value currently on the controller. Then the variable list is compared successively up to the specified length. If you have edited a declaration (for example, the name or data type), then the variable is reinitialized. Otherwise its value is retained. When the download is repeated, CODESYS checks whether the variable list declared in the persistence editor is still consistent with the variable list already on the controller.
The mechanism works well when the variables themselves are not modified significantly. Too extensive changes of the identifiers and the data types continue to lead to a reinitialization and the loss of persistence. If you anticipate frequent changes due to your application requirements, then this kind of a list is not recommended. Moreover, in an online change after a data type change, a persistent variable is less robust than a variable with a normal lifespan.
It is good practice to clear any gaps in the variable list after a while (command "Reorder List and Clear Gaps"). After cleaning, however, the list no longer matches the list on the controller and you have triggered an initialization of all persistent variables. The persistence of all variables is lost.

For versions before V3.5 SP1, changes in the persistence editor always lead to reinitialization.

\section*{Recovering data with the recipe manager}

To clean up the global persistent variable list without losing persistence, you can save the data in a recipe using the Recipe Manager. This creates a list for all variables of the persistent variable list in the recipe manager, and at the same time its current values are stored by the controller as a recipe. Then execute the command "Reorder List and Clear Gaps" and perform a download again. Now when you execute the command "Restore Values from Recipe", the values saved in the recipe are restored.

Changing an existing declaration in the persistent variable list
if you change the name or data type of a variable, this is interpreted as a new declaration and causes a re-initialization of the variables at the next online change or download. For complex data types, a change occurs when a new component is added, or when you change the type of a variable from INT to UINT in the depth of a used structure used, for example.

Basically, complex user-defined data types are not suitable for administration in a persistent variable list, because even small changes cause the variable to be initialized with all components.

Double allocation of memory in the case of instance paths

You can persist global variables or variables declared locally in a function block or program. To do this, add the keyword PERS ISTENT to the declaration. In addition, you insert the instance path to this variable in the persistent global variable list. To do this, execute the "Add All Instance Paths" command in the persistence editor.

Persistence is guaranteed by the following mechanism:
- The cyclic tasks in which the variable is accessed are determined.
- At the end of the first cyclic task (in each cycle), the variable is copied to the persistent global variable list.
- After restarting the controller, the value of the persistent variable is copied to the ordinary variable.

The disadvantage of this mechanism is that memory is allocated both at the place of declaration and at the place of the instance path. This persistent variable has a double memory allocation. Moreover, the data is copied to both places in each cycle. This can be time consuming, especially when large structured values are involved.

Memory location in the case of persistent function block instances

A function block instance is always stored completely in memory. This is necessary so that the same code can work on different instances. If only one variable in a function block is marked with PERSISTENT, then the function block instance is stored completely with all variables in remanent memory, although only the one variable is treated as persistent. However, non-volatile memory is not available to the same extent as main memory.

A function block with a pointer to an instance in SRAM as a variable is not stored in the protected memory.

Importing from
CoDeSys V2.3 projects

When you open a CoDeSys V2.3 project to import it into CODESYS V3, the declarations of persistent variables are not preserved. You have to revise the declarations and create then again in a separate persistent global variable list.

See also
- ② Chapter 6.4.1.20.2.13 "Persistent Variable - PERSISTENT" on page 2122
- \({ }^{*}\) - Chapter 6.4.1.21.3.18.4 "Command 'Add all instance paths'" on page 2720
- \(y^{*}\) Chapter 6.4.1.3.3 "Opening a V2.3 project" on page 1809

\section*{Preserving data with retain variables}

Retain variables preserve their values after a warm start. However, the degree of value retention for persistent variables is higher.
A special non-volatile memory area on the controller, for example as NVRAM or UPS, is required to extend the lifespan. Securing the retain variables in such a memory does not require any additional time, which is an advantage over data retention with the Persistence Manager. If the controller does not provide hardware support, then the data is usually stored in a file. Then the data will be retained if you shut down the controller correctly. In the event of a power failure or a pulled plug, however, data will be lost.

Declaration To declare a retain variable, add the RETAIN keyword to a variable declaration.

Behavior Value retained for
- Uncontrolled exit
- Call of the "Reset Warm" command

Reinitialization for
- Repeated download of the application
- Call of the "Reset Cold" command (in contrast to persistent variables)
- Call of the "Reset Origin" command

When you restart an application, its variables are usually initialized with an explicitly preset initial value or with a default value. Variables marked with the RETAIN keyword are managed in a separate memory area depending on the target system and retain their value. Then the variables are protected from power failure, for example. This means that you can apply retain variables to a parts counter in a production line so that you can continue counting even after a power failure.

Memory loca- Function block instances are stored as one block in memory. This is necessary so that the same tion of persistent function block instances code can work on different instances. If a variable is marked with RETAIN in a function block, then each instance of the function block is protected with all variables. This is also true for the variables of the function block that are not marked this way. However, non-volatile memory is not available to the same extent as main memory.

A function block with a pointer to an instance in SRAM as a variable is not stored in the protected memory.

\section*{Importing of When you open a CoDeSys V2.3 project to import it into CODESYS V3, the declarations of CoDeSys V2.3 projects retain variables are preserved and remain effective as before.}

See also
- \({ }^{〔}\) Chapter 6.4.1.20.2.14 "Retain Variable - RETAIN" on page 2124
- \(\sum^{*}\) Chapter 6.4.1.3.3 "Opening a V2.3 project" on page 1809

\section*{Retaining data with variables of the persistence manager}

Persistent variables are managed in the Persistence Manager of the CODESYS Application Composer. The functionality of the "Persistence Manager" does not need any special memory on the controller in order to preserve values and data.

Declaration In the declarations, the variables managed in the Persistence Manager are marked with the pragma \{attribute 'ac_persist'\}.

The pragma makes sure that the variable with this attribute is managed in the Persistence Manager of the Application Composer. The variable value is retained even if you change the declaration of the variable, delete a variable from the application, or add a new one. The value is retained even if you change the data type and use the appropriate conversions.

Mechanism The variables of the Persistence Manager are stored with their values in an external archive file in TXT format.

The application code is extended with the code of the Persistence Manager, which leads to a greater memory requirement. This is at the expense of performance. Moreover, reading and especially writing a large number of persistent variables can take a long time. As a result, the executing task also blocks the execution for a long time.

Functionality - You can load and edit the TXT file in an external editor such as Notepad++.
- You can use the persistent variables of the file in another application.
- You can configure the behavior of persistent variables by defining persistence groups, assigning variables to them, and configuring the groups with their own save and read behavior.

\section*{Preserving data with recipes}

Variables are managed persistently in the Recipe Manager. The Recipe Manager does not need any special memory on the controller in order to preserve values and data.

Declaration A recipe definition consists of a set of variables with values and is created and edited in the "Recipe Manager" object and saved to a file.

Functionality - You can include a variable in multiple recipes, each with different values.
- In online mode, you can read in the actual values of the variables from the controller and save them as recipe values (specified value).
- You can use the Recipe Management library to programmatically implement the creation and editing of a recipe.
- You can save and backup a recipe as a recipe file.

See also
- \({ }^{4}\) Chapter 6.4.1.13.2.4 "Changing Values with Recipes" on page 2003
- ⓨ Chapter 6.4.1.21.3.18.2 "Command 'Save Current Values to Recipe"' on page 2719
- \({ }^{\mu}\) Chapter 6.4.1.21.3.18.1 "Command 'Reorder List and Clean Gaps'" on page 2719
- \({ }^{4} \boldsymbol{y}\) Chapter 6.4.1.21.3.18.3 "Command 'Restore Values from Recipe"" on page 2719
- \(\Leftrightarrow\) Chapter 6.4.1.21.2.25 "Object 'Recipe Manager'" on page 2519
- \({ }^{4}\) Chapter 6.4.1.21.2.26 "Object 'Recipe Definition'" on page 2522

\section*{Declaring VAR PERSISTENT Variables}

Below you will declare persistent variables in a persistent variable list and in a POU.
Requirement: A project is opened and contains a program POU. You have selected the option for the textual view in the "Declaration Editor" category of the options (menu command in "Tools \(\rightarrow\) Options").
1. Add the "Persistent Variables" object to the application object with the menu command "Project \(\rightarrow\) Add Object".
\(\Rightarrow\) CODESYS adds the persistent variable list T"PersistentVars" below the application object in the device tree and the editor opens.
2. In the editor, enter a variable declaration, for example ivarpersist1 : INT; between VAR_GLOBAL PERSISTENT RETAIN and END_VAR.
3. Double-click the POU in the device tree.
\(\Rightarrow\) The editor of the POU opens.
4. Specify the following declaration in the declaration part:

VAR PERSISTENT RETAIN
ivarpersist2: INT;
END_VAR
5. Click "Build \(\rightarrow\) Build".
\(\Rightarrow\) The message view opens. If CODESYS has compiled the application without errors, then close the message window and continue with the next step. Otherwise, correct the error(s) and select the menu command "Build \(\rightarrow\) Build" again.
6. Set the focus in the "PersistentVars" editor. Click "Declarations \(\rightarrow\) Add All Instance Paths"
\(\Rightarrow\) CODESYS adds the persistent variable from the persistent variable list "PersistentVars" to the POU:
// instance path of the persistent variables created POU.IVARPERSIST2 : INT

See also
- ② Chapter 6.4.1.20.2.13 "Persistent Variable - PERSISTENT" on page 2122
- \({ }^{4}\), Chapter 6.4.1.20.2.14 "Retain Variable - RETAIN" on page 2124
- ※y Chapter 6.4.1.21.2.20 "Object 'Persistent variable list"' on page 2476
- \(\xi^{\circ}\) Chapter 6.4.1.21.3.18.4 "Command 'Add all instance paths'" on page 2720

\section*{Saving the values of a persistent variable list in a recipe}

Requirement: a project is opened and a persistent variable list with declarations of persistent variables exists under an application object.
1. Double-click on the controller in the device tree and select the tab "Communication Settings".
2. Select your gateway and click on the button "Scan Network".
\(\Rightarrow\) Your device is shown in bold in the tree view of the gateway.
3. Select your device and click on the button "Set Active Path".
4. Select your application object in the device tree and select the context menu command "Set Active Application".
\(\Rightarrow\) The application object is displayed in bold.
5. Select the menu command "Online \(\rightarrow\) Login"
\(\Rightarrow\) Your application is logged in to the controller and the controller and the application object in the device tree have a green background.
6. Double-click on the persistent variable list and select the command "Declarations \(\rightarrow\) Save Current Values to Recipe".
\(\Rightarrow\) CODESYS creates the objects "Recipe Manager" and "PersistentVariables" under the application object.
7. Select the menu command "Online \(\rightarrow\) Logout".
\(\Rightarrow\) The application is logged out from the controller.

See also
- ⓨ Chapter 6.4.1.21.3.18.2 "Command 'Save Current Values to Recipe"' on page 2719
- \(y^{*}\) Chapter 6.4.1.9.19 "Data Persistence" on page 1920

\subsection*{6.4.1.9.20 Alarm Management}

For information about alarm management and alarm visualization, see the help for CODESYS Visualization.

\subsection*{6.4.1.9.21 Using POUs for implicit checks}

CODESYS provides special POUs that implement implicit monitoring functions. At runtime, these functions check the array limits or subrange types, the validity of pointer addresses, or division by zero.
1. Select the "Application" object in the device tree.

Click "Project \(\rightarrow\) Add Object \(\rightarrow\) POU for Implicit Checks"
\(\Rightarrow\) The "Add POU for Implicit Checks" dialog box opens.
2. Select the desired functions.
3. Click "Add".
\(\Rightarrow\) The selected POUs are inserted below the "Application" in the device tree.
4. Open the POUs in the editor.
5. Adapt the implementation suggestion to your requirements.

\section*{CAUTION!}

To obtain the feature for monitoring functions, do not edit the declaration section. However, you are permitted to add local variables.

See also
- \({ }^{\Downarrow}\) Chapter 6.4.1.21.2.21 "Object 'POU"' on page 2477
- \({ }^{*}\) Chapter 6.4.1.21.2.22 "Object 'POUs for Implicit Checks"" on page 2500

\subsection*{6.4.1.9.22 Object-Oriented Programming}

\section*{General}

CODESYS supports object oriented programming with function blocks and for this purpose provides the following features and objects:
- Methods
- Interfaces
- Properties
- Inheritance
- Method call, virtual function call
- Definition of function blocks as extensions of other function blocks

\section*{Application example}

Basic information on dealing with object-oriented programming (OOP) with AC500 V3 PLCs is given in an application example.

\section*{Application example}

In CODESYS V3, keywords can be used in object-oriented programming (OOP), e.g. to improve the programming style, debugging, etc. The application note OOP keywords explains how these keywords are used.

\section*{See also}
- \({ }^{\mu}\) Chapter 6.4.1.21.2.21.5 "Object 'Interface'" on page 2484

\section*{Extension of function blocks}

The extension of a function block is based on the concept of inheritance in object-oriented programming. A derived function block thereby extends a basic function block and in doing so is given the properties of the basic function block in addition to its own properties.
The extension of a function block means:
- The inherited function block contains all data and methods that are defined by the basic function block. You can use an instance of the basic function block in every context in which CODESYS expects a function block of the type of the basic function block.
- The derived function block can overwrite the methods that you have defined in the base function block. This means that the inherited function block can define a method with the same name, the same inputs and the same output as is defined by the basic function block. Tip: You have the following support when overwriting methods, actions, attributes, and transitions that are inherited by the base block: When you insert a method, action, etc. below an inherited block, the "Add Object" dialog includes a combo box with a list of methods, actions, etc. used in the base block. You can accept these and adapt them accordingly.
- The derived function block may not contain function block variables with the same names as used by the basic function block. The compiler reports this as an error.
The only exception: If you have declared a variable in the basic function block as VAR_TEMP, then the inherited function block may define a variable with the same name. In this case, the inherited function block can no longer access the variable of the basic function block.
- You can directly address the variables and methods of the basic function block within the scope of the inherited function block by using the SUPER pointer.

\section*{NOTICE!}

Multiple inheritance is not permitted.
Exception: A function block can implement multiple interfaces, and an interface can extend other interfaces.

\section*{Extension of a basic function block by a new function block}

Requirement: the currently opened project possesses a basic function block, for example "POU_1(FB)", which is to be extended by a new function block.
1. Right-click the "Application" object in the device tree and select "Project \(\rightarrow\) Add Object \(\rightarrow\) POU".
\(\Rightarrow\) The "Add POU" dialog opens.
2. Type the name for the new POU in the "Name" input field, for example "POU_Ex".
3. Select "Function block".
4. Click "Advanced" and then the more button (…).
5. In the category "Function blocks" under "Application" in the input assistant, select the \(\operatorname{POU}(\mathrm{FB})\) that is to serve as the basic function block, for example POU_1, and click "OK".
6. As an option, you can select an "Access modifier" for the new function block from the drop-down list.
7. Select from the "Implementation language" combo box (example: "Structured text (ST)".
8. Click "Add".
\(\Rightarrow\) CODESYS adds the POU_Ex function block to the device tree and opens the editor. The first line contains the text:
FUNCTION_BLOCK POU_Ex EXTENDS POU_1
The function block POU_Ex extends the basic function block POU_1.

Extension of a basic function block by an existing function block

Requirement: The open project possesses a base function block (example: POU_1 (FB) ) and another function block (example: POU_Ex (FB) ). The function block POU_Ex (FB) is also to be given the properties of the basic function block. This means that POU_EX (FB) should extend POU_1(FB).
1. Double-click the function block POU_Ex (FB) in the device tree. \(\Rightarrow\) The function block editor opens.
2. Extend the existing entry in the top line FUNCTION_BLOCK POU_Ex with EXTENDS POU_1.
\(\Rightarrow\) The function block POU_Ex extends the basic function block POU_1.

See also
- 党 Chapter 6.4.1.9.22.3 "Implementing interfaces" on page 1931
- \& Chapter 6.4.1.9.22.4 "Extending interfaces" on page 1933
- \({ }^{4}\) Chapter 6.4.1.20.2.15 "SUPER" on page 2125
- \({ }^{4}\) Chapter 6.4.1.20.2.16 "THIS" on page 2126
- \({ }^{4}\) Chapter 6.4.1.21.2.21.3 "Object 'Function Block'" on page 2479
- \(\quad y^{\prime}\) Chapter 6.4.1.21.2.21.9 "Object 'Property'" on page 2493
- \(\Leftrightarrow\) Chapter 6.4.1.21.2.21.10 "Object 'Action"' on page 2497
- \(\quad\) Chapter 6.4.1.21.2.21.11 "Object 'Transition'" on page 2499

\section*{Implementing interfaces}

Implementing interfaces is based on the concept of object-oriented programming. With common interfaces, you can use different but similar function blocks the same way.
A function block that implements an interface has to include all methods and attributes that are defined in that interface (interface methods and interface attributes). This means that the name and the inputs and outputs of the methods or attributes must be exactly the same. When you create a new function block that implements an interface, CODESYS adds all methods and attributes of the interface automatically to the tree below the new function block.

\section*{NOTICE!}

If you add more interface methods afterwards, then CODESYS does not add these methods automatically to the affected function block. To perform this update, you must execute the "Implement Interfaces" command explicitly.
For inherited function blocks, you have to make sure that any methods or attributes that were derived through the inheritance of an interface also receive the appropriate implementation. Otherwise they should be deleted in case the implementation that was provided in the basis should be used. Respective compile error messages or warnings are displayed, prompted automatically by added pragma attributes. For more information, refer to the help page for the "Implementing Interfaces" command.

\section*{NOTICE!}
- You must assign the interface of a function block to a variable of the interface type before a method can be called via the variable.
- A variable of the interface type is always a reference of the assigned function block instance.

A variable of the interface type is a reference to instances of function blocks. This kind of variable can refer to every function block that implements the interface. If there is no assignment to a variable, then the variable in online mode contains the value 0 .

\section*{Examples The I1 interface contains the GetName method.}
```

METHOD GetName : STRING

```

The functions blocks \(A\) and \(B\) implements the interface I1:
```

FUNCTION_BLOCK A IMPLEMENTS I1
FUNCTION_BLOCK B IMPLEMENTS II

```

For this reason, both function blocks must include a method named GetName and the return type STRING. Otherwise the compiler reports an error.

A function includes the declaration of a variable of interface I1 type.
```

FUNCTION DeliverName : STRING
VAR_INPUT
l_i : I1;
END_VA\overline{R}

```

Function blocks that implement the I1 interface can be assigned to these input variables.
Examples of function calls:
```

DeliverName(l_i := A_instance); // call with instance of type A
DeliverName(l_i := B_instance); // call with instance of type B

```

Calling of interface methods:
In this case, it depends on the actual type of \(1_{\text {_i }}\) whether the application calls A. GetName or B. GetName.
```

DeliverName := l_i.GetName();

```
- \(\quad\) Chapter 6.4.1.21.3.23.2 "Command 'Implement Interfaces'" on page 2744

\section*{Implementing an interface in a new function block}
\(\checkmark\) Requirement: The open project has at least one interface object.
1. Right-click "Application" in the device tree and select "Project \(\rightarrow\) Add Object \(\rightarrow\) POU". \(\Rightarrow\) The "Add POU" dialog box opens.
2. Type the name for the new POU in the "Name" input field, for example "POU_Im".
3. Select "Function block".
4. Click "Implemented" and then the more button (…).
5. In the input assistant, select the interface from the category "Interfaces", for example ITF1, and click on "OK".
6. To insert more interfaces, click ... and select a another interface.
7. As an option, you can select an "Access modifier" for the new function block from the selection list.
8. Select from the "Implementation language" combo box (example: "Structured text (ST)".
9. Click "Add".
\(\Rightarrow\) CODESYS adds the "POU_Ex" function block to the device tree and opens the editor. The first line contains the text:
FUNCTION_BLOCK POU_Im IMPLEMENTS ITF1
The interface and its methods and properties are now inserted below the function block in the device tree. Now you can type program code into the implementation part of the interface and its methods.

\section*{Implementing an interface in an existing function block}
\(\checkmark\)
Requirement: The currently open project has a function block (example: "POU_Im") and at least one interface object (example: "ITF1").
1. Double-click the "POU_Ex(FB)"POU in the device tree.
\(\Rightarrow\) The POU editor opens.
2. Extend the existing entry in the uppermost line FUNCTION_BLOCK POU_Im with IMPLEMENTS ITF1
\(\Rightarrow\) The "POU_Im" function block implements the "ITF1" interface.

See also
- \(\Rightarrow\) Chapter 6.4.1.21.2.21.3 "Object 'Function Block'" on page 2479

\section*{Extending interfaces}

You can extend interfaces just like function blocks. The interface is then also given the interface methods and interface properties of the basic interface in addition to its own.

Creation of an interface that extends another interface.
1. Select the object "Application" in the device tree.
2. Select the command "Project \(\rightarrow\) Add Object \(\rightarrow\) Interface". \(\Rightarrow\) The dialog box "Add Interface" opens.
3. Enter a name for the new interface.
4. Activate the option "Extended" and click on the button ....
5. The input assistant opens.
6. From the category "Interfaces", select the interface that is to be extended by the new interface.
- \({ }^{*} \boldsymbol{y}\) Chapter 6.4.1.21.2.21.5 "Object 'Interface"' on page 2484

\section*{Calling methods}

To implement a method call, the actual parameters (arguments) are passed to the interface variables. As an alternative, the parameter names can be omitted.

Depending on the declared access modifier, a method can be called only within its own namespace (INTERNAL), only within its own programming module and its derivatives (PROTECTED), or only within its own programming module (PRIVATE). For PUBLIC, the method can be called from anywhere.
Within the implementation, a method can call itself recursively, either directly by means of the THIS pointer, or by means of a local variable for the assigned function block.

Method call as a Virtual function calls can occur due to inheritance. virtual function call

Virtual function calls enable one and the same call to call various methods in a program source code during the runtime.

In the following cases the method call is dynamically bound:
- You call a method via a pointer to a function block (for example pfub^. method). In this situation the pointer can point to instances of the type of the function block and to instances of all derived function blocks.
- You call the method of an interface variable (for example interface1.method). The interface can refer to all instances of function blocks that implement this interface.
- A method calls another method of the same function block. In this case the method can also call the method of a derived function block with the same name.
- The call of a method takes place by means of a reference to a function block. In this situation the reference can point to instances of the type of the function block and to instances of all derived function blocks.
- You assign VAR_IN_OUT variables of a basic function block type to an instance of a derived FB type.
In this situation the variable can point to instances of the type of the function block and to instances of all derived function blocks.

\section*{Example}
```

Overloading
methods
The function blocks fub1 and fub2 extend the function block fubbase and implement the interface interface1. The methods method1 and method2 exist.

```
```

PROGRAM PLC_PRG

```
PROGRAM PLC_PRG
VAR_INPUT
VAR_INPUT
    b}\mp@subsup{}{}{-}\mathrm{ : BOOL;
    b}\mp@subsup{}{}{-}\mathrm{ : BOOL;
END_VAR
END_VAR
VAR pInst : POINTER TO fubbase;
VAR pInst : POINTER TO fubbase;
    instBase : fubbase;
    instBase : fubbase;
    inst1 : fub1;
    inst1 : fub1;
    inst2 : fub2;
    inst2 : fub2;
    instRef : REFERENCE to fubbase;
    instRef : REFERENCE to fubbase;
END_VAR
END_VAR
IF b THEN
IF b THEN
    instRef REF= inst1; (* reference to fub1 *)
    instRef REF= inst1; (* reference to fub1 *)
    pInst := ADR(instBase);
    pInst := ADR(instBase);
ELSE
ELSE
    instRef REF= inst2; (* reference to fub2 *)
    instRef REF= inst2; (* reference to fub2 *)
    pInst := ADR(instl);
    pInst := ADR(instl);
END_IF
END_IF
pIns̄t^.method1(); (* If b is TRUE, fubbase.method1 will
pIns̄t^.method1(); (* If b is TRUE, fubbase.method1 will
be called, otherwise fub1.method1 is called *)
be called, otherwise fub1.method1 is called *)
instRef.method1(); (* If b ist TRUE, fub1.methodl will be
instRef.method1(); (* If b ist TRUE, fub1.methodl will be
called, otherwise fub2.method1 is called*)
called, otherwise fub2.method1 is called*)
On the assumption that fubbase in the above example contains two methods method1 and method2, it overwrites fub1 method2, but not method1. The call of method1 takes place as follows:
pInst^.method1();
If \(b\) is TRUE, then CODESYS calls fubbase.method1. If not, then \(f u b 1\). method 1 is called.
```

Additional out- In accordance with the IEC 61131-3 standard, methods can have additional outputs declared, puts like normal functions. With the method call, you assign variables to the additional outputs.
Detailed information about this can be found in the topic "Function".

Syntax for the
call:

```
<function block name>.<method name>(<first input name> := <value> (,
<further input assignments>)+ , <first output name> => <first output
variable name> (,<further output assignments>)+ );
```


## Example

Declaration

```
METHOD PUBLIC DoIt : BOOL
VAR_INPUT
    iInput_1 : DWORD;
    iInput_2 : DWORD;
END VAR
VAR_OUTPUT
    iOutput_1 : INT;
    sOutput_2 : STRING;
ENDVAR
```

Call

```
fbInstance.DoIt(iInput_1 := 1, iInput_2 := 2, iOutput_1 =>
iLocal_1, sOUtput_2 => sLocal_2);
```

When the method is called, the values of the method outputs are written to the locally declared output variables.

Calling a method even if the application is in the STOP state

In the device description it is possible to define that a certain function block instance (of a library function block) always calls a certain method in each task cycle. If the method contains the input parameters of the following example, CODESYS processes the method even if the active application is presently in the STOP state:

## Example

```
VAR_INPUT
    pTaskInfo : POINTER TO DWORD;
    pApplicationInfo: POINTER TO _IMPLICIT_APPLICATION_INFO;
END_VAR
(*Now the status of the application can be queried via
pApplicationInfo and the instructions can be implemented: *)
IF pApplicationInfo^.state = RUNNING THEN <instructions> END_IF;
```

Calling methods recursively

Use recursions mainly for processing recursive data types such as linked lists. Generally, we recommend that you be careful when using recursion. An unexpectedly deep recursion can lead to stack overflow and therefore to machine downtime.

Within their implementation, a method can call itself:

- Directly by means of the THIS pointer
- Indirectly by means of a local function block instance of the basic function block

Usually, a compiler warning is issued for such a recursive call. If the
method is provided with the pragma \{attribute 'estimated-stack-usage' :=
'<sstimated_stack_size_in_bytes>'\}, then the compiler warning is suppressed. For an implementation example, refer to the section "Attribute 'estimated-stack-usage'".

See also

- $\boldsymbol{y}^{*}$ Chapter 6.4.1.20.6.3.14 "Attribute 'estimated-stack-usage"" on page 2281
- ${ }^{\text {• }}$ Chapter 6.4.1.20.2.16 "THIS" on page 2126
- $\Leftrightarrow$ Chapter 6.4.1.20.2.15 "SUPER" on page 2125
- $\quad$ \& Chapter 6.4.1.21.2.21.9 "Object 'Property'" on page 2493
6.4.1.10

Working with Controller Networks
General
With the following functionalities, CODESYS supports communication between controllers (PLC) and the insertion of a safety controller below a PLC:

- Symbol Configuration: CODESYS creates symbols with certain access rights for the variables in an application. With these symbols, you can access the variables from outside, for example from an OPC server.
- Data Source Manager: Manages the connection settings and the data transmission to remote devices (data sources). The transmitted data is mapped in data source variables that are accessed in the visualization or local application. An example of this is a control panel that controls remote devices and displays the state of the device as an HMI application.
- Network Variables: Network variables are variables whose values are accessible to different controllers in the network. The variables have to be defined in rigid, identical lists in both the transmitter device and the receiver device. These lists are assigned to applications, but can be located in different projects.
- A safety controller can be inserted below a PLC in the device tree. The communication links of the safety controller to the field devices, controller networks, and development system are routed through this controller.


## The "DataServer" object is obsolete.

The data link with CODESYS DataServer has already been superseded with SP10 by a data link with data sources. With CODESYS 3.5 SP17, the functionality has now been completely removed.
In case you want to adapt an existing project with a "DataServer" object, you can do the following: Open the existing project with CODESYS V3.5 SP16, select the data server object, and click "Convert Data Server to Data Source Manager" in the context menu. After the conversion of the data link to a data source connection, you can open the project with a current CODESYS version.

See also

- ${ }^{〔}$ Chapter 6.4.1.10.3 "Symbol Configuration" on page 1941
- ${ }^{4}$ Chapter 6.4.1.10.6 "Data Link with Data Sources" on page 1947
- ${ }^{4}$ Chapter 6.4.1.10.5 "Network Variables" on page 1946
- ${ }^{2}>$ Chapter 6.4.1.10.2.2 "Network topology" on page 1937
- $\Leftrightarrow$ Chapter 6.4.1.10.2.3 "Addressing and Routing" on page 1937
- $\sum_{\text {• Chapter 6.4.1.10.2.4 "Address Structures" on page } 1939}$
- $\boldsymbol{y y}^{2}$ Chapter 6.4.1.10.7 "Subordinate safety controller" on page 1964


### 6.4.1.10.2 Network and Addressing

General
Constructing a control network hierarchically, so that extensive self-configuration is possible.
In CODESYS the network topology is mapped to clear addresses and the routing algorithm is kept simple by structured addresses. There is direct and relative addressing and automatic address determination during the bootup of the system.

See also

- « Chapter 6.4.1.10.2.2 "Network topology" on page 1937
- © Chapter 6.4.1.10.2.3 "Addressing and Routing" on page 1937
- \& Chapter 6.4.1.10.2.4 "Address Structures" on page 1939


## Network topology

Information and It is recommended to set up a network system so that the following are possible:
recommenda-
tions for the
topology of a
control network

- Extensive self-configuration (address assignment)
- Transparent support for every communication medium
- Transport of data packets between different networks

The routing mechanism should be so simple that each network node can reroute data packets, even if it has a low memory capacity. Therefore, avoid extensive routing tables, complex calculations or queries at runtime.

Construct the control network hierarchically. Each node may possess a parent node and any number of child nodes. A node without a parent is a "top level" node. Cycles are not permitted, i.e. each control network has the structure of a tree.

Parent-child relationships results from the specification of certain network areas. A network area can be, for example, a local Ethernet or a serial point-to-point connection. We differentiate between the main network (mainnet) and the subnetworks (subnet). Each node belongs at the most to one main network, to which its parent node, if one exists, also belongs. For each node any desired number of subnets can be configured, for which the node acts in each case as a parent.
A network area may have only one parent node. Therefore, a configuration in which a network area is defined at the same time as a subnet of several nodes is invalid.
See also

- Ehapter 6.4.1.10.2.3 "Addressing and Routing" on page 1937
- $\&$ Chapter 6.4.1.10.2.4 "Address Structures" on page 1939


## Addressing and Routing

Addressing means: the topology of the control network is mapped to unique addresses.
A node address is composed hierarchically: for each network connection the associated block driver determines a local address, which uniquely identifies the node within the local network. The complete node address is formed as follows: The local address is placed in front of the subnet index of the local network assigned by the parent. In turn, the subnet index is placed in front of the node address of the parent. The length of the subnet index (in bits) is thereby determined by the device. The length of the local address, conversely, is determined by the type of network. A node without a main network is a top level node with address 0 . A node with a main network that contains no parent is likewise a top level node. It is given the local address of the main network.

See an example of a control network here:


In the example the addresses of the nodes are represented in hexadecimal notation. The first 4 digits represent the address of the respective parent in the main network, for example $0 \times 007 \mathrm{~A}=122$ for PLC1. The next byte (in blue lettering) is reserved for the subnet index and is followed by the local address, for example $\mathrm{C}=12$ for node ID 12. The structuring of the addresses makes a lean routing algorithm possible. Routing tables, for example, are thus unnecessary. Information is queried only locally: via its own address and via the address of the parent node. On this basis a node can correctly process the data packets:

- If the destination address corresponds to the address of the current node, then this is meant to be the receiver.
- If the destination address starts with the address of the current node, then the data packet is either meant directly for a child or for a descendant of the node and must be forwarded.
- In all other cases the receiver is not a descendant of the current node and the data packet must be forwarded to its own parent.
Relative addressing is a special case: relative addresses do not contain the node number of the receiver, but directly describe the path from the sender to the receiver. The principle is similar to the relative path in the file system: the address consists of the number of steps via which the packet must be transported upwards. These are the steps to the corresponding parent and from the subsequent path downwards to the destination node.
The advantage of relative addressing is that two nodes in the same subtree can continue to communicate if the complete subtree is shifted to another place in the entire network. Whereas the absolute node addressing has to be modified due to this shift, the relative addressing is still valid.


## Address determination

For a node to know its own address it must either know the address of its parent node or know that it is a top level node. For this purpose the node dispatches a message during the bootup to all network devices for address determination. As long as it receives no response to this message, the node considers itself to be a top level node, but continues to search for a possible parent. A parent node responds by announcing its address. The node will thus independently complete its address and will announce it to the subnets. An address determination can be accomplished during the bootup or at the request of the PC used for programming.

See also

- ${ }^{*}>$ Chapter 6.4.1.10.2.2 "Network topology" on page 1937
- $\quad>$ Chapter 6.4.1.10.2.4 "Address Structures" on page 1939


## Address Structures

Network addresses

Network addresses represent a mapping of the addresses of the network type (for example IP) to logical addresses within a control network. This mapping is carried out by the corresponding block driver. The first three bytes of the IP address are identical for all network devices within an Ethernet network with "Class C" IP addresses. Consequently, the last 8 bits of the IP address suffice as network address, since they enable unambiguous mapping between the two addresses on the block driver.
A node has a different network address for each network connection. Different network connections can have the same network address, since each address need only be locally unique.

Terminology: the network address in the main network is usually designated as the network address of a node with no specification of the network connection.

The length of a network address is specified in bits and can be chosen by the block driver as required. The same length must be used for all nodes within a network area. A network address is represented by an array of bytes in accordance with the following coding:

- Length of the network address: n bits
- Necessary bytes: $b=(n+7)$ DIV 8
- The (n MOD 8) bits of lowest rank of the first byte and all others (n DIV 8) are used for the network address.

| Example of Length: 11 bit <br> network  <br> address coding  | Address: 11110001100 |
| :--- | :--- |



Node addresses The node address indicates the absolute address of a node within a control network and is therefore unique within the whole "network tree". The address is composed of up to 15 address components, each of which occupies 2 bytes. The lower a node is located within the network hierarchy, the longer its address.
The complete node address consists of the partial addresses of all preceding nodes and the partial address of the node itself. Each partial address consists of one or more address components. The length is therefore always a multiple of 2 bytes. The partial address of a node is formed from the network address of the node in its main network and the subnet index of the main network in the case of the parent node. The bits required for the subindex are determined by the router of the parent node. Filler bits can be inserted between the subnet index and the network address in order to ensure that the length of the partial address is a multiple of 2 bytes.

Special cases:

- A node without a main network: this means that there is neither a subnet index nor a network address in the main network. In this case the address is set to 0x0000.
- A node in the main network without a parent: In this case a subnet index with the length 0 is assumed. The partial address corresponds to the network address, if necessary extended by filler bits.


The node address is always specified in hexadecimal. The individual address components (two bytes in each case) are separated by a colon ":". Since this represents an array of bytes and not a 16-bit value, the components are not displayed in the Intel format. For manually input addresses, missing parts in an address component are supplemented by leading zeros: "274" = "0274". In order to improve the legibility, the display should also always contain the leading zeros.

## Absolute and relative addresses

Communication between two nodes can be based on relative or absolute addresses. Absolute addresses are identical to node addresses. Relative addresses specify a path from the sender to the receiver. They consist of an address offset and the descending path to the receiver.
The (negative) address offset describes the number of address components by which a packet must be passed upwards in the tree before it can be passed back down by the common parent node. Since nodes can use partial addresses that consist of more than one component, the number of parent components to be passed is always equal to the address offset. This means that the demarcation between the parent nodes is no longer clear. For that reason the common start of the address of the communication partners is used as the parent address. Each address component is counted as an upward step, independent of the current parent node. Each error resulting from this assumption can be detected by the corresponding parent node and must be handled by it accordingly.

After achieving the common parent node the relative path, as an array of address components, is followed downwards as usual. Formal: the node address of the receiver is formed by removing the last address offset components from the node address of the sender and by appending the relative path to the remaining address.

## Example of the

 formation of node addressesIn the following example each address component is represented by a letter; in each case a dot separates nodes from each other. Since a node can carry several address components, there are some in the example that are represented with several letters.
Node A: a.bc.d.ef.g
Node B: a.bc.i.j.kl.m

- Address of the lowest common parent: a.bc
- Relative address from $A$ to $B:-4 / i . j . k l . m$ (The number 4 results from the 4 components, $d, e, f$ and $g$, which must pass on the data packet in the upward direction)
In order to guarantee correct operation of the routing, the relative address must be adapted each time it passes an intermediate node. It is sufficient to adapt the address offset. This is always done by the parent node: If a node receives a data packet from one of its subnets, the address offset is increased by the length of the address component of this subnet.
- If the new address offset is $<0$, then the data packet must be passed further upward.
- If the address offset is $>=0$, then the data packet must be passed on to the child node whose local address corresponds to the relative path, starting from the address offset. First of all, however, the address offset must be increased by the length of the local address of the child node, so that the child node sees the correct address.
A special situation results if the error mentioned above occurs during the determination of the common parent node. In this case the address offset of the actual parent node is negative, but this value is larger than the length of the partial address of the subnet from which the packet originated. So that the next node sees a correct relative address in this case, the node concerned must do the following: it must discover the error, calculate the local address of the child node on the basis of the address of the predecessor node and the length difference, and adapt the address offset accordingly. In this case, too, the address components as such remain unchanged; only the offset is changed.


## Broadcast addresses

There are two types of broadcast - global and local. A global broadcast is sent to all the nodes in a network. The empty node address with a length of 0 is reserved for this purpose.
Local broadcasts are sent to all the devices in a network area. For this purpose, all the bits of the network address are set to 1 . This is permissible both in relative and in absolute addresses.
A block driver must be able to process both kinds of broadcast addresses. This means: empty network addresses as well as network addresses whose bits were all set to 1 must be interpreted and sent to all devices concerned.

### 6.4.1.10.3 Symbol Configuration

Use the symbol configuration for preparing symbols with specific access rights for project variables. You can use these symbols to access the variables externally.

For providing symbols to an OPC UA Server, we recommend using the new configuration editor (IEC Symbol Set Configuration) for CODESYS 3.5 SP18 and higher.
«y Chapter 6.4.1.10.4 "IEC Symbol Set Configuration" on page 1945

When generating code, CODESYS also generates a symbol file (*.xml) which includes the description of the symbols.
The symbol file is stored in the project directory. The name of the symbol file is composed as follows: <project name>.<device name>. <application name>.xml

```
Example
proj_xy.PLC1.application.xml
```

You can also generate the symbol file with the "Generate Code" command. This is very useful when downloading to the PLC is not possible.

The variables that you export as symbols can be bundled in the symbol configuration editor or defined in the variables declaration using the \{attribute 'symbol'\} pragma. Another option is using the element in the SFC editor, where you can define the implicitly generated element variables that should be exported to the symbol configuration.
The name of the symbol is generated in the symbol configuration in the following syntax: <application name>.<POU name>.<variable name>. When accessing the variable, you must always provide the complete symbol name in this syntax.
Example MyApplication.PLC_PRG.a or MyApplication.SymFB.a

As a rule, read-only access applies to symbols for input addresses and for variables that are mapped to input channels. Write access is possible for testing purposes in simulation mode only.

The number of symbols which can be exchanged via OPC UA is limited based on the used AC500 V3 PLC type. OPC UA symbols can be configured via two ways:

- Select the symbols in the "Symbol Configuration" object and enable the flag "Support OPC UA features". All selected symbols are counted for the OPC UA tag limitation.
- Define the symbols in the "Symbol Set" object(s) below
"Communication Manager $\rightarrow$ OPC UA Server". All defined symbols are counted for the OPC UA symbol limitation, independently if they are defined in several "Symbol Set" objects.
In case both ways are used in parallel the sum of the symbols is considered for the OPC UA symbols limitation.

The symbol file is downloaded together with the application to the PLC. Depending on the device description, this file can be generated as an additional (child) application. This application is then listed on the "Application" tab of the device editor.
Syntax: <application name>._symbols.
The symbol application is regarded as a "normal" application with respect to the maximum number of applications on the PLC.
If your controller has a user management, then you can assign different access rights to a symbol to the individual user groups (clients). To do this, place the same symbol in different symbol sets and allow the individual user groups (clients) either to access a symbol set or not. An on-site operator or an operating data record, for example, receives more information and access to the same symbols as remote maintenance.

## NOTICE!

When attribute pragmas are used for POUs and variables, the behavior of these POUs and variables can change with respect to the symbol configuration.

## See also

- ${ }^{2}$ Chapter 6.4.1.21.2.27 "Object 'Symbol Configuration'" on page 2523
- ${ }^{*}$ Chapter 6.4.1.20.6.3.43 "Attribute 'symbol"' on page 2314
- ${ }^{*}$ Chapter 6.4.1.20.6.3.46 "Effects of Pragmas on Symbols" on page 2315
- \& Chapter 6.4.1.20.1.5.8.6 "SFC element properties" on page 2080
- ${ }^{*}$ Chapter 6.4.1.21.2.8.5 "Tab 'Applications"" on page 2434
- \& Chapter 6.4.1.21.3.7.23 "Command 'Simulation" on page 2640

| Creating a |
| :--- |
| symbol configu- |
| ration |

1. Select the "Application" object in the device tree.
2. Click "Project $\rightarrow$ Add Object $\rightarrow$ Symbol Configuration".
$\Rightarrow$ The "Symbol Configuration" object is added to the device tree and the objects editor opens.
3. Open the "View" menu of the editor and activate the categories of variables that should be provided in the configuration editor. Click "Build" in the symbol configuration editor.
$\Rightarrow$ All variables (according to the currently defined filter in the "View" menu) are displayed in a tree structure.
4. Select the check boxes of individual variables.

Note: Pay attention to the current settings (see the "Settings" button in the menu bar of the editor).
$\Rightarrow$ In the field below the menu bar of the editor, information is provided about the current situation with accompanying instructions, as well as controls for corrective actions.
5. Follow the prompt in the field below the menu bar. In the following case, this should be only the information that the modified symbol configuration is transferred with the next download or online change.
Click "Build $\rightarrow$ Generate Code" on the CODESYS menu bar.
$\Rightarrow$ The <project name>.<device name>.<application name>.xml file is generated in the project directory.

CODESYS transmits the symbol configuration to the PLC for an application download or online change.

See also

- ${ }^{\sharp}$ Chapter 6.4.1.21.2.27 "Object 'Symbol Configuration'" on page 2523

Creating symbol A symbol set is a defined set of symbols. If supported by the target device, you can combine sets with different access rights for different control clients different symbol sets from the symbols of the application in the symbol configuration editor. The information about the symbol sets is downloaded to the controller. Then you can define the user group that has access to each symbol set. Rights are assigned on the "Symbol Rights" tab of the device editor.

As a result, symbol sets allow different client-specific access rights to a symbol in the controller.
You can download changes to a symbol set definition to the controller in an online change. When the application is deleted on the controller, the symbol sets are also deleted. When building the application, you can create and save a symbol file in XML format for each symbol set.
When symbol sets should be used with OPC UA, it is a requirement that a certificate has to be stored for OPC UA.

As a requirement for creating symbol sets and for granting permissions on the controller, the user management has to be configured both for optional and for forced device user management:

1. In the editor of the controller, open the "Users and Groups" tab and click the $\ddagger$ button.
2. When the user management is still not enabled, first click [Yes] to acknowledge the prompt about enabling the user management. In the "Add Device User" dialog, specify a new user name and a password for a user of the Administrator group. Then confirm the password and click [OK].
3. In the "Device User Logon" dialog, specify the "User name" and "Password". Then click [OK]. The user and groups are displayed on the "Users and Groups" tab.

In the following section, you will see an example of steps for creating symbol sets and the assignment of rights on the controller.

## Requirements - The application has a defined symbol configuration in the project.

- A user management is configured for the controller (see above). For the example here, there should be a user group that has the necessary rights for the servicing of the plant. By default, this type of user group, named "Service", is already created.
- The "Enable symbol sets" option is enabled in the settings of the symbol configuration.
- The connection to the controller is configured in the "Communication Settings" of the device editor.

1. Click the $\mp$ button in the editor of the "Symbol Configuration" in order to create a new symbol set. In the "Add New Symbol Set" dialog, specify "Startup" as the name for the group.
2. In the toolbar of the dialog, click the ${ }^{* i}$ button ([Build]) in order to display all symbols which are available in the project. Select the users who should belong to the group. Save the project.
3. Click Online $\rightarrow$ Login to connect the application to the controller. Click [Yes] to the prompt of whether or not the application should be downloaded to the device.
4. In the editor of the symbol configuration, click the [Configure Symbol Rights] button. The "Symbol Rights" tab of the device editor opens.
5. Click the $\ddagger$ button([Synchronization]) to synchronize the display of the symbol sets with the device.
$\Rightarrow$ In "Symbol Sets", you see all sets that have currently been downloaded for the application (for this example, at least "Startup"). In "Rights", a table shows the user groups that are created in the user management of the controller. In the example, we assume that the default groups "Administrator" and "Service" have been created. When a symbol set is selected on the left, you see on the right the permissions of the individual user groups to this symbol set ( $\Longleftarrow$ access granted; $\sim$ access denied). The possible type of access is already defined for each symbol in the symbol configuration (read, write, execute).
6. On the left, select the [Startup] symbol set and double-click the preset minus sign for "Administrator" as well as for "Service".
$\Rightarrow$ The symbol changes into a plus sign. The "Administrator" and "Service" groups now have access to the symbols in the [Startup] symbol set.

See also

- $\Rightarrow$ Chapter 6.4.1.11.4 "Handling of Device User Management" on page 1971

Chapter 6.3.6.2.1 "General" on page 1787

> The number of symbols which can be exchanged via OPC UA is limited based on the used AC500 V3 PLC type. OPC UA symbols can be configured via two ways:

- Select the symbols in the "Symbol Configuration" object and enable the flag "Support OPC UA features". All selected symbols are counted for the OPC UA tag limitation.
- Define the symbols in the "Symbol Set" object(s) below
"Communication Manager $\rightarrow$ OPC UA Server". All defined symbols are counted for the OPC UA symbol limitation, independently if they are defined in several "Symbol Set" objects.
In case both ways are used in parallel the sum of the symbols is considered for the OPC UA symbols limitation.

In addition to the known "Symbol Configuration Editor", with CODESYS Communication there is also a new editor for configuring IEC symbols for exchanging with other controllers.
Currently, the "IEC Symbol Publishing" object for the publication of symbols via "OPC UA" is available. The object has to be inserted in the device tree below the application below the "Communication Manager" object and there again below an "OPC UA Server" object.
You can insert multiple "IEC Symbol Publishing" type objects to configure different symbol sets for different purposes. Individually assign the object names accordingly. In the OPC UA Client, a separate folder with the assigned name is then created for each symbol set.

The IEC symbol set configuration is based on precompile information. In contrast to the known "Symbol Configuration", the project does not necessarily have to be in an error-free compilable state to get the variables and types available in the project for a configuration in the editor. Discrepancies between the editor and the project, which may result due to changes in the project, will be brought to your attention and you will be assisted in making corrections.
« Chapter 6.4.1.10.3 "Symbol Configuration" on page 1941

Creating and configuring a symbol set

Requirement:
You have a project with IEC variables and IEC data types. You want to exchange a specific set of variables as symbols with other OPC UA Clients via an OPC UA Server. For the following instructions: To control the publication, you have also created a data source of type OPC UA in the project. See here: Establishing the Connection of a Data Source OPC UA Client to an OPC UA Server.
Chapter 6.4.1.10.6.7 "Data Source OPC UA Client" on page 1960
In the following, you create the required symbol configuration for a symbol set in the editor of an object of type "IEC Symbol Publishing":

## ② Chapter 6.4.1.21.2.14 "Object: IEC Symbol Publishing" on page 2466

1. Click [Add Object] to add a "Communication Manager" below the application.
2. Add an "OPC UA Server" below the communication manager.
3. Add an "IEC Symbol Publishing" object below the "OPC UA Server" and rename the object (for example, "webvisu", or "client xy", depending on the intended use of the symbol set to be configured).
4. Double-click the symbol publishing object to open the editor.
$\Rightarrow$ Below "Precompile Sets" in the left part of the window, the variables and types which are currently supported in the project for publishing are listed.
5. Select one or more variables or a POU containing variables and drag the selection to the right part of the window.
$\Rightarrow$ The variable(s) are inserted into the table.
6. Configure all desired variables in the table for export as symbols. If necessary, assign a different symbol name and set the access rights. When assigning the access rights, note the "Use access rights defined in the user management" option at the top of the editor. When this option is selected, the rights assigned in the user management of the controller for the symbol set are also considered in addition to the access rights set here in the editor. The stronger restriction is applied.
7. If you have structured data types in the project, then on the "Symbol Type Editor" tab you can make a specific symbol configuration for different variants of the type:
8. For example, on the "Symbol Type Editor" tab, drag a POU of type STRUCT to the table on the right side of the window. Click the ... button to open the Data type members of <STRUCT> dialog.
9. Select the members of the <STRUCT> POU to be exported as symbols and configure accordingly. Close the dialog.
10. Option: On the "Symbol Type Editor" tab, drag the same POU of type STRUCT to the table a second time and configure it with the corresponding name as a "variant" of the first one.
11. In the project, change something in the declaration of the STRUCT type, for example, add another member. Then click $\$$ "Refresh" in the symbol editor.
$\Rightarrow$ An 6 icon is displayed at the corresponding entry in the table. In the tooltip of the icon, you will be prompted to click the ... button to resynchronize between the editor and the project.
12. Click the $\ldots$ button.
$\Rightarrow$ The "Data type members of<STRUCT>" dialog opens and by clicking [D] the editor and project are synchronized again. The new member is added to the configuration table.
13. When the symbol configuration is finished, log in to the controller and open the data source object created for OPC UA to check the symbols for publishing.
$\Rightarrow$ On the "Variables" tab, you can see the symbol set(s) with the variables and types you have defined as symbols.

### 6.4.1.10.5 Network Variables

Network variables can be used to exchange data between two AC500 V3 PLCs. The variables must be defined in strict and identical lists on both PLCs which are sender and receiver. The lists can be in one or more projects.
A step-by-step description on how to use network variables can be found in the application example.

The network variable list in the sender is a global variable list where specific log and transfer parameters are defined in their object properties. By adding these properties, you create a "network variable list (sender)" from an ordinary "GVL". You can also insert a "Network Variable List (Sender)" object directly into the device tree when this object already has these parameters set.
The network variable list in the receiver is of type "Network Variable List (Receiver)". When creating one, select the respective network variable list of the server. As an alternative, you can read this variable list from an export file that was generated from the sender list. An export file is required anyway for defining the sender list in another project.
The network variables are transmitted as broadcasting in one direction only: sender to receiver. However, it is also possible for a device to contain both sender and receiver lists.
For the NetVarUdp library version 3.5.7.0 and later, a receiver channel is no longer assigned when confirmed transfer is not selected. In this way, network variable exchange is also possible between two controllers on one hardware device .

## NOTICE!

- If the exchanging devices should be senders and receivers, then the variable list identifiers must be unique in order to prevent abnormal operation. The variable list identifiers are defined in the "Properties" dialog of an object GVL.
- Data exchange via network variables is not possible when:
- The device (target system) does not support it.
- A firewall blocks the communication.
- Another client or application is using the UDP port that is set in the properties of the network variable list.
- More than one application per sender device and receiver device use network variable lists.
- Only arrays that have limits defined with a literal or constant are transmitted to the receiver application. Constant expressions are not permitted for this purpose.
Example: "arrVar : ARRAY[0..g_iArraySize-1] OF INT ;" is not transmitted, but "arrVar : ARRAY[0..10] OF INT ;" is transmitted.
- The maximum size of a network variable is 255 bytes. The possible number of network variables is unlimited.
- If the size of the GVL exceeds the maximum length of the network telegram, then the data is split into multiple telegrams. Depending on the configuration, this can result in data inconsistencies.

See also

- « Chapter 6.4.1.21.4.11.12 "Dialog 'Properties' - 'Network Variables'" on page 2760
- $y^{*}$ Chapter 6.4.1.21.2.12 "Object 'GVL' - Global Variable List" on page 2465
- $\Leftrightarrow$ Chapter 6.3.5.2 "Data exchange between AC500 V3 PLCs via EtherCAT" on page 1751


### 6.4.1.10.6 Data Link with Data Sources

## General

In order to have read/write access to the remote devices and their running applications, you can add a data source manager to your application with one or more data sources.
The functionality of the data source manager allows for establishing connections and communication to remote devices, and it makes its data available through data source variables. At this time, the partners communicate by means of a point-to-point connection. Depending on the network where the controllers are located, a connection is established via the CODESYS data source types or CODESYS ApplicationV3.

Data source The data source type CODESYS Symbolic is available only together with a CODESYS HMI type CODESYS Symbolic device. However, then it is advantageous to use this type.

Below a CODESYS HMI device, you can configure the data link either with data source type CODESYS Symbolic or with data source type CODESYS ApplicationV3. We recommend that you select the data source type CODESYS ApplicationV3 only when there are no resources for the symbol configuration available on the remote device. For example, this is the case with embedded or mini PLCs whose applications often do not contain a symbol configuration.

The requirement for a connection setup is that symbols have been configured in the remote device and as a result a symbol file exists. The application in the remote device has a symbol configuration. Then the data link can take place via symbolic monitoring.

In the case of symbolic monitoring, the symbol file on the remote device is read and the stored variable information is used for the data source variables and the data transfer. The advantage is that the application does not have to be updated in the local device when someone modifies the remote application without updating its symbol configuration. If the symbol file is also located on your development system (either a file or a symbol configuration object as part of your project), then the local symbol file can also be read. Then you can work offline during the development phase.

During the development phase, you can create a variable list offline by means of local symbol configuration files. In this way, you can develop a local application offline based on the symbol information without a connection to a data source.

The following connection types are possible:

- "CODESYS V2":

The devices exist in the same network. The V2 runtime on the remote PLC provides a communication interface.

- "CODESYS V2 (Via gateway)"

The devices do not exist in the same network. They are connected via a V2 gateway. Note: For this connection, a "CoDeSys V2.3 Gateway Server" (V2 gateway) has to be installed on the development computer where CODESYS V3 is running.

- "CODESYS V3"

The devices exist in the same network. The V3 runtime on the remote PLC provides a communication interface.

- "CODESYS V3 (Via gateway)":

The devices do not exist in the same network. They are connected via a V3 gateway.

## Data source

 type CODESYS ApplicationV3This data source type is available below all device types.
The data link with CODESYS ApplicationV3 data source type in done by means of address monitoring. This requires that the address information between the remote PLC and the local device match. The runtime system of the local application needs valid communication parameters in order to establish the connection.


Disadvantage: If you modify the remote application, then you also have to update the local application afterwards (for example, the HMI application.

The advantage is that a symbol configuration is not required in the remote application.

## Data transmission

At runtime of the local application, the data source variables that appear in the data source editor of the "Variables" tab are updated in configurable time intervals. The remote application is also executed at this time. Variables that are configured in the visualization, in the trend, as alarms, or for recipes are transferred and stored automatically. When a variable is accessed in IEC code only, the variable is not updated automatically. In this case, you have to select the "Update always" option in the data source editor of the "Variables" tab.
The data source types support the (read or write) data access to variables of the source PLC for the following data types:

- Scalar value at top level

Example: PLC_PRG.hugo

- Property to a program or GVL by means of a call when it is marked with \{attribute
monitoring := 'call'\}.
Example: PLC_PRG.PropertyCall
- Variable which is mapped to bit addresses

Example in PLC_PRG: x AT \%MX0. 5 : BOOL;

- Variable (type BIT) in a function block

Example: Declaration in DUT: x, y : BIT, bit access: PLC_PRG.dutInst.y

- Structured obtainable variable

Example: PLC_PRG.outerInst.innerInst.dwVar

- Property to POU instance when it is marked with \{attribute monitoring $:=$ 'call'\}.
Example: PLC_PRG.instance.PropertyCall
- Property at top level and to an instance when it is marked with \{attribute monitoring := 'variable'\}.
Note: This cannot be written by monitoring or by the data sources.
- Array access with literal index

Example: PLC_PRG.arrofints [3], PLC_PRG.inst.arrOfBool[1]

- Nested access (for example, array of structures or structure of arrays)



## CAUTION!

Bit access used in visualizations that are transferred via a data source connection function only if they contain literal offset specifications. A visualization cannot process an offset specification by defined constants.

* Chapter 6.4.1.21.2.5 "Object 'Data Source Manager'" on page 2412
*) "Dialog 'Add Data Source"' on page 2413
« Chapter 6.4.1.21.2.6 "Object 'Data Source"' on page 2414


## Initially Adding a Data Source

For data exchange between your local device and a remote device, add a "Data Source Manager" object and then a "Data Source" below your application in the device tree. A wizard guides you through the configuration of the data source. Afterwards, you can change the settings at any time in the editor of the object. However, it is not possible to modify the data source type later.

Use the "CODESYS Symbolic" data source type unless there are no resources available in the remote PLC for a symbol configuration. As long as the symbol configuration in the remote device is not impacted by an application change, you have the advantage that the application in the local device does not have to be updated.

When adding a data source, select a data source type. Then specify the connection settings of the point-to-point connection to the remote device. Ideally, the remote device is running during this time and CODESYS can establish the connection to it immediately. Then all available data source variables from the remote PLC are displayed. Select the variables that should be transferred. You can also select all variables. Then the data source is initialized automatically, the data source variables are created below the "DataSources_Objects" folder, and another data source is added below the data source manager.

If the data is transferred using symbolic monitoring and the symbol file is stored on your development system, then you can read the variable information from the symbol file and work offline. The symbol file is stored either as a file on your development system or as a symbol configuration object as part of your project (in CODESYS).
The initial settings can be changed at any time in the data source editor.
See also

- ${ }^{4}$ ) Chapter 6.4.1.21.2.5 "Object 'Data Source Manager'" on page 2412

Initially connecting devices symbolically via 'CODESYS V3'

A "CODESYS Control Win V3" is running on the remote device. Moreover, an application is running with a symbol configuration.

1. Below your application in the device tree, insert a "Data Source Manager" object.
2. Select the "Data Source Manager" object and click "Add Object $\rightarrow$ Data Source". $\Rightarrow$ The "Add Data Source" dialog opens.
3. In the "Name" field, specify the data source name.
$\Rightarrow$ Example: ds_Remote_Device
4. As the data source type, select "CODESYS Symbolic".
$\Rightarrow$ The data transfer is done via symbolic monitoring. The "Initialize Data Source Wizard Provider settings" dialog opens.
5. As the connection type, select "CODESYS V3".
6. For "Type of name or address", select the "Node name" option.
7. In the "Connection Settings" group, specify the connection parameters for configuring the remote device. Example: "[03A7)"
$\Rightarrow$ The connection to the remote device is established and the application is read. The "Initialize Data Source Wizard - Browse data items" dialog also opens. The read remote control variables are displayed in the tree view on the "Variables" entry. The top node is the application, which is displayed with its remote application name.
8. In the tree view, select which control variables should be transferred. Then click "Finish".
$\Rightarrow$ The data source is initially configured. The ds_Remote_Device object is added below the "Data Source Manager" node. The object is open, and on the "Variables" tab, the data source variables to be generated are displayed in the tree view. The GVL ds_Remote_Device, where the data source variables are declared, is located below the "DataSource_Objects" folder.

Initially connecting devices symbolically via 'CODESYS V3 (Via gateway)'

A "CODESYS Control Win V3" is running on the remote device. Moreover, an application is running with a symbol configuration. The remote device exists in another network so that the communication has to be routed via a gateway.

1. Below your application in the device tree, insert a "Data Source Manager" object.
2. Select the "Data Source Manager" object and click "Add Object $\rightarrow$ Data Source". $\Rightarrow$ The "Add Data Source" dialog opens.
3. In the "Name" field, specify the data source name.
$\Rightarrow$ Example: ds_Remote_Device
4. As the data source type, select "CODESYS Symbolic".
$\Rightarrow$ The data transfer is done via symbolic monitoring. The "Initialize Data Source Wizard Provider settings" dialog opens.
5. As the connection type, select "CODESYS V3 (Via gateway)".
$\Rightarrow$ You can also specify the communication parameters for the gateway.
6. For "Type of name or address", select the "Node name" option.
7. In the "Connection Settings" group, specify the connection parameters for configuring the remote device. Example: "[03A7)"
$\Rightarrow$ The connection to the remote device is established and the application is read. The "Initialize Data Source Wizard - Browse data items" dialog also opens. The read remote control variables are displayed in the tree view on the "Variables" entry. The top node is the application, which is displayed with its remote application name.
8. In the tree view, select which control variables should be transferred. Then click "Finish".
$\Rightarrow$ The data source is initially configured. The ds_Remote_Device object is added below the "Data Source Manager" node. The object is open, and on the "Variables" tab, the data source variables to be generated are displayed in the tree view. The GVL ds_Remote_Device, where the data source variables are declared, is located below the "DataSource_Objects" folder.

Initially con- A "CoDeSys V2.3 SP PLCWinNT V2.4" is running on the remote device. Moreover, an applicanecting devices symbolically via 'CODESYS V2'

1. Below your application in the device tree, insert a "Data Source Manager" object.
2. Select the "Data Source Manager" object and click "Add Object $\rightarrow$ Data Source".
$\Rightarrow$ The "Add Data Source" dialog opens.
3. In the "Name" field, specify the data source name.
$\Rightarrow$ Example: ds_Remote_Device
4. As the data source type, select "CODESYS Symbolic".
$\Rightarrow$ The data transfer is done via symbolic monitoring. The "Initialize Data Source Wizard Provider settings" dialog opens.
5. As the connection type, select "CODESYS V2".
6. In the "Connection Settings" group, specify the connection parameters for configuring the remote device.

Example: driver type TCP/IP (Level 2 Route), address localhost, port 1200
$\Rightarrow$ The connection to the remote device is established and the application is read. The "Initialize Data Source Wizard - Browse data items" dialog also opens. The read remote control variables are displayed in the tree view on the "Variables" entry.
7. In the tree view, select which control variables should be transferred. Then click "Finish".
$\Rightarrow$ The data source is initially configured. The ds_Remote_Device object is added below the "Data Source Manager" node. The object is open, and on the "Variables" tab, the data source variables to be generated are displayed in the tree view. The GVL ds_Remote_Device, where the data source variables are declared, is located below the "DataSource_Objects" folder.

Initially connecting devices symbolically via 'CODESYS V2 (Via gateway)'

A "CoDeSys V2.3 SP PLCWinNT V2.4" is running on the remote device. Moreover, an application is running with a symbol configuration. The remote device exists in another network so that the communication has to be routed via a gateway.

1. Below your application in the device tree, insert a "Data Source Manager" object.
2. Select the "Data Source Manager" object and click "Add Object $\rightarrow$ Data Source".
$\Rightarrow$ The "Add Data Source" dialog opens.
3. In the "Name" field, specify the data source name.
```
A Example:ds_Remote_Device
```

4. As the data source type, select "CODESYS Symbolic".
$\Rightarrow$ The data transfer is done via symbolic monitoring. The "Initialize Data Source Wizard Provider settings" dialog opens.
5. As the connection type, select "CODESYS V2 (Via gateway)".
$\Rightarrow$ You can also specify the communication parameters for the gateway.
6. In the "Connection Settings" group, specify the connection parameters for both the gateway and the device configuring the remote device.
Example: driver type TCP/IP (Level 2 Route), address localhost, port 1200
$\Rightarrow$ The connection to the remote device is established and the application is read. The "Initialize Data Source Wizard - Browse data items" dialog opens. The remote control variables are displayed in the tree view on the "Variables" entry.
7. In the "Connection Settings" group, specify the connection parameters for configuring the remote device. Example: "[03A7)"
$\Rightarrow$ The connection to the remote device is established and the application is read. The "Initialize Data Source Wizard - Browse data items" dialog opens. The read remote control variables are displayed in the tree view on the "Variables" entry. The top node is the application, which is displayed with its remote application name.
8. In the tree view, select which control variables should be transferred. Then click "Finish".
$\Rightarrow$ The data source is initially configured. The ds_Remote_Device object is added below the "Data Source Manager" node. The object is open, and on the "Variables" tab, the data source variables to be generated are displayed in the tree view. The GVL ds_Remote_Device, where the data source variables are declared, is located below the "-DataSource_Objects" folder.

Initially adding data source variables from a symbol file

Ideally, the same symbol file on the remote device is saved on your development system.

1. Below your application in the device tree, insert a "Data Source Manager" object.
2. Select the "Data Source Manager" object and click "Add Object $\rightarrow$ Data Source" in the context menu.
$\Rightarrow$ The "Add Data Source" dialog opens.
3. In the "Name" field, specify the data source name.
$\Rightarrow$ Example: ds_Symbols
4. As the data source type, select "CODESYS Symbolic".
$\Rightarrow$ The data transfer is done via symbolic monitoring. The "Initialize Data Source Wizard Provider settings" dialog opens.
5. In "Variable information", click the "From symbol file" entry.
6. In "Select symbol file", specify the location and the file name of the symbol file. When the code is generated, an XML symbol file is created in the project directory by default.
$\Rightarrow$ Example: D: \Projects\V3.5 SP11\Project_A. Device.Application.xml
Hint: When a symbol file is specified, no additional connection settings have to be configured. A connection is not established. You are working offline. You have to configure the connection settings only when you need current data from the controller which is transferred online. In the "Variable information" settings, select the "From connection settings" option.
7. Click the "Next" button.
$\Rightarrow$ The "Initialize Data Source Wizard - Browse data items" dialog opens. The read symbols are displayed in the tree view on the "Variables" entry.
8. In the tree view, select the symbols to be transferred. Then click "Finish".
$\Rightarrow$ The data source is initially configured. The ds_Symbols object is added below the "Data Source Manager" node. The object is open, and on the "Variables" tab, the data source variables that were generated based on the symbol file are displayed in the tree view. The GVL ds_Symbols, where the data source variables are declared, is located below the "DatāSource_Objects" folder.

Initially adding data source variables from a symbol configuration

Your active project contains the control application for the remove device. The control application includes a symbol configuration with symbols that are added to your local application as data source variables.

1. Below your local application in the device tree, insert the "Data Source Manager" object.
2. Select the "Data Source Manager" object and click "Add Object $\rightarrow$ Data Source".
$\Rightarrow$ The "Add Data Source" dialog opens.
3. In the "Name" field, specify the data source name.
```
A Example:ds_Symbols
```

4. As the data source type, select "CODESYS Symbolic".
$\Rightarrow$ The data transfer is done via symbolic monitoring. The "Initialize Data Source Wizard Provider settings" dialog opens.
5. In "Variable information", select the "<remote device>.<application>.symbol configuration" entry.
$\Rightarrow$ Example: Device.Application.Symbol Configuration Hint: When a symbol file is specified, no additional connection settings have to be configured. A connection is not established. You are working offline.
6. Click the "Next" button.
$\Rightarrow$ The "Initialize Data Source Wizard - Browse data items" dialog opens. The read symbols are displayed in the tree view on the "Variables" entry.
7. In the tree view, select the symbols to be transferred. Click "Finish".
$\Rightarrow$ The data source is initially configured. The ds_Symbols object is added below the "Data Source Manager" node. The object is open, and on the "Variables" tab, the data source variables that were generated based on the symbol configuration are displayed in the tree view. The GVL ds_Symbols, where the data source variables are declared, is located below the "DatāSource_Objects" folder.

Initially con-
necting devices with address monitoring

A "CODESYS Control Win V3" is running on the remote device. The project of the remote device is located on your development computer. The engineered application there does not contain a symbol configuration.

Use this communication link only if there are no resources available in the remote PLC for a symbol configuration.

1. Below your application in the device tree, insert a "Data Source Manager" object.
2. Select the "Data Source Manager" object and click "Add Object $\rightarrow$ Data Source". $\Rightarrow$ The "Add Data Source" dialog opens.
3. In the "Name" field, specify the data source name.
$\Rightarrow$ Example: ds_Remote_Device
4. As the data source type, select "CODESYS ApplicationV3".
$\Rightarrow$ The data transfer is done by means of address monitoring. The "Initialize Data Source Wizard - Provider settings" dialog opens.
5. For "Select the project type", select the "Other Project" option.
6. For "Choose file", specify the file and location of the project on the remote device.

Example: C:\Data\Projects \PLC_A.project.
$\Rightarrow$ The remote device is displayed in the tree view of the window below, and as a result the connection was established.
7. Click the "From device" link.
$\Rightarrow$ The connection parameters to the remote device are read and displayed in the dialog. The connection is configured.
8. Click "Next>".
$\Rightarrow$ The "Initialize Data Source Wizard - Browse data items" dialog opens. The remote control variables are displayed in the tree view on the "Variables" entry.
9. In the tree view, select which control variables should be transferred. Then click "Finish".
$\Rightarrow$ The data source is configured. A connection is established. The settings are stored in the object and can be modified in the editor of the object.
The data source is initially configured. The ds Remote Device object is added below the "Data Source Manager" node. The object is open, and on the "Variables" tab, the data source variables to be generated are displayed in the tree view. The GVL ds_Remote_Device, where the data source variables are declared, is located below the "DataSource_Objects" folder.

See also

- « Chapter 6.4.1.21.2.6.4 "Tab 'Communication' via CODESYS Symbolic" on page 2417
- ${ }^{*}$ Chapter 6.4.1.21.2.6.5 "Tab 'Communication' via CODESYS ApplicationV3" on page 2422


## Editing data source variables

In runtime mode, the remote data is saved to the data source variables. The data source variables and their mapping to the remote variables are displayed in the data source editor below of the "Variables" tab. If the local and remote variables have the same names and the same data types, then the data is mapped 1:1. The variables and the data types are created automatically. That is the regular procedure.
You can also map to existing variables. This is necessary, for example, if a visualization includes a data type in an interface. Then the same data must be passed to this visualization. In this case, the declared local variable and the remote variable have the same data type, for example from one library. Moreover, you can map a local variable with a conforming data type to a remote variable. The data type can be created in the "Type Mappings" tab.
The specifically created variables and data types are declared in the "DataSources_Objects" folder. For each data source, a global variable list of the same name as the data source is declared there. Moreover, the data source variables usually have the identical or conforming data type as the remote control variable and they are declared as user-defined data types (DUT objects). Considering all data sources, multiple declaration of the same data types is avoided.

Do not edit the data interface in the "DataSources_Objects" folder manually. It is created initially when adding a data source. Changes can be made later in the editor of the data source.

See also

- *2 Chapter 6.4.1.10 "Working with Controller Networks" on page 1936
- $\&$ Chapter 6.4.1.10.6.5 " Updating data interfaces" on page 1957
- Chapter 6.4.1.21.2.6.2 "Tab 'Variables'" on page 2415

Selecting varia- You can edit the selection of the data source variables. bles for data transfer

Requirement: The remote device and its application are running. A data source manager is already inserted below the local application with a data source.

1. Open the editor of the data source.
2. Select the "Variables" tab.
3. Click "Update Variables".
$\Rightarrow$ The "Browse Variables" dialog opens.
4. Activate the variables that should be transferred and click "OK" to close the dialog.
$\Rightarrow$ The data source variables are modified according to the selection. The declaration of variables and data types is also modified.

The "Variables" tab shows the modified selection. Moreover, the mapped remote variable is listed in the "Remote variable" column.

See also

- ${ }^{*} \gg$ Chapter 6.4.1.10 "Working with Controller Networks" on page 1936
- 告 Chapter 6.4.1.10.6.5 " Updating data interfaces" on page 1957
- ${ }^{4}$ Chapter 6.4.1.21.2.6.2 "Tab 'Variables'" on page 2415


## Mapping remote variables to a new variables

You need to map a remote variable to a global implicit variable that is created new. That is the regular procedure for transposing data source to 1:1.
$\checkmark$ Requirement: A project is open. A data source manager and a data source below it are located in the device tree of the local application.

1. Open the editor of the data source.
2. Select the "Variables" tab.
$\Rightarrow$ The data source variables are listed.
3. Select a variable and click the symbol in the "Create or map" column.
4. Specify a name in "Local variable".
$\Rightarrow$ A variable is declared automatically and it contains the same value as the mapped remote variable.

## Mapping remote <br> variables to a <br> existing variables <br> You need to map a remote variable to an existing variable.

$\checkmark \quad$ Requirement: A data source manager and a data source below it are located in the device tree of the local application. The remote data that should be transferred is displayed in the editor of the data source in the "Variable" tab

1. Open the editor of the data source.
2. Select the "Variables" tab.
3. Select a variable and click the *symbol in the "Create or map" column.
$\Rightarrow$ A variable contains the same value as the mapped remote variable.
Mapping remote
variables to $\quad$ First, create a conforming data type and then use it for a variable.
local variables
with a con-
forming data
type
$\checkmark$ Requirement: A data source manager and a data source below it are located in the device tree of the local application. The remote data that should be transferred is displayed in the editor of the data source in the "Variable" tab
4. Open the editor of the data source.
5. Select the "Type Mappings" tab.
6. Select the data type in the list that you want to edit.
$\Rightarrow$ The elements of the data type are listed in the window below the data type list
7. Specify a name for the data type. Example: DataType_A. Select the name for the remote data types to which the local type should conform. Example: Library1. DataType_A.
8. Modify it in the window below the data type list and remove the elements that are not necessary for the data transfer.
9. Select the symbol for this data type in the "Create or map" column. $\Rightarrow$ The data type DataType_A is declared in the "DataSources_Objects" folder.
10. Select the "Variables" tab.
11. Specify a name in the "Local variable" column. Example: Var_A
12. Select the symbol in the "Create or map" column.
13. Specify the data type DataType_A in the "Mapping type" column.
14. Select the remote variable with the data that should be transferred. Example: appPLC_A. Data_A. Use the input assistance for this.
$\Rightarrow$ A variable Var_A is declared automatically with the user-defined data type DataType_A. During data transfer, it receives the data of the mapped remote variables.

## Example

## Library SnakeUtil

The example demonstrates how variables of the data source are created. At this time, new variables are created, data is mapped to existing data types and their variables, and new data types are created with type-conforming mapping.
The remote PLC uses POU instances from the SnakeUtil library and the HMI device visualizes these POU instances. This is why the HMI application requires a variable in the operating interface that has a data type appropriate for a visualization template. As a result, the SnakeUtil library is linked integrated into the HMI application and the HMI variables instantiate the SnakeUtil. SnakeVisu visualization function block.

The following library function blocks from the SnakeUtil library are used in the remote PLC.

- Function block SnakeUtil. Snake: Equipped with much logic and calling from external functions.
- DUT SnakeUtil. PositionInfo: Two values (of the variables $x$ and $y$ )
- DUT: SnakeUtil. DrawingInfo: Image ID
- The SnakeUtil. SnakeVisu visualization function block with transfer parameter SnakeUtil. Snake visualizes the Snake function block.
The following settings are entered in the editor of the "Type Mappings" tab:

| SnakeUtil.Positioninfo |  | PositionInfo | SnakeUtil.Positioninfo |
| :--- | :--- | :--- | :--- |
| GeneratedType_2 | GeneratedType_2 | GeneratedType_2 |  |
| GeneratedType_3 | GeneratedType_3 | GeneratedType_3 |  |
| Snake | Snake | SnakeUtil.Snake |  |
| SnakeUtil.Drawinginfo | Drawinginfo | SnakeUtil.Drawinginfo |  |

In the visualization, a frame is inserted with a reference to SnakeUtil. SnakeVisu. This expects to have the type SnakeUtil. Snake.
The data types SnakeUtil. PositionInfo and SnakeUtil. DrawingInfo are mapped to existing data types (* symbol in the "Create or map" column). The data types are small and contain data only.
The SnakeUtil. Snake function block is very complex and calls external functions that are not available in the HMI visualization. The function block with code is not required in the visualization. You need a less extensive but compatible and conforming type in the HMI visualization. Therefore, do not create the original data type directly. Instead, first modify the original data type and remove the unnecessary elements. Then create the new data type Snake by selecting the symbol in the "Create or map" column.

## Editing Communication

You have added a "Data Source Manager" object and below it a "Data Source" object below your application in the device tree. The connection parameters are displayed in the data source editor of the "Communication" tab. You can modify it there.
The data source type and the current connection type are listed in the status bar. It is not possible to modify the data source type later.

See also

- *y Chapter 6.4.1.10.6.2 "Initially Adding a Data Source" on page 1949
- $\quad$ « Chapter 6.4.1.21.2.6.4 "Tab 'Communication' via CODESYS Symbolic" on page 2417
- ${ }^{4}$, Chapter 6.4.1.21.2.6.5 "Tab 'Communication' via CODESYS ApplicationV3" on page 2422


## Updating data interfaces

The data source variables are updated cyclically in runtime mode. Only the data is updated that either is used in the current visualization or has the property "Update always".

You can define the time interval. Moreover, you can define variables whose data is transferred in each update interval, and therefore they are always update. To update variables that are not used in the application code, you can implement an update programmatically with the help of interface functions from the data source manager.

## NOTICE!

If data traffic between the remote and local device is too high, then the update rate is reduced automatically. This can lead to an incomplete transfer.

See also

- ${ }^{\text {® }}$ Chapter 6.4.1.21.2.6.2 "Tab 'Variables'" on page 2415


## Setting the update rate

1. Open the editor of the data source.
2. Click the "General and Diagnosis" tab.
3. Specify a value in the "Update rate" field.

Example: 100
$\Rightarrow$ The data from the remote device to the local device is transferred every 100 ms .

See also

- ② Chapter 6.4.1.21.2.6.7 "Tab 'General and Diagnosis"' on page 2426


## Selecting the

 variable for 'Update always'

1. Open the editor of the data source.
2. Activate the option "Update always" for a variable.
$\Rightarrow$ The data of the variables is transferred at each update cycle, even when the data has not changed.

See also

- $\&$ Further information on page 2415

The data source manager provides interface functions in the Datasources library. If a data source manager is integrated in the application code, then the global variable g_Datasources is instantiated automatically. This provides access to the interface functions.
Then you can update individual variables that are not called in the active visualization.

| Example | The variable ivar is activated and deactivated by means of methods from the Datasources <br> library so that its value is transferred. Furthermore, you can configure that the variable is <br> updated only over a defined duration in order to save transfer capacity. |
| :--- | :--- |

```
//Synchronize with DatasourcesTask and block until access is
possible
//Regard the feedback in ERR_OK or in ERR_DE_MULTITASKING_LOCKED
g_Datasources.BeginDataConfiguration(TRUE);
// Activate variable
g_DataServer.UseData(ADR('RemoteDevice.Application.PLC_PRG.iVar'));
// Deactivate variable
//
g_DataServer.ReleaseData(ADR('RemoteDevice.Application.PLC_PRG.iVar'
));
g_DataServer.EndDataConfiguration();
```

The data configuration is started with BeginDataConfiguration (TRUE), thus initializing the synchronization of the task DatasourceTask with the application task. The value TRUE blocks the processing until the access to the variable is possible; FALSE repeats access attempts without blocking. The return values ERR_OK and ERR_DS_MULTITASKING_LOCKED provide feedback about the access attempts.
When synchronization is successful, the variable is activated by means of the UseData method. Then the data configuration is completed with the EndDataConfiguration method and the synchronization triggered again with the task DatasourceTask.
The ReleaseData method is used in the same way for deactivating the variable again at the desired processing time.

## Using remote data

The variables that are listed in the data source editor of the "Variables" tab (and declared in the "DataSources_Objects" folder) can be used in your application like IEC variables. For example, you can visualize the variables.
If multiple data sources are available and therefore conflicts occur regarding unique variable names, then you must specify the data source name as the prefix. If no conflicts occur, then this is not necessary and you can map the variables without a data source prefix.
<data source name>.<function block name>.<variable name>

Displaying vari- You need to show the variable value iTemp of a remote device in a visualization element of a able values from visualization in the local application (with the data source manager). the remote device

Initial situation: A data source dsRemotePLC is below the local data source manager where the connection to the remote device is configured. In addition, the variable iTemp is selected in the data source editor of the "Variables" tab.

1. Select the visualization element in the editor view. Select the properties "Text variables" "Text variable" in the "Properties" view.
2. Select the iTemp variable.
$\Rightarrow$ The variable mapping is qualified. Example: dsRemotePLC.PLC_PRG.iTemp.
3. Select the "Text" property of the visualization element and type in the following: Temperature: \%s
$\Rightarrow$ The value of the iTemp variable from the remote device RemoteDevice is displayed.
4. Download and start the remote application.
5. Download and start the local application.
$\Rightarrow$ The visualization starts and displays the actual value of iTemp.

## NOTICE!

The visualization integrated in CODESYS does not display actual values of variables that are transferred by means of a data source connection. The integrated visualization displays only the initialization values or the last otherwise delivered values because they do not establish a connection to the data sources.

## NOTICE!

If variables are used that are not called in the visualization code, then the variables must be updated in the application code by means of functions from the data source interface.

## Data Source OPC UA Client

## Establishing the Connection of a Data Source OPC UA Client to an OPC UA Server

Requirement:

- An OPC UA Server is available. For a description of the OPC UA Server which is included in the standard installation CODESYS, see the chapter "OPC UA Server".
« Chapter 6.3.6.2 "OPC UA server for AC500 V3 products" on page 1787
- You have installed the CODESYS Security Agent add-on in CODESYS.
- CODESYS is open.
- The "Allow anonymous login" option is selected for your controller in the "Change Communication Policy" dialog of the device editor ("Communication Settings" tab, "Change Communication Policy" command, "Device" menu). Or the user management has been explicitly disabled (for example, by switching to "Optional user management" in the "Change Communication Policy" dialog and then "Reset Origin").

1. Start the OPC UA Server.
2. Create a new CODESYS project.
3. Add a "Data Source Manager" object to the application.
4. Add a "Data Source" "OPC UA-Client" to the "Data Source Manager".
$\Rightarrow$ The "Initialize Data Source" dialog opens.
5. In the "Initialize Data Source" dialog, configure the data source for communication in the following.


The settings from this dialog are reflected accordingly on the Communication tab of the data source manager:

- For "Where to find the server layout", define how the information about the existing variables and types should be detected. When you select the "Browse Live Server" option, the OPC UA Client connects to the OPC UA Server for this purpose and reads the information there. When you select "From Information Model", the client reads the same information from an installed information model and does not require a running OPC UA Server to do this.
- Click the [Show All Endpoints] button to open the "Available Endpoints" dialog.
- Select an endpoint which defines an "Encrypt \& Sign" message security mode and a corresponding security strategy. After the dialog is closed, these settings are transferred to the "Security" section of the "Initialize Data Source" dialog.
- Choose a suitable "Client certificate" to access the server for browsing purposes. If a certificate is not available for selection yet, then you can have one generated immediately. To do this, click the 数 button to open the "Generate self-signed certificate" dialog. Define a password for your private key and a file name for the certificate. When you click [OK], the certificate is generated and automatically entered into the certificate store. The *.cer and *.pfx certificate files are stored with the project file. As a result, you can "give" the certificate with the public key (*.cer) to the server so that it "knows" the certificate. You can also share the private key ( ${ }^{*}$. pfx ) to make the project usable on another machine (for browsing).
Note that this certificate can be used only for browsing the server for variables and data types. An additional certificate is required for data exchange in online mode. Its creation is described below.

6. Click "Next".Now the client scans the OPC UA Server to find the variables and types of the OPC UA Server. The OPC UA Server has to be online to do this.
7. Now select one or more variables.
$\Rightarrow$ These variables can be exchanged later via encrypted communication between the OPC UA Client and the OPC UA Server. For the variables, components are created in the "Devices" view, in the "DataSources_Objects" folder. The variables can be used in the application.
8. In the next steps, you create a certificate for the encrypted communication from the OPC UA Client to the OPC UA Server.
9. Click "View $\rightarrow$ Security Screen".
10. Switch to the "Devices" tab.
11. In the view on the left, select the controller.
$\Rightarrow$ In the right view, all services of the controller are displayed which require a certificate.
12. Select the service "CmpOPCUAClient".
13. Create a new certificate for the device. To do this, click the icon.
$\Rightarrow$ The "Certificate Settings" dialog opens.
14. Define the certificate parameters and click "OK" to close the dialog.
$\Rightarrow$ The certificate is created on the controller.
15. Click the button and save the certificate to the local file directory of the OPC UA Server, in the folder certs.
$\Rightarrow$ Now when you restart the OPC UA Server, it will recognize the client certificate. The the server sends its certificate to the client. In the following steps, this certificate will be made "trusted" to the client.
16. To do this, in the "Security Screen" view, on the "Devices" tab, click the "Certificates in Quarantine" folder in the left area.
$\Rightarrow$ The certificate is displayed in the right area.
17. Drag this certificate to the "Trusted Certificates" folder.
$\Rightarrow$ Now the server certificate is "trusted" by the client.
18. Now when you connect to the controller and the application starts, the data source variables of the OPC UA Client can be exchanged with the OPC UA Server via the encrypted connection.

See also

- ${ }^{4}$ Chapter 6.4.1.21.2.6.6 "Tab 'Communication' via OPC UA Server" on page 2424


## Error Messages

Error 8448 Message: Identity mismatch
Typical error cause: The application information on the target device and the HMI to not match.
Correction: A new download should be done on the HMI to update the application information.

## Error $8449 \quad$ Message: Login failed <br> Typical error cause

- The communication settings are incorrect.
- The user name or password is not correct.

Correction: Check the communication settings for the target device. Make sure that the user name and password are correct.

Error 8450 Message: Login necessary
Typical error cause: There is no connection to the controller.
Correction: Check the communication settings for the target device. Make sure that the user name and password are correct.

Error 8452 Message: The device cannot be reached.
Typical error cause

- The device is switched off.
- The communication settings are incorrect.
- The network connection does not work.

Correction: Make sure that the target device is switched on. Check the communication settings and the network connection to the target device.

Message: The device name is not unique.
Typical error cause: Multiple devices with the same name were found in the target system. Correction: Make sure that the target device has a unique name.

Error $8454 \quad$ Message: No symbols available
Typical error cause: Either no application has been downloaded to the target device, or an application with another symbol configuration.

Correction: Make sure that the application is downloaded with the correct symbol configuration to the target device.

Error 8457 Message: The connection is denied due to no resources.
Typical error cause: All available communication channels of the target device are used.

## Using a Dynamic Connection to an OPC UA Server

The settings for the communication of a OPC UA Client data source to an OPC UA Server can be dynamically configured from the IEC code and can also be changed at runtime. For this purpose, a structure is available in the DatasourceOpcUAServer library.

When using this kind of "dynamic connection", at runtime, for example, the server URL can be changed again if the device is located in a different network than at the time of configuration. It can also be used to configure detailed settings on the OPC UA connection.

## Requirement

- You have created an OPC UA Client data source in the CODESYS project. For more information, see: Establishing the Connection of a Data Source OPC UA Client to an OPC UA Server.
\#y Chapter 6.4.1.10.6.7 "Data Source OPC UA Client" on page 1960

1. On the "Communication" tab of the data source, now select the "Read Connection Settings from IEC Variable" option.
2. In static memory (for example, in a GVL or in a program in your project), create a variable of type DatasourceOpcUaServer.UserConfigurationV1.

Example: config variable in the PLC_PRG program

```
PROGRAM PLC_PRG
VAR
    config : DatasourceOpcUaServer.UserConfigurationV1 := (
        SessionName := 'MyCustomSession',
                                    EndpointUrl := 'opc.tcp://DevXY:4840',
                        SecurityPolicyUri :=
DatasourceOpcUaServer.CmpOpcUaStack_Interfaces.OpcUa_SecurityPolic
y_None);
            value : INT;
END_VAR
```

3. In the input field below the "Read Connection Settings from IEC Variable" option, reference the variable created in step 2:
Example: PLC_PRG.config
Now at any time you can fill the fields of this UserConfigurationV1 structure with values. For example, this can be done when starting the controller, from a configuration file, or from a visualization dialog.
When filling the structure with setting values, the following applies:

- The EndpointUrl setting always has to be defined.
- All other settings can be assigned as desired. They allow, for example, the use of a different security mode than in the usual editor. Some settings, which are not available in the editor, can also be configured on the OPC-UA connection.

When all connection settings are configured, you can set the xValid parameter of the UserkonfigurationV1 to TRUE.

Example: In the implementation part of PLC_PRG:
config.SubscriptionPublishingInterval := config.SamplingInterval := TIME\#2S;
config.xValid := TRUE;
Then the data source starts to establish the connection in the next cycle.
Changes to the settings while a connection is established have no effect and are only applied the next time a new connection is established.
When the xValid setting is set to FALSE, the data source automatically disconnects from the server and the settings can be changed again.
In the documentation view of the DatasourceOpcUAServer library, you can see an overview of the currently available setting parameters of the structure and their default values in the library manager. New versions of the data sources might provide new options.

### 6.4.1.10.7 Subordinate safety controller

If a safety controller is below the standard controller, then the communication with the development system and the data exchange run via the standard controller. The communication links of the safety controller can interrupted the execution of commands that affect the standard controller. You find a notice about this for each these command.

## Possible interruptions

- Temporary interruption: During the execution of the command (for example: download), the connections with the safety controller are interrupted first and then are automatically available again afterwards. If the interruption time is too long, then safety-oriented reactions can occur in the output devices and connected network variable receiver safety controllers. Then in the safety controller, the corresponding communication errors must be acknowledged (if not done automatically) in order to end the safety-oriented reactions. This affects the connection to their field devices and network variable receiver connections to other sender safety controllers. In the case of a connected safety controller with network variable senders, the communication errors must be acknowledged in the other safety controllers.
- Permanent interruption: The execution of commands (for example: delete) leads to an interruption that is ended again by another action (for example: download). As a result of the interruption, safety-oriented reactions can occur in the output devices and connected network variable receiver safety controllers. After ending the interruption, the corresponding communication errors must be acknowledged in the safety controller (if not done automatically) in order to end the safety-oriented reactions.

For a subordinate safety controller, the routing runs via "<Name of SafetyApp>_Mapping". In some cases, it can happen that the user can see this application in the device tree.

## CAUTION!

No commands may be executed in the application "<Name of SafetyApp>_Mapping".

- 乡 $\boldsymbol{y}$ Chapter 6.4.1.10 "Working with Controller Networks" on page 1936


### 6.4.1.11 Downloading an Application to the PLC

6.4.1.11.1 General. ..... 1965
6.4.1.11.2 Configuring the Connection to the PLC. ..... 1965
6.4.1.11.3 Encrypting Communication, Changing Security Settings ..... 1967
6.4.1.11.4 Handling of Device User Management ..... 1971
6.4.1.11.5 Generating Application Code. ..... 1976
6.4.1.11.6 Downloading the application code, logging in, and starting the PLC. ..... 1977
6.4.1.11.7 Generating boot applications. ..... 1978
6.4.1.11.8 Downloading source code to and from the PLC. ..... 1980

### 6.4.1.11.1 General

The multi download tool can be used to update multiple PLCs with firmware or an application at the same time.
ⓨ Chapter 6.6.4 "Multi download tool" on page 4306

In order to transfer your application to the PLC, the program has to be compiled without any errors and the connection settings for the PLC have to be set.

If the communication with the controller is encrypted and/or restricted to specific users, then you need the respective certificates and permissions. See here:

- $\quad$ \& Chapter 6.4.1.11.4 "Handling of Device User Management" on page 1971
- $\quad$ Chapter 6.4.1.9.18 "Protecting an application" on page 1915

You can edit the basic security policy for communication with the device in a dialog on the "Communication Settings" tab of the device editor. See here:

- そhapter 6.4.1.11.3 "Encrypting Communication, Changing Security Settings" on page 1967

When these requirements are fulfilled, the application is downloaded to the PLC at login.

### 6.4.1.11.2 Configuring the Connection to the PLC

The connection to the controller is established by means of a gateway. This gateway can be your development computer or another network computer connected to the controller. The "Communication Settings" dialog is available for configuring the connection path. This dialog opens automatically when you attempt to log in, but the communication settings have not been configured yet.

> If the communication with the controller is encrypted and secured by means of user management, then you need a corresponding certificate and credentials to establish the connection to the controller. In this case, see the relevant instructions on the "Encrypting communication and Changing Security Settings" help page.

Requirement: The project can be compiled without any errors. A programmable logic controller (PLC) is inserted in the device tree. The use of a user management is required for the device, but it is not enabled.

1. In the device tree, select the PLC and click "Project $\rightarrow$ Edit Object".
$\Rightarrow$ The PLC opens in the editor.
2. Click the "Communication Settings" tab.
3. On the menu bar, click "Scan Network".
$\Rightarrow$ The "Select Device" dialog opens. All available devices in the network are shown on the left side.
4. Select the desired device and click "OK".
$\Rightarrow$ A dialog prompt is displayed with the notice that a user management is required for the device, but it is not enabled yet. You are prompted to enable the user management if you want. The notice is displayed that in this case you have to create a new administrator account and then log in as this user.
5. Click "Yes" to close the dialog prompt.
$\Rightarrow$ The "Add Device User" dialog opens to create an initial device administrator.
6. Define the credentials ("Name" and "Password") for the device administrator. Select the "Password can be changed by the user" option.

## NOTICE!

Remember the seriousness of the password: From within the development system, there is no way to access the controller again if you forget the password.

Click "OK" to close the dialog.
$\Rightarrow$ The "Device User Logon" dialog opens.
7. Enter the credentials for the device administrator which you defined in the previous step.
$\Rightarrow$ The connection path for the PLC is set.


You can reset the communication settings view to the original view in the CODESYS options of the device editor.

## See also

- ${ }^{\text {n }}$, Chapter 6.4.1.11.3 "Encrypting Communication, Changing Security Settings" on page 1967
- $\Leftrightarrow$ Chapter 6.4.1.21.2.8.3 "Tab 'Communication Settings'" on page 2427
- $\sum^{\Perp}$ Chapter 6.4.1.21.4.14.7 "Dialog 'Options' - 'Device Editor'" on page 2786


### 6.4.1.11.3 Encrypting Communication, Changing Security Settings

## NOTICE!

Recommendations for data protection
In order to minimize the risk of data security violations, we recommend the following organizational and technical actions for the system where your applications are running. Whenever possible, avoid exposing the PLC and control networks to open networks and the Internet. Use additional data link layers for protection, such as a VPN for remote access. Install firewall mechanisms. Restrict access to authorized persons. Use high-strength passwords. Change any default passwords regularly before and after commissioning.

Use the security features supported by CODESYS and the respective controller, such as encryption of communication with the controller and intentionally restricted user access.

Communication with the device can be protected by means of encryption and user management on the device. You can change the current security preset on the "Communication Settings" tab of the device editor.

## Establishing a connection to the controller, logging in, installing a trusted certificate for encrypted communication

$\checkmark$ Requirement: Encrypted communication with the controller and user management are enforced on the controller. However, an individual password does not exist yet. A certificate has not been installed on your computer and the connection to the controller has not been configured yet.

1. In the device tree, double-click the controller.
$\Rightarrow$ The device editor opens.
2. Click the "Communication Settings" tab.
3. Click "Scan Network".
4. Select a controller.
$\Rightarrow$ A dialog opens, informing you that the certificate of the device does not have a trusted signature for communication. You are prompted whether or not to install this certificate as trusted in the local "Controller Certificates" store on your computer, or accept a session only for this one.

## NOTICE!

A controller certificate installed in this way is valid for only 30 days. This gives you time for the following long-term solutions:

- Creation of an additional self-signed certificate with a longer term (for example, 365 days). You can do this on the security screen if you have installed the CODESYS Security Agent, even if a certificate already exists. Using the PLC shell of the device editor is not a convenient workaround.
See below: "Configuring encrypted communication with a controller certificate with a more long-term validity period"
- Importing a CA-signed certificate. This is currently only possible via the PLC shell commands of the runtime. Therefore, we recommend to use self-signed certificates first.

5. If you want to install the certificate, then select the first option and click "OK" to confirm the dialog prompt.
$\Rightarrow$ The certificate is listed as trusted. After accepting the self-signed certificate for the first time, you can establish an encrypted connection with the controller again and again without further prompts.
A dialog prompt is displayed with the notice that a user management is required for the device, but it is not enabled yet. You are prompted to enable the user management if you want. The notice is displayed that in this case you have to create a new administrator account and then log in as this user.
6. Click "Yes" to close the dialog prompt.
$\Rightarrow$ The "Add Device User" dialog opens to create an initial device administrator.
7. Create a device user in order to edit the user management as this user. In this case, only the "Administrator" group is available. Specify a "Name" and "Password" for the user. The password strength is displayed. Note also the set options regarding a password change. By default, the password can be changed by the user at any time. Click "OK" to confirm.
$\Rightarrow$ The "Device User Logon" dialog opens.
8. Enter the credentials for the device administrator which you defined in the previous step.
$\Rightarrow$ You are logged in on the controller. On the "Users and Groups" tab, you can use the button to switch to synchronized mode. The device user management is displayed there and you can edit it.
After you click "OK" to confirm, the device user management is displayed in the editor view. It contains the user of the "Administrator" group who you just defined. The name of this user is also displayed in the taskbar of the window as "Device User".
9. All saved controller certificates (from Step 5) are stored in the local Windows Certificate Store on your computer. You can access this memory by means of the "Execute", certmgr.msc command.
$\Rightarrow$ All registered certificates for encrypted communication with controllers are listed here in "Controller Certificates".

Configuring a Requirement: The CODESYS Security Agent add-on product is installed. You want to replace controller certif- the temporary certificate (as described above) acquired the first time you connected to the icate with a more long-term validity period
for encrypted communication by means of CODESYS Security Agent (recommended)
Installing a controller certificate for encrypted communication via the PLC shell of the device editor

In this case, the "Security Screen" view provides an additional tab: "Devices". This allows for the simple configuration of certificates for the encrypted communication with controllers. For operation, see the help for CODESYS Security Agent: "Encrypted Communication with Devices via Controller Certificates".

Choose this less convenient method when the CODESYS Security Agent is unavailable to you. In this case, you can set up a certificate with a more long-term validity period for communication encryption on the "PLC Shell" tab of the device editor.

Requirement: You are connected to the controller.

1. At first, you check if a qualified certificate is already on the controller. If no certificate is available, then you create a new certificate.

Open the device editor by double-clicking the controller in the device tree, and select the "PLC Shell" tab.
$\Rightarrow$ The tab appears with a blank display window. Below that is a command line.
2. Type the following command in the command line: cert-getapplist.
$\Rightarrow$ All used certificates are listed. The list includes information about the runtime component and whether or not the certificate is available.
3. If a certificate still does not exist for the component CmpSecureChannel, then type the following command in the input line:

```
cert-genselfsigned <number of the component in the applist>
```

4. Click the "Log" tab and then the refresh button (
$\Rightarrow$ The display shows whether or not the certificate was generated successfully.
5. Change back again to the "PLC Shell" tab and type the command cert-getapplist. $\Rightarrow$ The new certificate for the component CmpSecureChannel is displayed.
6. In the next two steps, activate encrypted communication in the security screen of CODESYS.
7. Open the "Security Screen" by double-clicking in the status bar.
8. On the "User" tab, select the "Enforce encrypted communication" option in the "Security Level" group.
$\Rightarrow$ The communication to all controllers is encrypted. If there is not a certificate on a controller, then you cannot log in to it.

The connecting line between the development system, the gateway, and the controller is displayed in yellow on the "Communication Settings" tab of the device editor of the controller.
9. As an alternative to the "Enforce encrypted communication" option which applies to all controllers, you can also define encrypted communication for specific controllers only. To do this, select the "Communication Settings" tab in the editor of the respective controller. Then click "Encrypted Communication" in the "Device" list box.
$\Rightarrow$ The communication with this controller is encrypted. If there is not a certificate on the controller, then you cannot log in to it.

The connecting line between the development system, the gateway, and the controller is displayed in yellow on the "Communication Settings" tab of the device editor of the controller.
10. When you log in to the controller for the first time, a dialog opens with information that the certificate of the controller is not signed by a trustworthy authority. In addition, the dialog displays information about the certificate and prompts for you to install it as a trustworthy certificate in the local store in the "Controller Certificates" folder.

When you confirm the dialog, the certificate is installed in the local store and you are logged in to the controller.

In the future, communication with the controller will be encrypted automatically with this control certificate.
11. To increase security for key exchange for controllers < V3.5 13.0, you can generate DiffieHellman parameters on the controller. To do this, type the command cert-gendhparams in the input line.
This is no longer required for controllers $>=\mathrm{V} 3.5 .13 .0$.

## NOTICE!

Caution: Generating the Diffie-Hellman parameters can last for several minutes or even several hours. However, this process must be executed only one time for each controller. The Diffie-Hellman parameters increase security for key exchange and for future attacks against encrypted data recording.

## Changing the communication policy (encryption, user management)

## Enabling and <br> disabling <br> enforced encrypted communication

$\checkmark$ Requirement: The connection to the device is established.

1. In the device tree, double-click the controller.
$\Rightarrow$ The device editor opens.
2. Click the "Communication Settings" tab.
3. Open the "Device" menu in the header of the editor. Click "Change Communication Policy".
$\Rightarrow$ The "Change Communication Policy" dialog opens.
4. In the upper part of the dialog, you can toggle between the "Optional encryption", "Enforced encryption", and "No encryption" settings.
5. In the lower part of the dialog, you can toggle between the "Optional user management" and "Enforced user management" settings.

Requirement: The device supports encrypted communication.

1. In the device tree, double-click the controller.
$\Rightarrow$ The device editor opens.
2. Click the "Communication Settings" tab.
3. Open the "Device" menu in the header of the editor. Click "Encrypted Communication". The status toggles between enabled and disabled.
$\Rightarrow$ If the "Encrypted communication" option is selected, then the connection line between the development system, the gateway, and the device is highlighted in the editor in bold and in color in the graphical representation.

See also

- $\Rightarrow$ Chapter 6.4.1.11.4 "Handling of Device User Management" on page 1971
- Ey "Encryption with certificates" on page 1820


### 6.4.1.11.4 Handling of Device User Management

## NOTICE!

## Recommendations for data protection

In order to minimize the risk of data security violations, we recommend the following organizational and technical actions for the system where your applications are running. Whenever possible, avoid exposing the PLC and control networks to open networks and the Internet. Use additional data link layers for protection, such as a VPN for remote access. Install firewall mechanisms. Restrict access to authorized persons. Use high-strength passwords. Change any default passwords regularly before and after commissioning.
Use the security features supported by CODESYS and the respective controller, such as encryption of communication with the controller and intentionally restricted user access.

For devices that support a device user management, the device editor includes the "Users and Groups" tab and the "Access Rights". When offered by the device, you can view the user management for the device here as well as edit it in synchronization mode (not in online mode). Here, you can grant or deny specific permissions on the controller to the defined user groups.
The device user management can already be set up in the device description.

Note the commands in the "Online $\rightarrow$ Security" menu. You can easily add, edit, or remove a user account on the controller where you are currently logged in.

In order for the "Access Rights" tab to be available in the device editor, the corresponding CODESYS option must be selected in the device editor and unlocked in the device description. If the device editor is not available, then contact the manufacturer of the controller.

In order to grant access rights to a user group, first the users and user groups have to be configured on the "Users and Groups" tab of the device editor. User management first has to be set up on the controller before access rights can be configured on it. In case the user management of a device is not enabled yet, it can be enabled in the following way: Either by switching to the synchronized mode on the "Users and Groups" tab, or by adding a new user by means of the command "Online $\rightarrow$ Security $\rightarrow$ Add Device User".

## General information about device user management

For the CODESYS Control devices, a user management is enforced by default.

Access rights can be granted to groups only, not individual users. Therefore every user has to be a member of a group.
Access rights can be granted for the following actions which are executed on the individual objects of the controller:

- Add/Remove
- Modify
- View
- Execute

An object on the controller is usually assigned to just one controller component.
Each object can use all of the listed actions, but usually only the permissions for the following actions are needed on an object:

- "View"
- "Modify"

The objects are organized in a tree structure. There are two root objects for the two kinds of objects:

- "Runtime objects $\rightarrow$ Device": In these objects, all objects are managed that have online access in the controller and therefore have to control the access rights.
- "File system objects $\rightarrow$ /": In these objects, the permissions can be granted to folders of the current execution directory of the controller.
The child objects inherit the access rights from the root object (also "Device" or ""/""). If a permission of a user group is denied or explicitly granted to a parent object, then this affects all child objects.

A single permission can be explicitly granted or denied (green plus sign or red minus sign), or remain "neutral" (light gray character). Neutral means that the permission has been neither explicitly granted nor denied. In this case, the permission of the parent object applies.

If no permission has been explicitly granted or denied in the entire hierarchy of the object, then it is by definition denied. As a result, all permissions are initially denied (exception: the access right for the "View" action). Initially, this permission is explicitly granted for every user group both on the "Device" runtime object as well as on the "" $/$ " file system object. This allows read access to all objects, unless it is explicitly denied in child objects.
For an overview table for the objects, see the "Tab 'Access Rights'" chapter.
See the following instructions for handling the editor for the device user management:

First-time login Requirement: The connection to the controller is configured. The controller supports a device on the controller user management, but one is not active yet. in order to edit or view its user management

1. Double-click the controller device object in the device tree.
$\Rightarrow$ The device editor opens.
2. Click the "Users and Groups" tab.
3. Click
$\Rightarrow$ A dialog opens prompting whether the device user management should be activated.
4. Click "Yes" to acknowledge the dialog prompt.
$\Rightarrow$ The "Add Device User" dialog opens.
5. Now create a device user in order to edit the user management as this user. In this case, only the "Administrator" group is available. Specify a "Name" and "Password" for the user. The password strength is displayed. Note also the set options regarding a password change. By default, the password can be changed by the user at any time. Click "OK" to confirm.
$\Rightarrow$ The "Device User Logon" dialog opens.
6. Specify a "User name" and "Password" for the user who you just defined.
$\Rightarrow$ After you click "OK" to confirm, the device user management is displayed in the editor view. It contains the user of the administrator group who you just defined. The name of this user is also displayed in the taskbar of the window as "Device User".

7. Double-click the controller device object in the device tree.
$\Rightarrow$ The device editor opens.
8. Click the "Users and Groups" tab.
9. Click (Synchronization) to load the user management configuration from the controller to the editor. If you are not logged in to the device yet, then the "Device User Logon" dialog opens for entering the user name and password.
$\Rightarrow$ The user management configuration of the device is shown in the editor.
10. In the "Users" view, click "Add".
$\Rightarrow$ The "Add User" dialog opens.
11. Specify the name of the new user and assign the user to a group. This counts as the user's minimum required default group. The user can be assigned to other groups later. Define and confirm a "Password" for the user. Define whether the user can change the password and whether the user has to change the password at the first login. Click "OK" to confirm.
$\Rightarrow$ The new user appears in the "Users" view as a new node and in the "Groups" view as a new subentry of the selected default group.
```
Changing of
access rights to
controller
objects in the
user manage-
ment of the con-
troller
```

1. Double-click the controller device object in the device tree.
$\Rightarrow$ The device editor opens.
2. Click the "Access Rights" tab.
3. Click (Synchronization) to load the rights management configuration from the controller to the editor. If you are not logged in to the device yet, then the "Device User Logon" dialog opens for entering the access data.
$\Rightarrow$ The access rights management configuration of the device is shown in the editor.

4. Select the object whose access right you want to change to the left in the object tree.
$\Rightarrow$ In the "Rights" view, a table shows the access rights to this object for all configured user groups.
5. Double-click the right in the table that you want to change.
$\Rightarrow$ If the object has child objects, then a dialog prompts whether you want to modify the permissions for the child objects.
6. Click "Yes" or "No" to close the prompt.
$\Rightarrow$ The permissions are switched from "allowed" $\&$ to "not allowed" $ص$, or the other way around. The symbol in the table cell changes accordingly. Explicitly set permissions appear in the table as green or red symbols. Rights that are inherited from a parent object appear as gray symbols.

## Transferring and enabling a saved user management in offline mode from a DUM2 file to a controller

In V3.5 SP16 and higher, a file (*.dum2) to be encrypted with a password is used for exporting a user management.

1. Double-click the controller device object in the device tree.
$\Rightarrow$ The device editor opens.
2. Click the "Users and Groups" tab.
3. Click 图.
$\Rightarrow$ The dialog for selecting a file from the local file system opens.
4. Select the file (<file name>. dum2) with the desired user management from the local file system and click "Open" to confirm.
$\Rightarrow$ The "Enter Password" dialog opens.
5. Specify the password that was assigned when the user management file was exported (possible by means of the 回 button).
CAUTION: The import of a device user management by means of a *. dum2 file completely overwrites the existing user management on the device. In order to log in to the device again afterwards, you need authentication data from the recently imported user management.
$\Rightarrow$ When the password is entered correctly, the configuration from the downloaded user management file is now displayed in the editor view.
6. Edit the configuration however you like. For example, change the user password or add a new user.
$\Rightarrow$ Every change is downloaded immediately to the device.

See also

- ${ }^{\wedge}$ Chapter 6.4.1.21.2.8.15 "Tab 'Access Rights'" on page 2453
- $\Rightarrow$ Chapter 6.4.1.21.4.14.7 "Dialog 'Options' - 'Device Editor"' on page 2786
- „y Chapter 6.4.1.21.2.8.14 "Tab 'Users and Groups'" on page 2450
- ${ }^{*}>$ Chapter 6.4.1.8.2 "Configuring Devices and I/O Mapping" on page 1836
- ${ }^{*}>$ Chapter 6.4.1.11.3 "Encrypting Communication, Changing Security Settings" on page 1967

Configuring the password policy and login lock

You can configure a password policy and login lock for the runtime system. Using these two mechanisms makes sure that the credentials configured for the controller are as secure as possible and that attackers cannot guess the credential by repeated attempts. By default, a password policy is not enabled. The login lock is enabled by default for the administrator user group because this user group needs to fulfill increased security criteria and should also be better protected. When the maximum number of login attempts has been exceeded, the user will be locked out for the configured amount of time ( 3600 seconds by default).

## Unlocking a locked user

The following options are available for unlocking a user who has been locked for a specific period of time:

- An administrator or a member of a user group with write permission for the user group of the locked user assigns a new password for the user.
- The runtime system is restarted.


## Enabling and changing a password policy

1. In the device tree, double-click the controller. $\Rightarrow$ The device editor opens. The "Communication Settings" tab is displayed.
2. In the header, click the "Scan Network" button, and in the "Select Device" dialog, select the desired device. Then click "OK".
$\Rightarrow$ The active path to the controller is set.
3. In the "Device" menu, select "Change Runtime System Password Policy".
4. In the "Change Runtime System Password Policy" dialog, select the "Password policy is active" option.
$\Rightarrow$ The password policy with the displayed password settings is enabled.
$\Rightarrow$ The detailed default password settings can be found in the description of the dialog.
5. Change individual password settings as needed.
6. Click "OK" to confirm the changes made.
$\Rightarrow$ The configured password policy is immediately applied when creating a new user and when changing the password for an existing user. Only new passwords which fulfill the password policy can be created.

## Configuring a login lock

1. In the device tree, double-click the controller.
$\Rightarrow$ The device editor opens. The "Communication Settings" tab is displayed.
2. In the header, click the "Scan Network" button, and in the "Select Device" dialog, select the desired device. Then click "OK".
$\Rightarrow$ The active path to the controller is set.
3. In the "Device" menu, select "Change Runtime System Password Policy".
4. In the "Change Runtime System Password Policy" dialog, you can change the default settings for the login lock or disable the login lock.
5. Click "OK" to confirm the changes made.
$\Rightarrow$ The configured login lock is immediately applied when logging in to the device user management for users of the user group selected in the "Scope".
$\Rightarrow$ When the number of login attempts specified in the "Maximal Retries" field is exceeded, the user will be locked out for the amount of time which is specified in the "Lock duration" field.

### 6.4.1.11.5 Generating Application Code

The application code is the machine code that a PLC executes when you start an application.
CODESYS automatically generates the application code from the source code that was written in the development system. This is done automatically before downloading the application to the PLC. Before the application code is generated, a test is performed for checking the allocations, the data types, and the availability of libraries. Moreover, the memory addresses are allocated when the application code is generated.

You can click "Build $\rightarrow$ Generate Code" to execute this command explicitly. This is useful for detecting any errors in your source code, even when the PLC is not connected yet. The errors are output in the message view in the "Build" category.

## NOTICE!

If you have encrypted the application, then consider the following information: If a (new) boot application is generated on request after an online change, then the boot application is formed in the RAM with the current code that is not encrypted.

Explicitly gener- Requirement: The application can be compiled without any errors. ating the application code
$\triangleright$ Click "Build $\rightarrow$ Generate Code".
$\Rightarrow$ The application code is generated. Detailed information about memory allocation is output in the message view.

See also

- $\sum^{4}$ Chapter 6.4.1.11.7 "Generating boot applications" on page 1978
- $\sum^{2}$ Chapter 6.4.1.21.3.6.1 "Command 'Generate Code"' on page 2618

Messages when When you generate the application code, CODESYS outputs information about memory allocagenerating the tion in the message view. Gaps form in the memory because reallocation is only for new and application code changed POUs and variables due to the incremental memory build. Online changes have the same effect. This fragmentation reduces the amount of available memory. However, you can completely reallocate the memory by clicking "Clean" and therefore increase the amount of free memory.

Syntax errors and bugs that CODESYS detects during the code generation and memory allocation are also output in the message view ("Build" category).
Output information about memory allocation:

- "Size of the generated code" (in bytes): Sum of all code segments
- "Size of global data" (in bytes): Total memory used by the global variables. Inputs and outputs are not included unless inputs or outputs are mapped in the area of the global variables.
- "Total allocated memory size for code and data" (in bytes): The total allocated memory is composed of the already used memory areas plus the reserved, not yet used memory for incremental builds and online changes. After the first build, the already used memory is approximately equal to the highest used address (see below). The largest contiguous memory gap (see below) still corresponds approximately to the difference to the total allocated memory. However, as the number of incremental builds and online changes increases, the number of memory gaps also increases, and the largest contiguous memory gap becomes smaller.
- "Memory area <n>": Contents of the individual reserved memory areas Background: It depends on the PLC which data and code is stored in which memory areas. For example, code and data are located in the same area on the CODESYS Control Win V3. For the addresses $\% I, \circ \mathrm{M}$, and $\% Q$, memory is always reserved, even when a variable is not assigned to an address. After cleaning the application, the memory is reallocated completely. In this case, small gaps could result from the predefined alignment (normally 8). Larger gaps result from changing a date without cleaning, for example by increasing an array area. In this case, only the affected POUs are recompiled. Furthermore, in the case of an online change, the memory is used only for new variables and new code. Memory that was previously reserved by deleted variables and code is made available again. As a result, memory fragmentation can occur after many incremental builds and online changes. This creates many small gaps that might not be usable at all in some cases. To clarify how much memory is safely available, the "largest contiguous memory gap" of the memory area is output during code generation.
- "highest used address" (Byte) : This is the highest reserved address in the entire allocated memory area. During the first build after a "Clean" operation, the memory addresses are output to variables in ascending order, taking into consideration the alignment (usually 8 bytes). As a result, the highest address used at this time corresponds approximately to the amount of memory used. The rest of the allocated memory area is still completely available for incremental builds and online changes.
- "Largest contiguous memory gap" (in bytes): This is the memory size that is available for backup.
Resulting gaps in the allocated memory are reused whenever possible for other changes. When, for example, a global variable of type Byte is added, it is placed in the first free byte of the memory. Even just a small gap is enough for this. However, an FB instance, a variable of the type structure or array, or the code for a POU has to be stored contiguously and therefore occupies more memory accordingly. As a result, they can be allocated only to the largest contiguous memory area. This is why during code generation the "largest contiguous memory gap" that is safely available is output (in bytes), as well as its percentage of the total memory.

Note the options for generating applications.
See also

- $\sum_{\text {- }}$ Chapter 6.4.1.21.4.11.10 "Dialog 'Properties - Application Build Options"" on page 2759


## Encrypting the application code

See also

-     * Chapter 6.4.1.9.18 "Protecting an application" on page 1915


### 6.4.1.11.6 Downloading the application code, logging in, and starting the PLC

In order to download the source code of your application to the PLC, you must log in to the PLC with application. If there are several applications in the project, then you must switch explicitly to the correct application first.

When you download an application to the PLC, CODESYS performs the following checks:
The list of applications on the PLC is compared with the applications available in the project. If they do not match, then you are prompted to download the application that is not on the PLC yet or delete existing applications.
For "externally implemented" blocks in the application to be downloaded, CODESYS checks whether these are available on the PLC. If they are not available, then the message "unresolved reference(s)" is printed to a dialog prompt and to the message view. Then CODESYS compares the parameters (variables) of the blocks in the application to be downloaded and the parameters of the same-named blocks in the application that exists on the PLC (signature check). If there are any discrepancies, then the message "invalid signature(s)" is printed to a dialog prompt and to the message view.
If the "Download Application Info" check box is selected in the application properties, then additional information about the application contents are downloaded to the PLC.
if multiple applications exist for the same device, then notice that the "//O Mapping" dialog contains the definition for which of the applications is used for for the I/O mapping of the device.

See also

- ${ }^{\#}$ Chapter 6.4.1.21.2.2 "Object 'Application"' on page 2410

Transferring an application and starting the program

Requirement: The application contains no errors and the communication settings of the PLC are correct. The application does not exist yet on the PLC: The application and the communication with the controller are not encrypted.

1. Select the required application in the device tree. Skip to Step 3 if you have only one application.
2. Click "Set Active Application".
$\Rightarrow$ The application name appears in bold typeface.
3. Click "Online $\rightarrow$ Login".
$\Rightarrow$ A dialog prompts you whether the application should be created on the PLC.
4. Click "Yes" to confirm.
$\Rightarrow$ The application is downloaded to the PLC.
5. Click "Debug $\boldsymbol{\rightarrow}$ Start" or press the [F5] key.
$\Rightarrow$ The application is running on the controller.

### 6.4.1.11.7 Generating boot applications

A boot application is the application that is started automatically when the controller is switched on or started. For this to happen, the application on the controller must exist as a file <application name>.app.
For each application that is running on the controller, a boot application can also be saved there.
By default, CODESYS generates the boot application automatically when an application is downloaded and transfers them to the PLC. The defaults for generating automatically are located in the "Boot application" category of the application "Properties". When logging in with a changed application, you are still prompted whether or not to generate a new boot application.
In addition, you can create a (new) boot application at any time in online mode by clicking "Online $\rightarrow$ Create boot application".
You can create and save a local copy of a boot application in offline mode as well. Then, you can copy this application to the controller with external tools. In this way, you transfer an application to the controller, even when there is no connection to CODESYS.

Generating boot Requirement: Offline mode; the application is compiled without errors. The connection to the applications on the controller automatically and explicitly controller is configured and the controller is running. The application is active. The following steps demonstrate the options:

1. Click "Online $\rightarrow$ Login".
$\Rightarrow$ The boot application file <application name>. app is created on the controller with the checksum of the boot application <application name>.crc.
2. Click "Online $\boldsymbol{\rightarrow}$ Create Boot Application" explicitly.
$\Rightarrow$ The files on the controller are replaced by new files.
3. Log out.
4. Change the application. Log back in to the controller.
$\Rightarrow$ You are prompted whether an online change should be performed. You see the "Update boot application" check box in the same dialog box. This is cleared by default, but this can be changed in the "Boot Application" category of the application "Properties".
5. Keep the check box cleared and continue login.
$\Rightarrow$ A new boot application file is not created.
6. Log out. Close the project. Stop the controller. Restart the controller.
$\Rightarrow$ The boot application that was created above is running on the controller.


See also

-     * Chapter 6.4.1.21.4.11.10 "Dialog 'Properties - Application Build Options"' on page 2759
- *) Chapter 6.4.1.21.4.11.3 "Dialog 'Properties' - 'Boot Application"' on page 2754
- \& Chapter 6.4.1.14.2 "Executing the online change" on page 2025
- Chapter 6.4.1.21.3.7.4 "Command 'Create Boot Application"' on page 2627


## Creating boot applications in offline mode

Requirement: Offline mode; the application is compiled without errors. You want to generate a boot application for an application and save it in the file directory for copying it later to the controller by using external tools (without CODESYS).

1. Click "Online $\rightarrow$ Create Boot Application".
$\Rightarrow$ A dialog box opens for specifying a save location in the local file system.
2. Click a save path and then click "Save".
3. If the application has changed since the last boot application was generated, then you are prompted to use a new code for the boot application. In this case, click "Yes".
$\Rightarrow$ The "Save as" dialog box opens.
4. Select a directory and click "Save".
$\Rightarrow$ The boot application file <application name>.app is created in the given path.
You are prompted whether or not the build information for the boot application is saved.
5. Click "Yes".
$\Rightarrow$ The build information is saved to the project directory as a file named <application name>. compileinfo. It is a requirement for a possible online change the next time the application is updated. Please note: Clicking "Build $\rightarrow$ Clean" deletes this file.

See also

- ${ }^{4}$ Chapter 6.4.1.21.3.7.4 "Command 'Create Boot Application"' on page 2627


### 6.4.1.11.8 Downloading source code to and from the PLC

The source code of an AC500 V3 project can be downloaded to a V3 PLC memory. At a later point in time, this project can be uploaded and restored directly from the PLC without the engineering PC being required for this $\stackrel{y}{ }{ }^{3}$ Chapter 6.3.5.3 "Source download/upload" on page 1751.

### 6.4.1.12 Testing and Debugging

### 6.4.1.12.1 General

CODESYS provides various options for testing your application and detecting errors. You can start your application in simulation mode, even without connecting any hardware. Using breakpoints and stepping commands, you can examine specific parts of a program. By writing values to variables, you can influence the running program.
Commands are provided that reset your application in various different ways, from resetting only non-persistent variables to completely resetting the controller to factory settings.
See also

- $\Rightarrow$ Chapter 6.4.1.12.3 "Using Breakpoints" on page 1981
- ${ }^{\text {• }}$ - Chapter 6.4.1.12.4 "Stepping Through a Program" on page 1985
- \& Chapter 6.4.1.12.6 "Resetting applications" on page 1990


### 6.4.1.12.2 Testing in simulation mode

Use simulation mode for testing and debugging your program when you do not have a physical target device. In this mode, the application is started on a simulated device.
The command is available only when you are logged out.
Requirement: You program contains no errors (compiler error messages or compile errors) and you are not logged in.

1. Activate simulation mode as follows:

- Click "Online $\rightarrow$ Simulation", or
- Right-click the controller in the device tree and click "Simulation".
$\Rightarrow$ The name of the controller in the device tree is displayed in italics. In the status line, "Simulation" appears highlighted in red. The "Simulation" command is selected in the main menu.

2. Click "Online $\rightarrow$ Login".
3. When logging in with the active application, you will be prompted whether the application "Sim.<device name>.<application name>" should be created and loaded. Click "Yes" to confirm.
$\Rightarrow$ The application is logged onto the PLC.
4. Now you can check and correct the program flow with the commands provided in the main menu in "Debug".
5. Log out from the controller and end the simulation mode.

## See also

- \& Chapter 6.4.1.21.3.7.23 "Command 'Simulation" on page 2640

Limitations

- The focus of the simulation mode is testing and debugging your program of the PLC. That means the functionality of the simulated PLC is limited. Keep in mind that some POUs have no function. They are not creating any compile or download errors, they will simple not work.
- Without an extra available "Virtual Commissioning" license the "Online mode" of the simulated PLC is limited to 2 hours. After 2 hours starting from the "Login", the "Online mode" is automatically terminated and the PLC is logged out.
- It is not possible to create a "Boot Application" in the simulated PLC. Every "Login" starts with an empty simulated PLC and a download of the application is required.
- When logging in to a simulated PLC the first time a "Windows Security Alert" is displayed. Depending on the application, e.g. if any network communication is implemented, it might be necessary to allow the "Virtual AC500" to communicate on one or multiple network types.


### 6.4.1.12.3 Using Breakpoints

Breakpoints are commonly used for debugging programs. CODESYS supports breakpoints in all IEC editors.

You can set breakpoints at specific positions in the program to force an execution stop and to monitor variable values. You can set special data breakpoints to halt program execution when the value of a specific variable changes.
The halt at a breakpoint or data breakpoint can be linked to additional conditions. You can also redefine breakpoints and data breakpoints as execution points where specific code is executed instead of stopping the program.

The "Breakpoints" view provides an overview of all defined breakpoints. It also includes additional commands for processing batch changes to multiple breakpoints.

In the editor, the following symbols identify the status of a breakpoint or execution point:

- The breakpoint is enabled.
- The breakpoint is disabled.
- Dreakpoint is set in another instance of the POU open in the editor.
- The program is halted at the breakpoint.
- 0 The conditional breakpoint is enabled.
- The conditional breakpoint is disabled.
- The execution point is enabled.
- The execution point is disabled.
- 5 The conditional execution point is enabled.
- The conditional execution point is disabled.
- The data breakpoint is enabled.
- The data breakpoint is disabled.
- Halt at data breakpoint
- The data execution point is enabled.
- The data execution point is disabled.
- Halt at data execution point
- The conditional data execution point is enabled.
- $\quad$ The conditional data breakpoint is enabled.

See also

- \& Chapter 6.4.1.21.3.4.12 "Command 'Breakpoints'" on page 2586

Data breakpoints

## The function of data breakpoints depends on the target system. Currently, data breakpoints are possible only with the CODESYS Control Win V3.

Program execution stops at a data breakpoint when the value of a particular variable or memory address changes. As with ordinary breakpoints, the halt can be linked to an additional condition, or specific code can be processed instead of the halt (converted to a data execution point).
You set a data breakpoint either by means of the "New Data Breakpoint" command in the "Debug" menu or by means of the "New" button in the "Breakpoints" view. You specify a qualified variable name or a memory address directly which is to be monitored for changes in its value.

## Example In the following sample code, the memory of the variable inumber is overwritten unintentionally. However, a data breakpoint at the variable iNumber will detect when its value changes. The processing then stops with a corresponding message at the array access, which overwrites the variable value: $I d x=7$. See also below: "Setting a data breakpoint".

```
PROGRAM PLC_PRG
VAR
    Idx : INT;
    Ary : ARRAY[0..3] OF BYTE;
    iNumber : INT := 55;
END VAR
FOR idx := 0 TO 6 DO
    Ary[idx] := 0;
END_FOR
```

Breakpoints in applications with multiple tasks

Basically, debugging is not possible for multiple tasks at the same time. While you are working on a task with breakpoints or stepping, breakpoints are ignored in other tasks.
If a POU containing a breakpoint is used by multiple tasks, then only the debug task is halted because it reaches the breakpoint first. All other tasks continue. The "Call Stack" dialog shows which task is currently halted.

If you need a breakpoint to affect only one specific task, then you can define this in the breakpoint properties.

Breakpoints operate separately for each application so that a "HALT ON BP" does not affect any other applications. This applies also to parent/child applications, even if the breakpoint is set in a block that is used by several applications and whose code is located only once on the PLC.

## NOTICE!

The I/Os that are called by the debug task are not updated at a halt in the breakpoint, even if you select the "Refresh I/Os in Stop" check box in the PLC settings.

If the application stops at a breakpoint on the PLC, then an online change or download causes all tasks to halt which means the PLC will stop. In this case, CODESYS prompts you whether or not to continue with the login.

See also

- ${ }^{\circ}$ Chapter 6.4.1.21.3.4.15 "Command 'Call Stack"' on page 2590

Setting a single breakpoint (example in ST editor)

Requirement: The application is in online mode and running. The operating mode is "Debug".

1. In the editor, open a POU programmed in structured text (ST).
2. Place the cursor in the line where a breakpoint will be set.
3. Click "Debug $\rightarrow$ Toggle Breakpoint" or press [F9].
$\Rightarrow$ The line is marked in red and identified by the "breakpoint enabled" symbol ( $)_{\text {) . If the }}$ program is halted at the breakpoint, then the line is marked by the "stop at breakpoint" symbol ( $\boldsymbol{\bullet})$. The processing of the program is stopped. This is identified in the status line by the HALT ON BP status highlighted in red.
4. Click "Debug $\rightarrow$ Start" or press [F5].
$\Rightarrow$ The program continues.
5. Set more breakpoints and check the variable values at the break position.
6. Place the cursor in the line where a breakpoint should be removed.
7. Click "Debug $\rightarrow$ Toggle Breakpoint" or press [F9].
$\Rightarrow$ The marking disappears. The breakpoint is deleted.

See also

- \& Chapter 6.4.1.21.3.8.9 "Command 'Toggle Breakpoint'" on page 2646


## Defining a breakpoint condition (example in ST editor)

1. In the editor, open a POU programmed in structured text (ST).
2. Place the cursor in the line where a breakpoint will be set.
3. Click "Debug $\rightarrow$ New Breakpoint".
$\Rightarrow$ The "Breakpoint Properties" dialog opens.
4. Click the "Condition" tab.
5. Click "Break when the hit count is a multiple of" in the "Hit Count" list box.

Specify the value " 5 " in the field to the right.
6. In addition, you can define a Boolean condition for when the breakpoint should be active. Select the "Break, when true" check box. Specify a Boolean variable in the text field to the right.
7. Select the "Enable breakpoint immediately" check box.
8. Close the dialog.
$\Rightarrow$ The line is marked red and identified by the "conditional breakpoint enabled" symbol (©)

Monitor the running program. As long as the Boolean variable for the condition is FALSE, the breakpoint condition is not fulfilled and the program continues to run. If you set the variable to TRUE, then the condition is fulfilled and the program halts at the breakpoint every 5th pass.

See also

- ${ }^{\text {B }}$, Chapter 6.4.1.21.4.6 "Dialog 'Breakpoint Properties'" on page 2747


## Defining an execution point (example in ST editor)

1. In the editor, open a POU programmed in structured text (ST).
2. Place the cursor at the position for an execution point.
3. Click "Debug $\rightarrow$ New Breakpoint".
$\Rightarrow$ The "Breakpoint Properties" dialog opens.
4. Click the "Execution Point Settings" tab.
5. Select the "Execution point" option.

In the "Execute the following code" field, type the following statement: iCounter := iCounter + 1;

In the "Print a message in the device log" field, type the following text: Execution point reached \{iCounter\}
6. Close the dialog.

When the program reaches the execution point, it does not halt, but executes the code defined above. In addition. a message is issued to the device log.

See also

- ${ }^{2}>$ Chapter 6.4.1.21.4.6 "Dialog 'Breakpoint Properties'" on page 2747


## Setting a data Requirement: The application is in online mode and running. <br> breakpoint

1. Click "View $\rightarrow$ Breakpoints".
2. Click "Debug $\rightarrow$ New Data Breakpoint".
3. Click the .... button in the "New breakpoint" dialog ("Data" tab.
4. In the "Input assistant" dialog ("Watch Variables" tab), select the variables for which the program should halt when changed.
As an alternative, specify the qualified name of the variable on the "Data" tab directly in the input line. Example: PLC_PRG.iNumber. The exact number of bytes to be monitored is specified as the "Size". A value that corresponds to the data type is set here automatically by default. You can also specify fewer bytes to be monitored.
5. In the "Breakpoints" view, select the line with the data breakpoint and click the 覀 button.
$\Rightarrow$ The line is marked and identified by the "Data breakpoint enabled" symbol ( ). When the program reaches the data breakpoint (meaning when the value of the selected variables changes), the program processing halts. In the implementation part of the POU, the next line is identified by an arrow 0 . This is identified in the status line by the HALT ON BP status highlighted in red.
6. Click "Debug $\rightarrow$ Start" or press [F5].
$\Rightarrow$ The program continues running and halts again when the value of the variables changes again.

See also

- $\boldsymbol{y}^{\boldsymbol{y}}$ Chapter 6.4.1.21.4.6 "Dialog 'Breakpoint Properties'" on page 2747


### 6.4.1.12.4 <br> Stepping Through a Program

You can step through an application and navigate the code. This is useful to check the status of your code at runtime. You can examine the call process, track variable values, or locate errors.
Stepping commands are provided in the "Debug" menu for this purpose. The commands become available when you set breakpoints in online mode and then halt execution at a breakpoint: the application is in "HALT ON BP" state (debug mode). During debug mode, the current break position is highlighted in yellow and marked with the $\Rightarrow$ symbol in the text editors.

## Switching to debug mode

1. Download your application to a controller.
$\Rightarrow$ The application is highlighted in green. CODESYS and the editors of the POUs are in online mode.

2. In the POUs, set breakpoints at the locations in the code that you want to examine.
$\Rightarrow$ All breakpoints are listed in the "Breakpoints" view.
3. Start the application.
$\Rightarrow$ The application starts and the code is processed until the first breakpoint.
Now the application is in debug mode. In the device tree, the application is labeled with "[halt on breakpoint]". The status bar provides information about the operating state:

## HALT ON BP

The editor was opened at the current break position. The line of code with an active breakpoint where program execution was halted is highlighted in yellow and marked by the $\Rightarrow$ symbol. This statement highlighted in yellow has not been executed yet.


Now you can select the various stepping commands or display the call tree.

## Alternatively，you can first start the application and then set a breakpoint．

## Behavior of the stepping com－ mand in the＇De－ bug＇menu

－Command $[$ 프＂Step Over＂
The statement at the breakpoint position is executed．Program execution halts before the next statement in the POU．
If the statement contains a call（from a program，function block instance，function，method， or action），then the subordinate POU is processed completely in one step．
－Command $\ddagger$ 亚＂Step Into＂
The statement at the breakpoint position is executed．Program execution halts before the next statement．
If the statement contains a call（from a program，function block instance，function，method， or action），then the program execution jumps to this subordinate POU．The first statement there is executed and the program execution halts before the next statement．The new current breakpoint position is then in the called POU．
－Command ${ }^{\text {互＂Step Out＂}}$
The command executes the POU from the current breakpoint position to the end of the POU and then jumps back to the calling POU．Program execution halts at the calling position（in the line with the call）．
If the current breakpoint position is in the main program，then the POU is run through to the end．Then the program execution jumps back to the beginning（to the program start at the first line of code in the POU）and halts there．
－Command + 트＂Run to Cursor＂
First set the cursor at any line of code and then execute the command．The program is executed from the current breakpoint position and halts at the current cursor position without executing the code of this line．
－Command $\S$＂Set Next Statement＂
First set the cursor at any line of code（also before the current breakpoint position）and then execute the command．The statement marked with the cursor is executed next．All statements in between are ignored and skipped．
－Command $\Rightarrow$＂Show Next Statement＂
If you do not see the current breakpoint position，then execute the command．Then the window with the current breakpoint position comes into focus and the breakpoint position is visible．
Click＂View $\rightarrow$ Call Stack＂to completely show the previous call tree for the breakpoint position currently reached in the program processing．

The＂Call Stack＂view shows the location of the block in the call structure of the program at all times，even before compiling the application．

See also
－※ Chapter 6．4．1．21．3．8．11＂Command＇Step Into＇＂on page 2647
－祭 Chapter 6．4．1．21．3．8．10＂Command＇Step Over＇＂on page 2646
－« ${ }^{*}$ Chapter 6．4．1．21．3．8．12＂Command＇Step Out＂＇on page 2647
－$\Leftrightarrow$ Chapter 6．4．1．21．3．8．13＂Command＇Run to Cursor＇＂on page 2648
－$\Leftrightarrow$ Chapter 6．4．1．21．3．8．14＂Command＇Set Next Statement＂＂on page 2648
－$\sum^{*}>$ Chapter 6．4．1．21．3．8．15＂Command＇Show Next Statement＂＂on page 2648
－${ }^{4}$ Chapter 6．4．1．21．3．4．16＂Command＇Call tree＂＇on page 2590

### 6.4.1.12.5 Forcing and Writing of Variables



## CAUTION!

Unusual changes to variable values in an application currently running on the controller can lead to undesired behavior of the controlled machinery.
Evaluate possible dangers before forcing variable values. Take the respective safety precautions. Depending on the controlled machinery, the result may lead to damage to machinery and equipment or injury to health and life of personnel.

In CODESYS, variable values in the PLC can be changed in online mode. Here we make a distinction between forcing and writing a previously prepared value.

Writing is done with the "Write Values" command ([Ctrl]+[F7]) and sets the variable to the prepared value one time. In this way, the value can be overwritten again by the program at any time.
Forcing is done with the "Force Values" command ([F7]) and sets the prepared value permanently. For more information, see below.

The preparation of a value for forcing or writing is possible at different places:

- Declaration part: "Prepared value" field
- Implementation part of the FBD/LD/IL editor: inline monitoring field
- Watch view: "Prepared value" field

For instructions about this, see below. In the case that you want to prepare a value again for an already forced value, the "Prepare Value" dialog opens with options for handling the current force value.

Functionality of The prepared value is set to the respective variable at the beginning and end of a task cycle (or forcing
of a processing loop in the case of other task types).

The processing order in each cycle of a task is as follows:

1. Read the inputs
2. Force: Before the first program call, all prepared values are written to the variables by the runtime system, regardless of whether or not they are used by the task.
3. Process the IEC code
4. Force: After the last program call, all prepared values are written to the variables by the runtime system, regardless of whether or not they are used by the task.
5. Write the outputs

Note: It is possible that a forced variable temporarily gets a different value in the cycle while the code is being processed because the IEC code performs an assignment. Then the variable receives the forced value again only at the end of the cycle. The variable value can also be overwritten by the write access of a client to symbols of the application in mid-cycle. For this case, see the "Access variables in sync with IEC tasks" option in the "Properties" of the device object, or the "Configure synchronization with IEC tasks" setting in the symbol configuration. In this way, a PLC handler-supported synchronization of the write accesses by clients can be enabled with the task cycle.


See also
－ⓨ Chapter 6．4．1．9．17＂Task Configuration＂on page 1914
－$\quad$ ${ }^{\circ}$ Chapter 6．4．1．21．4．11．20＂Dialog＇Properties＇－＇Options＇＂on page 2765
－«＂Setting：Configure synchronization with IEC tasks＂on page 2527

Forcing in the declaration part

Requirement：Your application includes a POU with declarations．The application is in online mode．
1．Open the POU in the editor by choosing the command＂Project $\rightarrow$ Edit Object＂．
2．In the declaration part of the editor，double－click in column（1）＂Prepared Value＂of a variable．
$\Rightarrow$ The field can be edited and a value can be entered．When it is a Boolean value，you change the value by clicking in the field．

## Device－Application1．PLC＿PRG

| Expression | Type | Value | Prepared value |
| :---: | :---: | :---: | :---: |
| 考 iColorR | INT | 0 | $100 \text { (1) }$ |
| 零 iColory | INT | 0 | 200 |
| 考 iColorg | INT | 0 | 300 |

3．Perform Step 2 for other variables．
4．Click＂Debug $\rightarrow$ Force Values＂．
$\Rightarrow$ The variable values are overwritten with the prepared values．The values are marked with the Ey symbol．

| Device Application1．PLC＿PRG |  |  |  |
| :---: | :---: | :---: | :---: |
| Expression | Type | Value | Prepared value |
| 1\％iColorR | INT | （F） 100 |  |
| $1{ }^{3} \mathrm{iColory}$ | INT | （F） 200 |  |
| 1 ）iColorg | INT | （F） 300 |  |

You can also force the variable values in the "Watch" view.

## Forcing in the implementation part

## Viewing and editing all forced variables one list

Requirement: The application is in online mode.

1. Open the POU in the editor by choosing the command "Project $\rightarrow$ Edit Object".
2. In the implementation part of the editor, double-click an inline monitoring field (1).
$\Rightarrow$ The "Prepare Value" dialog opens.
3. Enter the new value in the field "Prepare a new value for the next write or force operation".
$\Rightarrow$ The prepared value appears in the inline monitoring field.

4. Click "Debug $\rightarrow$ Force Values".
$\Rightarrow$ The value of the variables is overwritten with the prepared values. The values are marked with the © symbol.


Requirement: The application is in online mode. Multiple variables are forced.

1. Click "View $\rightarrow$ Watch $\rightarrow$ Watch all Forces".
$\Rightarrow$ The "Watch all Forces" view opens. It contains all currently forced variables of the application in the form of a watch list.
2. Select all lines in the list and click "Unforce $\rightarrow$ Unforce and Keep All Selected Values" in the list box in the upper left part of the view.
$\Rightarrow$ The variables are unforced and they get the values that they had before forcing.

Forcing a func- Requirement: An application has a CFC POU that contains a function block, and the application tion block input in CFC

## NOTICE!

This kind of forcing uses a data breakpoint internally and is therefore different from forcing with the "Force Values" command or [F7].
Values that were forced by the command "Force FB Input" do not respond to the commands "Show All Forces" or "Unforce Values".

1. Open the editor of the CFC POU by double-clicking the object in the tree.
2. When using compiler version 3.5.11.x or 3.5.12.x, enable the "forceability" for the desired function block. Select the POU element in CFC and click "CFC $\rightarrow$ Prepare Box for Forcing".
3. Log in to the application on the target device. In CFC, select the input of the POU and click "Force Function Block Input" in the context menu.
$\Rightarrow$ The "Force Value" dialog opens.
4. Set a new value for the input. Example in the case of a TON POU: FALSE for the Boolean input IN, or $t \# 4$ s for the PT input (TIME). Click "OK" to confirm.
$\Rightarrow$ The set value is forced immediately. A green circle is displayed at the upper left of the POU element and the name of the input in the element is highlighted in green. In the case of a Boolean value, a small monitoring view with the value also opens at the input. In the monitoring views, the forced value is displayed, for example in the "Value" column, as in the declaration part.
5. To remove the forced value, click "Force Function Block Input" again. In the "Force Value" dialog, select the "Remove value" option.
$\Rightarrow$ Forcing is canceled. The input gets the current value from the controller.

See also

- ${ }^{4}$ Chapter 6.4.1.21.3.13.34 "Command 'Prepare Box for Forcing'" on page 2697
- $\quad>$ Chapter 6.4.1.21.3.13.35 "Command 'Force Function Block Input"' on page 2697
- $\boldsymbol{y y}^{*}$ Chapter 6.4.1.21.4.8 "Dialog Box 'Prepare Value'" on page 2749
- $\Leftrightarrow$ Chapter 6.4.1.21.3.8.16 "Command 'Force Values'" on page 2649
- $\Leftrightarrow$ Chapter 6.4.1.21.3.8.18 "Command 'Unforce Values"' on page 2650
- $\&$ Chapter 6.4.1.21.3.8.17 "Command 'Write Values"" on page 2649
- $\boldsymbol{y y}^{4}$ Chapter 6.4.1.13.2.3 "Using watch lists" on page 2002


### 6.4.1.12.6 Resetting applications

Resetting the application stops the program and resets the variables to their initialization values. Depending on the type of reset, retain variables and persistent variables are also reset.

- Reset warm: All variables are reset, except RETAIN and PERSISTENT variables.
- Reset cold: All variables are reset, except PERSISTENT variables.
- Reset origin: All variables are reset.
- Reset origin device: All variables are reset and all applications are deleted.

The following sample program and statements clarify the functionality of the various resets.
See also

- « "Lifespan of variables when calling online commands" on page 1922


## Sample program

```
Example
Declaration
```

```
VAR
```

VAR
iVar: INT := 0;
iVar: INT := 0;
END VAR
END VAR
VAR RETAIN
VAR RETAIN
iVarRetain: INT :=0;
iVarRetain: INT :=0;
END_VAR
END_VAR
VAR PERSISTENT
VAR PERSISTENT
iVarPersistent : INT:= 0;
iVarPersistent : INT:= 0;
END_VAR
END_VAR
Implementa-
tion iVar := 100;
iVarRetain := 200;
iVarPersistent :=300;

```
1. Insert the "Persistent Variables" object below the application and open it in the editor.
2. Click "Build \(\rightarrow\) Build".
3. Click "Declare \(\rightarrow\) Add All Instance Paths".
\(\Rightarrow\) The instance path of the persistent variables is inserted.
4. Download the application to the controller.

\section*{Executing a \\ "Reset warm", "Reset cold", and "Reset origin"}

Requirement: The sample program runs on the controller.
1. Click "Online \(\rightarrow\) Login" to switch to online mode.
2. Monitor the variables iVar, iVarRetain, and iVarPersistent.
3. Click "Online \(\rightarrow\) Reset Warm".
\(\Rightarrow\) You are prompted whether you really want to execute the command.
4. Click "Yes" to confirm the dialog.
\(\Rightarrow\) The application is reset. The iVar variable is set to the initialization value 0 . Both of the other variables retain their values.
5. Click "Online \(\rightarrow\) Reset Cold".
\(\Rightarrow\) You are prompted whether you really want to execute the command.
6. Click "Yes" to confirm the dialog.
\(\Rightarrow\) The application is reset. The iVar and iVarRetain variables are set to the initialization value 0 . The iVarPersistent variable retains its value.
7. Click "Online \(\rightarrow\) Reset Origin".
\(\Rightarrow\) You are prompted whether you really want to execute the command.
8. Click "Yes" to confirm the dialog.
\(\Rightarrow\) The application is reset. All variables are reset to their initialization values.

\section*{See also}
- \({ }^{*}\) Chapter 6.4.1.21.3.7.11 "Command 'Reset Warm'" on page 2634
- \({ }^{\bullet} \gg\) Chapter 6.4.1.21.3.7.10 "Command 'Reset Cold'" on page 2633
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.7.12 "Command 'Reset Origin'" on page 2635
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.7.13 "Command 'Reset Origin Device"" on page 2636

\subsection*{6.4.1.12.7 Flow Control}

With flow control, you can monitor the processing of the application program. Flow control is provided for the ST, FBD, LD, and CFC language editors.
With an activated flow control, CODESYS displays the variable values and results from function calls and operations at the respective processing location and time. In this way, the exact lines of code and networks that process the current cycle are marked in colors. Compare this to standard monitoring, in which CODESYS delivers only the value that a variable has between two processing cycles.

Flow control works in all parts of the editor view that are currently visible. "Flow control enabled" is then displayed in the status line as long as the function is active and flow control positions (processed parts of code) are visible in an editor view.

You can write values in the declaration part and implementation part. Forcing is not possible.

\section*{NOTICE!}

Values are written at the end of the current cycle.

\section*{NOTICE!}

When you enable flow control, the cycle time of the application is prolonged.
When "Confirmed Online Mode" is selected in the communication settings, a dialog prompt appears when switching on the flow control to cancel the operation.

When flow control is enabled, it is not possible to use breakpoints or step through the program.

See also
- « \({ }^{*}\) Chapter 6.4.1.21.2.8.3 "Tab 'Communication Settings'" on page 2427
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.8.22 "Command 'Flow Control"" on page 2652

Display of the flow control in different language editors

By default, CODESYS displays the flow control positions of the processed parts of code as green fields. Unprocessed parts of code are displayed in white.

Note that the displayed value of an unprocessed code position is an ordinary monitoring value. This is the value between two task cycles.


In network editors, CODESYS marks the processed networks with bars on the left edge in the flow control color.

In LD, CODESYS displays the currently processed connecting lines in green and all others in gray. The actual value of the connection is also displayed: TRUE by a bold blue line, FALSE by a bold black line, and unknown or analog values by thin black lines. Combinations of these lines are displayed as dashed lines.


In IL, for each statement CODESYS uses two fields for the display of the actual values. One is located to the left of the operator with the current accumulator value, and one is located to the right of the operand with the operand value.

6.4.1.12.8 Determining the current processing position with the call stack

You can use the call stack for determining the current position of the program flow. This function is very useful when stepping into programs.

Requirement: The application is in online mode. The program is halted at a breakpoint or you are stepping into it.
\(\triangleright\) Open the call stack by clicking "View \(\rightarrow\) Call Stack".
\(\Rightarrow\) The call stack opens. The list shows the current location with the complete call path.

The call stack is also available in offline mode and normal online mode (without using debugging functions). In this case, it receives the last displayed location during a stepped execution, but it is displayed in gray.

See also
- \({ }^{*}\) Chapter 6.4.1.21.3.4.15 "Command 'Call Stack"' on page 2590
- \(\quad\) - \({ }^{2}\) Chapter 6.4.1.12.4 "Stepping Through a Program" on page 1985

\subsection*{6.4.1.12.9 Checking the Task Deployment}

The Tab "Task Deployment" of the device editor indicates in an overview the tasks that process the individual inputs and outputs of the I/O mapping of your application and the priority with which they do so. You can check here whether an unintentional overwriting of values is caused, which can lead to undefined values.
1. Generate code for the application: to do this select, for example, the command "Build \(\rightarrow\) Generate Code".
2. Open the device editor by double-clicking on the device object in the device tree. Select the "Task Deployment" tab.
\(\Rightarrow\) You obtain a display of the inputs and outputs of your application and the assignment of the tasks and their priorities. See the description of the "Task Deployment" tab for details.

See also
- \(\sum^{\wedge}\) Chapter 6.4.1.21.2.8.18 "Tab 'Task deployment'" on page 2459
- \(乡\) "General information about I/O mapping" on page 1836

\subsection*{6.4.1.13 Application at Runtime}
\(\qquad\)
6.4.1.13.2 Monitoring of Values 1995
6.4.1.13.3 2007
6.4.1.13.3 Data Recording with Trace.......................................................... 2007
6.4.1.13.4 Data Recording with Trend............................................................ 2016
6.4.1.13.5 Monitoring tasks........................................................................... 2021
6.4.1.13.6 Reading the PLC log................................................................... 2021
6.4.1.13.7 Using PLC shell for requesting information................................... 2022
6.4.1.13.8 PLC operation control via system variables.................................. 2022
6.4.1.13.9 Backup and restore..................................................................... 2024

\subsection*{6.4.1.13.1 General}

When the application is running on the PLC, in the CODESYS Development System there are some features for monitoring and changing the values of the variables as well as for recording and storing the value charts.
Furthermore, you can poll some information from the PLC, you can have a look into the PLClog, display a core dump and monitor the time behavior of the tasks.
Regard also the possibility to restrict the access on the running application in critical states of the machine via online commands provided by CODESYS Development System. For this purpose some system variables are available in a module of the ComponentManager library.

\subsection*{6.4.1.13.2 Monitoring of Values}

\section*{General}

In runtime mode, you can monitor the current variable values of a programming object at different places in a project. The following is what we refer to as monitoring:
- Online view of the programming editor of an object: inline monitoring
- Online view of the declaration editor of an object
- Object-independent, configurable watchlists

When you set the \{attribute 'monitoring'...\} pragma, you can monitor the results from function calls and the current variable values in property-type objects.


More options for recording current variable values:
- Read and save recipes
- Record values on a timeline for displaying the history immediately or later: trace and trend features

\section*{See also}
- \(\Rightarrow\) Chapter 6.4.1.20.6.3.26 "Attribute 'monitoring'" on page 2295
- \({ }^{\bullet} \gg\) Chapter 6.4.1.13.2.3 "Using watch lists" on page 2002
- \(\Leftrightarrow\) Chapter 6.4.1.13.2.4 "Changing Values with Recipes" on page 2003
- \& Chapter 6.4.1.13.3 "Data Recording with Trace" on page 2007
- \(\quad\) \& Chapter 6.4.1.13.4 "Data Recording with Trend" on page 2016

\section*{Calling of monitoring in programming objects}

When an application is running on the controller, the actual values of variables are displayed in the editors of the POUs. This is how the values of variables are monitored.
\(\checkmark\) Requirement: The "Enable inline monitoring" option is activated in "Tools \(\rightarrow\) Options" in the "Text Editor" category on the "Monitoring" tab.
1. Download an application to the controller and start it.
2. Click "Debug \(\rightarrow\) Display Mode \(\rightarrow\) Decimal".
\(\Rightarrow\) The display format of the actual values is set.
3. Click a programming object in the "Devices" view or "POUs" view.
\(\Rightarrow\) The respective editor opens. Actual values of the variables are refreshed continually for both the declaration and implementation.

\section*{Monitoring in the declaration editor}


The actual value of an expression (1) is displayed in the "Value" column (3).
You can write and force a value in the "Prepared Value" (4) column. During the forcing, the actual value is decorated with a red symbol (E).
The expression of an interface reference can be expanded. If the interface points to a global instance, then this global instance is displayed as the first entry below the reference. Afterwards, if the interface reference changes, then the displayed reference is collapsed.

Monitoring in the implementation (inline monitoring)

Inline monitoring is the display of the current variable value in the implementation.
Depending on the implementation language, the following displays are possible in the implementation part:
- Variables have a window with the current value displayed after their name: nResult2 \(\square 17\) If you have prepared values for variables for forcing or writing, then they are displayed in angle brackets in the inline monitoring view after the current value.
After forcing, the respective values are identified by the Eisymbol.
- Network editors and the CFC editor:

Connecting lines are displayed in color according to their actual Boolean value (blue means TRUE, black means FALSE).
- LD editor:

The contact and coil elements are also marked.
For contacts and coils, a prepared value (TRUE or FALSE) is shown in a small view next to the element.
- SFC editor:

Transitions with the value TRUE are displayed in color according to their actual Boolean value (blue means TRUE, black means FALSE).
Active steps are displayed in blue.
Forced transition values are displayed in red in the implementation.
- IL tabular editor:

Current values are displayed in a separate column.



See also
- « Chapter 6.4.1.20.1.4.1 "ST Editor" on page 2049
- \& Chapter 6.4.1.20.1.4.2 "ST editor in online mode" on page 2050
- \(\xi^{*}\) Chapter 6.4.1.20.1.6.2 "FBD/LD/IL editor in online mode" on page 2086
- \(\Leftrightarrow\) Chapter 6.4.1.20.1.5.2 "SFC Editor in Online Mode" on page 2063
- \(\Leftrightarrow\) Chapter 6.4.1.20.1.7.5 "CFC Editor in Online Mode" on page 2103

\section*{Partial monitoring of an array}

Limiting the monitoring range

An expanded array shows the actual values for up to 1000 elements. However, this can be confusing. In addition, an array can contain more than 1000 elements. Then it is helpful to limit the range of displayed elements. You can do this in online mode in the following way.
\(\square\) Requirement: An application is running. It contains a multidimensional array variable with more than 1000 elements. Example: arrBig : ARRAY [0..100, -9..10, -19..20] OF INT;
1. Click in the field of the "Data Type" column for the arrBig variable.
\(\Rightarrow\) The "Monitoring Range" dialog opens.
2. Specify the value \([1,-9,-19]\) for "Start".
3. Specify the value \([1,10,20]\) for "End".
\(\Rightarrow\) The actual values of 800 array elements are displayed in the declaration editor. The range is limited to the elements of the index [1, <i>, <j>] with i from -9 to 10 and j from -19 to 20.

See also
- \({ }^{\text {² }}\) Chapter 6.4.1.20.1.2 "Declaration Editor" on page 2047
- \& Chapter 6.4.1.12.5 "Forcing and Writing of Variables" on page 1987
- ※y Chapter 6.4.1.21.4.10 "Dialog 'Monitoring Range'" on page 2752

Monitoring a When you double-click the editor view of a function block in online mode, a dialog opens where function block you can choose between viewing the basic implementation or a specific instance.
If you select the basic implementation, then the code is displayed in the editor without current values. Now set a breakpoint in the basic implementation. If the execution halts there, then the current values of the instance that is processed first in the program flow are displayed. Now you can step successively through all instances.

If you select one of the instances, then the editor opens with the code of the function block instance. The current values are displayed in the declaration and, if applicable, in the implementation, and are updated continuously.
See also
- \({ }^{4}\), Chapter 6.4.1.21.2.21.3 "Object 'Function Block'" on page 2479
- \({ }^{4}\) Chapter 6.4.1.12.3 "Using Breakpoints" on page 1981

\section*{Monitoring a property}

You can monitor variables in a property object 野 by setting a breakpoint in the function during online mode. When halted there, the current values are displayed.

In addition to your own values, the values of the variables of the superordinate instance are displayed automatically. In the declaration part of the property, the THIS pointer, which points to the superordinate instance, appears in the first line with the current data type information and values.
```

Example
Code

```
```

FUNCTION_BLOCK FB_BaseAlfa

```
FUNCTION_BLOCK FB_BaseAlfa
VAR
VAR
    iBaseLocal : INT;
    iBaseLocal : INT;
    sBaseLocal : STRING;
    sBaseLocal : STRING;
END_VAR
END_VAR
iBaseLocal := iBaseLocal + 1;
iBaseLocal := iBaseLocal + 1;
sBaseLocal := 'Testing Text';
sBaseLocal := 'Testing Text';
FB_BaseAlfa.PorpBeta.Get
FB_BaseAlfa.PorpBeta.Get
iBāseLocal := iBaseLocal + 1;
iBāseLocal := iBaseLocal + 1;
IF iBaseLocal > O THEN
IF iBaseLocal > O THEN
    PropBeta := TRUE;
    PropBeta := TRUE;
END_IF
END_IF
FB_BaseAlfa.PorpBeta.Set
FB_BaseAlfa.PorpBeta.Set
IF PropBeta = TRUE THEN
IF PropBeta = TRUE THEN
    iBaseLocal := 0;
    iBaseLocal := 0;
    sBaseLocal := 'Tested IF';
    sBaseLocal := 'Tested IF';
END_IF
END_IF
PROGRAM PLC_PRG
PROGRAM PLC_PRG
VAR
VAR
    fb_BaseAlfa : FB_BaseAlfa;
    fb_BaseAlfa : FB_BaseAlfa;
END_VA\overline{R}
END_VA\overline{R}
fb_BaseAlfa();
fb_BaseAlfa();
IF fb_BaseAlfa.PropBeta = TRUE THEN
IF fb_BaseAlfa.PropBeta = TRUE THEN
    xResult := TRUE;
    xResult := TRUE;
END_IF
END_IF
IF xReset THEN
IF xReset THEN
    fb_BaseAlfa.PropBeta := TRUE;
    fb_BaseAlfa.PropBeta := TRUE;
    xReset := FALSE;
    xReset := FALSE;
END_IF
END_IF
# FB_BaseAlfa.PropBeta.Get x
Device_AppMonitoringProperty.PLC_PRG..b_BaseAMfa.__getPropBeta
\begin{tabular}{|c|c|c|c|c|c|}
\hline Expression & Type & Value & Prepared value & Address & Comment \\
\hline - * THIS & POINTER TOFB_BASEALFA & 16\#08073548 & & & \\
\hline \(=\) THIS^ & FB_BaseAlfa & & & & \\
\hline - iBaselocal & INT & 1496 & & & In FB_BaseAlfa \\
\hline - sBaselocal & STRING & Testing Text' & & & \\
\hline - PropBeta & B00 & & & & \\
\hline *. _getPropBeta & BOOL & FALSE & & & \\
\hline - PropBeta & BOOL & FALSE & & & \\
\hline
\end{tabular}

\section*{See also}
- ③ Chapter 6.4.1.21.2.21.9 "Object 'Property"' on page 2493

Monitoring of You can monitor the values of subordinate properties 駩 in a function block or program in property access in the superordinate programming object addition to the variable values.
To do this, add either the pragma \{attribute 'monitoring' = 'variable'\} or \{attribute 'monitoring' = 'call'\} to the subordinate property object in the declaration. If you open the superordinate program instance or function block instance at runtime, then the current property values are displayed in the editor in addition to the current variable values.
See also
- « \({ }^{*}\) Chapter 6.4.1.21.2.21.9 "Object 'Property"' on page 2493
- \& Chapter 6.4.1.20.6.3.26 "Attribute 'monitoring"' on page 2295

Monitoring a You can monitor variables in a method object \(\mathbb{M}\) by setting a breakpoint in the method during method online mode. When halted there, the current values are displayed.
In addition to your own values, the values of the variables of the superordinate instance are displayed automatically. In the declaration part of the method, the THIS pointer, which points to the superordinate instance, appears in the first line with the current data type information and values.


\section*{See also}
- © Chapter 6.4.1.21.2.21.9 "Object 'Property'" on page 2493
- \({ }^{\bullet}\) Chapter 6.4.1.21.2.21.6 "Object 'Method"' on page 2485

Monitoring a You can monitor variables in a function object by setting a breakpoint in the function during function online mode. When halted there, the current values are displayed.

Monitoring the return value of a function call

In the ST editor of a POU, the current return value is displayed as inline monitoring at the position of the POU where a function is called.
The following conditions must be fulfilled:
- The value can be interpreted as a 4-byte numeric value. Example: INT, SINT, or LINT.
- The pragma \{attribute 'monitoring' := 'call'\} is inserted into the function.

See also
- \({ }^{*}>\) Chapter 6.4.1.21.2.21.4 "Object 'Function'" on page 2482
- \(\quad>\) Chapter 6.4.1.20.6.3.26 "Attribute 'monitoring'" on page 2295

\section*{Using watch lists}

What is a watch A watch list is a user-defined list of project variables that are collected in one view for the list? purpose of monitoring their values. In online mode, you can write and force variable values in a watch list. Monitoring, writing, and forcing are handled the same way as the declaration editor in online mode. You can customize the format of the representation of floating-point values in the options for monitoring.

There are four, ready-to-use watch lists (Watch <n>) available in a project. Click "View
\(\rightarrow\) Watch".

If the expression is an interface reference, then it can be expanded. If the interface points to a global instance, then this global instance is displayed as the first entry below the reference. If the interface reference changes, then the displayed reference is collapsed.

\section*{See also}
- \({ }^{4}\) Chapter 6.4.1.20.1.2 "Declaration Editor" on page 2047
- 栄 Chapter 6.4.1.21.4.14.19 "Dialog 'Options' - 'Monitoring'" on page 2792

Creating and editing a watch list (offline or online mode)

Requirement: The project is in either online or offline mode. It includes an application with declared variables that you want added to one of the four possible watch lists.
1. Click "View \(\rightarrow\) Watch \(\rightarrow\) Watch \(<n>\) ".
\(\Rightarrow\) The Watch <n> view opens. It contains a blank table row.
2. Double-click the field in the "Expression" column and type a variable to monitor, either manually or with the input assistant.
Syntax: <device name>.<application name>.<object name>.<variable name>
Example: "Dev1.App1.PLC_PRG.ivar"
If you type the name of a structured variable, then the individual components are displayed automatically in other lines in online mode.
3. Define all successive variables that will be monitored with this list. You can change the order by using drag and drop operations.
\(\Rightarrow\) The "Execution point", "Type", "Address", "Comment" fields are filled in automatically according to the variables declaration. The symbol before the expression indicates the type of variable: input variable ( \({ }^{*}(\hat{*})\), output variable \(\left({ }^{( }\right)\)), or ordinary variable ( 0 ).

In online mode, you can also create or edit watch lists by right-clicking and choosing the "Add Watch" command.

\section*{See also}
- ⓨ Chapter 6.4.1.21.3.4.8 "Command 'Watch' - 'Watch <n>"" on page 2584

Adding varia- Requirement: A project is open and running. It includes an application with declared variables
bles by choosing the 'Add Watch' command (online mode)
that you want added to a possible watch list
1. Click "View \(\rightarrow\) Watch \(\langle n\rangle\) " to open the watch list.
2. Place the cursor on a variable in the declaration or implementation part of a POU and right-click to choose the "Add Watch" command.
\(\Rightarrow\) This adds an entry to the list for the selected variable.
3. You can add other variables in this way or by typing directly into the list in the "Expression" field as described above.
\(\Rightarrow\) The watch lists are updated immediately.

If a watch list is not open when you click "Add Watch" for a variable, then it is added automatically to the "Watch 1 " list.

Writing and forcing variable values is also possible in the watch lists. In online mode, the "Prepared value" column is also available.

See also
- \begin{tabular}{|c} 
\\
-
\end{tabular} Chapter 6.4.1.21.3.23.1 "Command 'Add Watch'" on page 2743
- \(\quad\) Chapter 6.4.1.13.2.2 "Calling of monitoring in programming objects" on page 1996
- \& Chapter 6.4.1.12.5 "Forcing and Writing of Variables" on page 1987

\section*{Changing Values with Recipes}

Use recipes to change or read recipes values for a specific set of variables (recipe definition) on the controller at the same time.
You define the basic settings for recipes, such as location and format, in the "Recipe Manager" object. Insert one or more recipe definitions below this object. A recipe definition is composed of one or more recipes for the contained variable. The recipe consists of specific variable values.
You can save a recipe to a file or write directly from files to the PLC.
Recipes can be loaded via the CODESYS development interface, the visualization element, or the application program.

\section*{Using recipes on remote devices}

The variable values from recipes are transferred automatically to and from another controller when they are data source variables and a data source exchange is configured. Reading and writing occurs synchronously. Therefore, CODESYS updates all variables in a recipe at the same time. After reading or writing, you can use the call g_RecipeManager. LastError to check whether or not the transfer was successful (9_RecipeManager. LastError = 0).

See also
- \(\Leftrightarrow\) Chapter 6.4.1.21.2.25 "Object 'Recipe Manager'" on page 2519
- \(\quad\) Chapter 6.4.1.21.2.26 "Object 'Recipe Definition'" on page 2522
- \({ }^{\text {• }}\), Chapter 6.4.1.10.6 "Data Link with Data Sources" on page 1947

\section*{Calling recipe commands in the CODESYS user interface}

The development interface of CODESYS provides commands for generating recipes as well as reading/writing in online mode.

See also
- * Chapter 6.4.1.21.3.20 "Menu 'Recipes'" on page 2723

\section*{Calling recipe commands programmatically or in a visualization}

At runtime, the recipe definition and the recipe commands can be called in the application program. To do this, you implement the process for writing, reading, or monitoring recipe definitions in your code by using the methods of the RecipeManCommands function block from the RecipeManagement library.

In the visualization, the call of the recipe commands is implemented by using the for a visualization element. By doing this, a visualization user can control the execution of recipe commands.

During the initialization process, the recipe management reads the values of the variables that are defined in the recipe definition. This operation takes place at the end of the initialization phase of the application. At this point, all initial values of the application variables are set. This is done to initialize missing values from recipe files correctly.

See also
- RecipeManCommands
- Input Configuration

\section*{Special functionality for floating-point numbers}

For variables of type REAL or LREAL, a distinction is made whether or not the value is exactly convertible.
- If it is possible to convert the value exactly, then only the numerical value is saved in the recipe file.
- If exact conversion is not possible, then a hexadecimal encoded string is written in the recipe file next to the numeric value. This should have the effect that also a non-convertible floating-point number returns the same value.

This feature can be disabled with the RECIPE_GENERATE_SIMPLE_STRINGREAL compiler define.

\footnotetext{
In the object properties of the parent application under "Build", you can enter this RECIPE_GENERATE_SIMPLE_STRINGREAL compiler define in the "Compiler Defines" option. As \(\bar{a}\) result, \(\overline{n o}\) hexadecimal encoded string is stored.

For more information, see: Dialog: Properties: Build
}

\section*{Creating a recipe}
1. In the device tree, select the "Application" object.
2. Click "Project \(\rightarrow\) Add Object \(\rightarrow\) Recipe Manager".
\(\Rightarrow\) The Recipe Manager is added to the device tree.
3. In the device tree, click the "Recipe Manager" object.
4. Click "Project \(\rightarrow\) Add Object \(\rightarrow\) Recipe Manager". \(\Rightarrow\) The recipe definition is added below the Recipe Manager.
5. Open the editor of the recipe definition by double-clicking the object.
6. Double-click the blank field below "Variable". Specify the name of a variable to define a recipe for. You can use the Input Assistant (...) button) for this.
7. Click "Recipes \(\rightarrow\) Add a New Recipe" and specify a name for the new recipe.
\(\Rightarrow\) A column with the new recipe name appears in the editor.
8. Enter the variable value for this recipe in this field.
9. Insert additional fields as needed.
10. Select a variable value for the recipe and click "Recipes \(\rightarrow\) Save Recipe". Select a location and file name.
\(\Rightarrow\) The recipe is saved in the format which was defined in the Recipe Manager.

See also
- \(\boldsymbol{y}^{*}\) Chapter 6.4.1.21.2.25 "Object 'Recipe Manager'" on page 2519
- \({ }^{\bullet} \gg\) Chapter 6.4.1.21.2.26 "Object 'Recipe Definition'" on page 2522
- \(\quad 4\) Chapter 6.4.1.21.3.20 "Menu 'Recipes'" on page 2723

\section*{Loading a recipe from a file}

Requirement: A Recipe Manager is available in the application. In a recipe definition, there is a "myRec" recipe with variable values. A myRec.txt recipe file is located on the file system and contains the entries for this recipe.
Example of the recipe file:
```

PLC_PRG.bVar:=0
PLC-PRG.iVar:=2
PLC-PRG.dwVar:=35232
PLC_PRG.stVar:='first'
PLC_PRG.wstVar:='123443245'

```
1. Double-click the "Recipe Definition" object in the device tree to open the tabular editor for the definition of the individual recipes.
\(\Rightarrow\) You see the myRec column with the current values for this recipe.
2. Edit the myRec.txt file in an external text editor and replace the variable values with other values that you want to load into the recipe definition in CODESYS. Save the file.
3. In the recipe definition, click the "myRec" column and in the context menu click "Load Recipe".
\(\Rightarrow\) A dialog prompt notifies you about the possibly needing to perform an online change when logging in again. An online change is necessary when you change the current values of the recipe variables by loading the recipe.
4. Click "Yes" to close the dialog and continue. Select the myRec.txt file from the file explorer for loading.
\(\Rightarrow\) The recipe values in the recipe definition are updated according to the values read in the file.

If you want to overwrite only individual recipe variables with new values, then remove the values for the other variables before loading to the recipe file. Entries without value definitions are not read, and therefore updating leaves these variables unchanged on the controller and in the project.
For values of the data type REAL/LREAL, the hexadecimal value is also written to the recipe file in some cases. This is necessary so that the exactly identical value is restored when converting back. In this case, you change the decimal value and delete the hexadecimal value.

\section*{See also}
- \({ }^{4}\) Chapter 6.4.1.21.3.20.4 "Command 'Load Recipe"" on page 2724
- \({ }^{*}\) Chapter 6.4.1.21.3.20.8 "Command 'Load and Write Recipe'" on page 2725

\section*{Recipe management on the controller; memory usage}

When you clear the "Recipe management in the PLC" option, the Recipe Manager and recipe definitions will not use any memory on the PLC.
If you select this option, then code is generated for the Recipe Manager and all recipe definitions, and this code is stored on the PLC. The size of the used memory primarily depends on the number pf recipes and their variables, as well as the data type of the variables. Whether or not the fields of the recipe definition are filled also has an effect. The memory usage of recipes cannot be calculated. It has to be determined by experimentation at the time it is needed. The following table merely provides some guiding principles.
\begin{tabular}{|l|l|l|l|}
\hline & \begin{tabular}{l} 
Code Size \\
(bytes)
\end{tabular} & \begin{tabular}{l} 
Data Size \\
(bytes)
\end{tabular} & Total (bytes) \\
\hline Recipe definition with 100 INT variables & 194406 & 79400 & 267352 \\
\hline Recipe definition with 200 INT variables & 238318 & 121284 & 459344 \\
\hline Recipe definition with 300 INT variables & 282230 & 163084 & 543856 \\
\hline Recipe definition with 100 BOOL variables & 192742 & 69884 & 343168 \\
\hline Recipe definition with 200 BOOL variables & 235446 & 101568 & 436872 \\
\hline Recipe definition with 300 BOOL variables & 278146 & 133284 & 510072 \\
\hline Recipe definition with 100 string variables & 203278 & 870084 & 1154000 \\
\hline Recipe definition with 200 string variables & 255570 & 1709784 & 2973296 \\
\hline Recipe definition with 300 string variables & 307886 & 2549484 & 2964112 \\
\hline
\end{tabular}

\section*{Loading recipe values from the controller}

You can apply recipe values on the controller to recipe definitions in the project, even if these definitions have been modified in the project.
Requirement: The "Recipe management in the PLC" is option is selected in the Recipe Manager.
1. Create a recipe definition RecDef1 in the project, containing the variables

PLC_PRG.ivar and PLC_PRG.bvar. Insert a recipe "R1": value for PLC_PRG.ivar: 33; value for PLC_PRG.bvar: TRUE.
2. Log in to the controller and download the application.
\(\Rightarrow\) The recipe file R1. RecDef1.txtrecipe is saved to the default directory of the controller (\$PlcLogic\$).
3. Logout and add another variable PLC_PRG. dwvar to the recipe definition in the project.
4. Edit the recipe definition file R1.RecDef1.txtrecipe on the device by changing the value for PLC_PRG.ivar from 33 to 34 .
Moreover, add another recipe " \(R 2\) " on the device. To do this, copy the
R1.RecDef1.txtrecipe and rename it to R2.RecDef1.txtrecipe. Then edit this
file and change the recipe values: PLC_PRG.ivar: 1, PLC_PRG.bvar: FALSE.
\(\Rightarrow\) Now two recipes "R1" and "R2" are available on the device. In the project, there is only " \(R 1\) ", and it also contains other values than "R1" on the device.
5. Log in to the controller by means of an online change.
6. Click "Load Recipes from Device" from the context menu.
\(\Rightarrow\) A dialog prompt notifies you that executing the command at the next login may trigger an online change, and that the recipes on the runtime system will overwrite the recipes of the current recipe definition.
7. Confirm that you want to continue.
\(\Rightarrow\) A dialog prompt notifies you that the recipe for PLC_PRG. dwvar loaded on the device cannot yield a value from the controller.
8. Confirm that you want to continue.
\(\Rightarrow\) The value of PLC_PRG.ivar in recipe "R1" of the recipe definition in the project changes to 34 . The recipe "R2" with the values 1 and FALSE is also listed in the recipe definition now. PLC_PRG. dwvar remains in the recipe definition.

\subsection*{6.4.1.13.3 Data Recording with Trace}

\section*{General}

You can use a "Trace" to follow the value history of variables on the controller in a similar way as a digital sampling oscilloscope. When the application is in runtime mode with trace, all statements are executed first within the task cycle. Then, data recording starts with value storage including time stamps. These time stamps are relative and refer to the start time of the data recording. The data yields a discrete time signal and CODESYS displays its course in the trace editor.
A sample (data record) is composed of the value and the time stamp. The runtime system writes the samples to a buffer with a definable size. CODESYS requests the data, saves it in the trace editor buffer, and displays it in the trace diagram as a function of time. You can monitor the value history of the configured variables continually because CODESYS displays the latest data.

You can trigger the data recording. When this happens, the application saves the data from the time of the trigger and CODESYS displays the data at the time of the trigger.
The configuration and the display of a trace are possible in the CODESYS project by means of trace objects in the trace editor. There are the following two object types:
- "Trace": Inserted below the IEC application in the device tree. This kind of object always contains a purely application-specific trace configuration. You can download this trace configuration to the controller and run it with the application.
- "DeviceTrace": Inserted below the device object in the device tree. If the PLC supports a trace manager, then you can use one or more "DeviceTrace" objects to access one or more traces that are running on the controller. These can be both application-specific or controller-specific traces. For example, a controller can support traces for recording the processor load. Menu commands allow for access from the CODESYS project to the trace manager in the device.
Access to the trace manager from IEC code is possible by means of the functions from the library Cmp TraceMgr. library. For more information, refer to the library documentation.

\section*{NOTICE!}

A running data recording with trace can lead to a significant increase in the cycle time of the IEC task.

Runtime system component CmpTraceMgr, "Trace manager"

\section*{NOTICE!}

Data recording with trace also continues running after logging out of the device.

The device description of a runtime system with trace manager includes the tracemanager entry in the TargetSettings section.
In this case, CODESYS transfers only the trace configuration when downloading the application to the PLC. When you start the trace, the application interprets the configuration on the RTS by means of the trace manager, executes the data recording, and buffers the data sets on the PLC.
The CmpTraceMgr runtime system component provides extended functionality, as compared to data recording with IEC code.
Data recording is therefore possible as follows:
- Parameters on the PLC (for example, the processor load (cpuload, plcload), or the temperature curve of a CPU or a battery). The measurement of the processor load per CPU core (cpuload) is interesting for multicore controllers.
- Device signals (for example, the current path of a drive)
- System variables of another runtime system component

You can configure parameters like IEC variables in the "Trace Configuration" dialog of the "Variable Settings".
The display of traces that run on the controller is possible in the trace editor of a DeviceTrace object.

\section*{See also}
- \# Chapter 6.4.1.13.3.5 "Accessing All Traces on the Controller" on page 2014
- \& Chapter 6.4.1.21.2.7 "Object 'DeviceTrace"' on page 2426
- \(\#\) Chapter 6.4.1.21.4.16.2 "Dialog 'Trace Configuration"' on page 2804

Data recording after triggering

To monitor data that depends on an event or a condition, you can free the data recording that depends on a trigger. At runtime, the application checks whether the event has occurred or the the condition is fulfilled, and then it buffers the data accordingly.
The trace configuration enables triggering by:
- a trigger variable that maps the event
- a condition as expression
- a combination of trigger variable and condition

Saving samples You can save samples from the development system to a file. The file can also include the trace to a file configuration.

Table 385: Possible file formats
\begin{tabular}{|c|c|c|}
\hline File Extension & File type & Description \\
\hline *.trace: & "Trace file" & Contains the samples and the trace configuration in XML format. You can execute the "Load Trace" command to load the file to the trace editor when offline and analyze the samples without a controller. \\
\hline *.txt & "Text File" & Contains the samples in ASCII format. You can edit the file with an external tool. \\
\hline *.trace.csv & "Trace dump" & \begin{tabular}{l}
File in CSV format includes the trace configuration and optional samples. \\
You can create the file by clicking "Export Symbolic Trace Config". You can transfer the file to the controller and load it to the application. Then you can execute the "Load Trace" command in CODESYS to display this in the trace editor. \\
You can also click "Trace \\
\(\rightarrow\) Save Trace" and select the *. trace. csv file format. You can transfer the file to the controller and load it with an HMI for analysis.
\end{tabular} \\
\hline *.traceconfig & "Symbolic trace configuration" & Contains the trace configuration CSV format. You can create the file by clicking "Export Symbolic Trace Config". The CmpTraceMgr runtime system component can read the file. \\
\hline
\end{tabular}

See also
- \({ }^{*}\) Chapter 6.4.1.21.3.22.15 "Command 'Save Trace"" on page 2741
- „y Chapter 6.4.1.21.3.22.8 "Command 'Load Trace"' on page 2737
- \(\xi^{\bullet}\) Chapter 6.4.1.21.3.22.7 "Command 'Export Symbolic Trace Config'" on page 2735
- \(\&\) Chapter 6.4.1.21.2.27 "Object 'Symbol Configuration"' on page 2523
* Chapter 6.4.1.21.2.31 "Object 'Trace'" on page 2541

Chapter 6.4.1.21.2.7 "Object 'DeviceTrace'" on page 2426

\section*{Getting started}

\section*{Program}

PLC_PRG PROGRAM PLC_PRG
VAR
iVar : INT;
rSin : REAL;
rVar : REAL;
END VAR
```

iVar := iVar + 1;
iVar := iVar MOD 33;
rVar := rVar + 0.1;
rSin := 30 * SIN(rVar);

```
\(\square\) Requirement: The application is running the PLC_PRG program on the controller.
1. In the device tree, select the application and add a new trace object by clicking "Project \(\rightarrow\) Add Object".
\(\Rightarrow\) The respective trace editor opens with the commands available in the "Trace" menu.
2. Click "Trace \(\rightarrow\) Configuration".
\(\Rightarrow\) The "Trace Configuration" dialog box opens.
3. Select a task for running the trace feature. Normally this is the same task that is running in PLC_PRG.
4. Click "Add Variable" to add an entry to the tree view of the trace configuration and assign an IEC variable (for example, PLC_PRG.rSin).
5. Click "Trace \(\rightarrow\) Download Trace".
\(\Rightarrow\) CODESYS loads the trace configuration to the controller. The application starts recording data and transmits the data to CODESYS, where it is displayed in the trace diagram as a graph. Commands are provided for navigating through the samples and controlling the data recording.

\section*{Example}

Trace the sine- The PLC_PRG program is running on the controller. When you follow the instructions for shaped data of "Getting S̄tarted", CODESYS displays the following trace diagram.
the IEC vari-
able \({ }_{\text {PLC_PRG.rSin }}\)

- (1) : "Configuration"
- (2) : "Add Variable"

See also
- yy Chapter 6.4.1.21.2.31 "Object 'Trace'" on page 2541

\section*{Creating trace configuration}

For a complete trace configuration, specify at least one task and one variable. In order to trigger the data recording, activate the trigger option and select a trigger variable or specify a recording condition.

\section*{See also}
- \(\quad\) Chapter 6.4.1.21.4.16.1 "Dialog 'Advanced Trace Settings'" on page 2803
- \(\quad\) Chapter 6.4.1.21.4.16.2 "Dialog 'Trace Configuration'" on page 2804
- \(\quad 4\) Chapter 6.4.1.21.4.18 "Dialog Box 'Advanced Trend Settings'" on page 2810
- \(\quad\) 2 Chapter 6.4.1.13.4.3 "Configuring trend recording" on page 2018

\section*{Assigning a task}

In this task, the data recording is executed in runtime mode. Usually the same task is selected where the variables are written.
1. Double-click the trace object.
\(\Rightarrow\) The trace editor opens with the commands available in the "Trace" menu.
2. Click "Trace \(\rightarrow\) Configuration".
\(\Rightarrow\) The "Trace Configuration" dialog opens. In the tree view "Trace Record", the top item is selected and the subdialog "Record Settings" is shown on the right.
3. Click the "arrow down" symbol ( \(\boldsymbol{*}\) ) in the "Task" drop-down list.
\(\Rightarrow\) The drop-down list opens with all tasks that are available throughout the application.
4. Select a task for the trace.

See also
- \(\quad\) Chapter 6.4.1.21.4.16.2 "Dialog 'Trace Configuration'" on page 2804

\section*{Configuring a trace variable}
1. Double-click the trace object.
\(\Rightarrow\) The trace editor opens. The commands of the "Trace" menu are available.
2. Click "Add Variable".
\(\Rightarrow\) The "Trace Configuration" dialog opens. The subdialog "Variable Settings" is displayed on the right.
3. Click ... in the input field of the "Variable" setting and select a trace variable in the "Input Assistant" dialog.
\(\Rightarrow\) The variable is configured for data recording. The trace record tree and the display tree were extended by the variable.
4. Click the "Add Variable" link.
\(\Rightarrow\) The trace record tree and the display tree receive a new variable. The settings of the variables are available on the right.
5. Select a trace variable.
6. Click "OK" to close the dialog.
\(\Rightarrow\) The variables are trace variables and are displayed in the trace variable list.

\section*{Deleting a trace} variable
1. Double-click the trace object.
2. Click a variable in the trace record tree.
3. Click the "Delete Variable" command or press [Del].
4. Click "OK" to close the dialog.
\(\Rightarrow\) The variable is removed from the trace variable list.

\section*{Tracing a parameter}
1. Double-click the trace object.
2. Click "Add Variable".
\(\Rightarrow\) The "Trace Configuration" dialog opens. The subdialog "Variable Settings" is displayed on the right.
3. Click \(\boldsymbol{*}\) (right of the "Variable" setting, left of the input field).
4. Select the "Parameter" option in the drop-down list.
5. Click ... and select a parameter from the "Input Assistant" dialog.
6. Configure how the parameter is displayed.
7. Click "OK" to close the dialog.
\(\Rightarrow\) The parameter will be traced and displayed in the trace variable list.

\section*{Configuring a trigger}
1. Double-click the trace object.
\(\Rightarrow\) The trace editor opens with the commands available in the "Trace" menu.
2. Click "Trace \(\rightarrow\) Configuration".
\(\Rightarrow\) The "Trace Configuration" dialog opens. The subdialog "Record Settings" is displayed on the right.
3. Select the "Enable trigger" check box.
4. Select the task in which the trend record is to be executed.
5. Select a variable from the "Trigger Variable" field.
6. Click "OK" to close the dialog.
\(\Rightarrow\) The data recording will be triggered.

The trigger time is displayed as a black line in the diagram in runtime mode.
1. Download the application and start it.
2. Click "Trace \(\rightarrow\) Download Trace".
\(\Rightarrow\) The trace configuration is loaded. After triggering, the runtime system saves the value graph of the trace variables. The data is displayed in the trace editor. The trigger time is displayed as a black line in the diagram.

\section*{Configuring the display of the time axis}
1. Double-click a trace object.
2. Click the "Configuration" link above the configuration tree. \(\Rightarrow\) The "Trace Configuration" dialog opens.
3. Select "Time axis" in the display tree (below "Presentation (Diagrams)").
\(\Rightarrow\) The display settings of the time axis are shown on the right.
4. Edit the presets and click the "Preview" link.
\(\Rightarrow\) The changes are seen in the coordinate system preview.
5. Click " \(Y\)-axis" in the display tree. The " \(Y\)-axis" item is below every configured diagram. Therefore, the display of the value axis is set for each diagram.
\(\Rightarrow\) The subdialog "Display Settings" of the selected axis is displayed on the right.
6. Change the preset value.
\(\Rightarrow\) The changes are applied in the coordinate system preview.
7. Click OK to close the "Trace Configuration" dialog.
\(\Rightarrow\) The display changes are visible in the affected diagrams.

\section*{Configuring the display of the trace variable}
1. Double-click a trace object.
2. Click the "Configuration" link.
\(\Rightarrow\) The "Trace Configuration" dialog opens.
3. Select a variable below "Trace Record".
\(\Rightarrow\) The subdialog "Variable Settings" of the selected variable is displayed on the right.
4. Change a setting, for example the "Line type".
5. Click "OK" to close the dialog.
\(\Rightarrow\) The display changes are visible in the affected diagrams.

\section*{Configuring the} buffer for data on the runtime system
1. Double-click a trace object.
2. Click "Trace \(\rightarrow\) Configuration".
\(\Rightarrow\) The "Trace Configuration" dialog opens. The subdialog "Record Settings" is displayed on the right.
3. Click "Advanced".
\(\Rightarrow\) The "Advanced Trace Settings" dialog opens.
4. Change the setting "Measure in every n-th cycle" or "Recommended runtime buffer size (samples)".
5. Click "OK" to close the dialog.
\(\Rightarrow\) The buffer settings are reconfigured. It is applied after the trace configuration is loaded to the RTS the next time.

Editing the trace configuration in runtime mode

Requirement: The application is running on the controller and a trace configuration is loaded.
1. Double-click a variable in the trace record tree.
\(\Rightarrow\) The "Trace Configuration" dialog opens.
2. Change the color, for example
\(\Rightarrow\) The variable is displayed in the new color in the affected diagrams without interrupting the execution of the application.

If you change essential settings, for example a trace variable, then you must download the trace configuration to the controller again.

\section*{Operating the data recording}

Use menu commands for controlling how data is recorded.
Requirement: The application is loaded on the runtime system and a trace is configured.

\section*{Menu commands}
- "Trace \(\rightarrow\) Download Trace"
- "Trace \(\rightarrow\) Start Trace"
- "Trace \(\rightarrow\) Stop Trace"
- "Trace \(\rightarrow\) Reset Trigger"

See also
- ⓨ Chapter 6.4.1.21.3.22.6 "Command 'Download Trace"' on page 2734
- \(\xi^{2}\) Chapter 6.4.1.21.3.22.16 "Command 'Start Trace"' on page 2741
- \(y^{*}\) Chapter 6.4.1.21.3.22.17 "Command 'Stop Trace'" on page 2741
- \({ }^{\circ} \gg\) Chapter 6.4.1.21.3.22.13 "Command 'Reset Trigger'" on page 2740

\section*{Accessing All Traces on the Controller}

If the controller supports the runtime system component CmpTraceMgr (Trace Manager), then you can access all traces from a CODESYS project which are running on the controller. In addition to application-related traces that capture the values of IEC variables, these can also be entirely controller-specific traces (for example, for recording device signal values or the CPU load).
For each trace running on the controller that you want to present in your project, you have to insert an individual "DeviceTrace" object in the device tree.

In order to show a trace from the device in this object, the connection to the PLC has to be configured correctly ("Communication Settings"). Then use one of the following menu commands:
- "Trace \(\rightarrow\) Upload Trace": Establishes the connection to the PLC and opens the "Online List" dialog for selecting a trace from the controller.
- "Trace \(\rightarrow\) Online List": Available in online mode only: Also opens the "Online List" dialog.

Now the trace uploaded from the controller can be started and traced in the editor of the DeviceTrace object. The configuration of the presentation (colors, labels, etc.) is the same as with traces for application variables configured in the project.

\section*{NOTICE!}

\section*{Closing the DeviceTrace editor terminates the connection to the controller.}

Please note that the connections to the controller is also terminated when the last open "DeviceTrace" editor is closed. In order for device traces to be displayed again in the project, you have to reload them into the "DeviceTrace" objects.
At this time, closing the editor is also the recommended procedure for deliberately terminating the connection to the controller. Logging out is not enough for this.

\section*{See also}
- \# Chapter 6.4.1.21.4.16.2 "Dialog 'Trace Configuration'" on page 2804
- \& Chapter 6.4.1.21.2.7 "Object 'DeviceTrace"' on page 2426
- « Chapter 6.4.1.21.3.22.19 "Command 'Upload Trace"' on page 2742
- \(\Rightarrow\) Chapter 6.4.1.21.3.22.12 "Command 'Online List"' on page 2739

Displaying the CPU load with DeviceTrace objects in the CODESYS project (example)

Requirement: The PLC device supports the Trace Manager. For the example described here, this is CODESYS Control Win V3. The device provides traces of the individual CPU loads (CpuLoad), as well as traces of the CPU load caused by the runtime system (PlcLoad). The possible display of the CPU load in the project can be helpful when using multicore functionality.
1. In the project, define the "Communication Settings" for the controller.
2. Select the PLC entry in the device tree and add a "DeviceTrace" object.
3. Rename "DeviceTrace" to "Trace_PlcLoad" ("Properties").
4. Set the focus in the trace editor and click "Trace \(\rightarrow\) Upload Trace". \(\Rightarrow\) The connection to the controller is established and the "Online List" dialog opens.
5. Select the "PIcLoad" entry in the dialog and click "Upload". Click OK to close the dialog.
\(\Rightarrow\) Multiple trace views open in the trace editor to show the CPU load in the runtime system. There are the traces for the particular CPUs and one trace for the average value. The following text appears for each: "No samples have been recorded."
6. Click "Trace \(\rightarrow\) Start Trace".
\(\Rightarrow\) The trace recording for the four parameters is displayed.
7. If you also want to display the traces for the CpuLoad per CPU with their average value in the project, then insert another "DeviceTrace" object into the device tree. Name it "Trace_CpuLoad" for example. Load and start the traces for "CpuLoad" in the editor as described above.
\(\Rightarrow\) Now you can monitor all traces in the project:
. If you want to change the appearance of the presentation, then click "Configuration" in the respective trace editor window to access the configuration dialogs. You can use these dialogs (except variable assignments) in the same way as for an IEC variable trace created in a project.
9. To disconnect from the controller, close all open DeviceTrace editor windows. If you are logged in to the device, then logging out is enough to terminate the connection.

See also
- \(\Leftrightarrow\) Chapter 6.4.1.21.2.8.3 "Tab 'Communication Settings'" on page 2427

\section*{Navigating into trace data}

Use menu commands to navigate the data in the trace diagram.
Requirement: The application is in online mode.
Menu commands
- "Trace \(\rightarrow\) Cursor"
- "Trace \(\rightarrow\) Mouse Zooming"
- "Trace \(\rightarrow\) Reset View"
- "Trace \(\rightarrow\) AutoFit"
- "Trace \(\rightarrow\) Compress"
- "Trace \(\rightarrow\) Stretch"
- "Trace \(\rightarrow\) Convert to Single-Channel"
- "Trace \(\rightarrow\) Convert to Multi-Channel"

See also
- \(\xi^{*}\) Chapter 6.4.1.21.3.22.5 "Command 'Cursor'" on page 2733
- \({ }^{*}\) Chapter 6.4.1.21.3.22.9 "Command 'Mouse Zooming'" on page 2737
- \({ }^{4}\) Chapter 6.4.1.21.3.22.14 "Command 'Reset View'" on page 2740
- \(\xi^{*}\) Chapter 6.4.1.21.3.22.2 "Command 'AutoFit"' on page 2733
- ※ Chapter 6.4.1.21.3.22.3 "Command 'Compress'" on page 2733
- \({ }^{4}\) Chapter 6.4.1.21.3.22.18 "Command 'Stretch'" on page 2742
- \(\quad y^{*}\) Chapter 6.4.1.21.3.22.10 "Command 'Convert to Multi-Channel"' on page 2737
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.22.11 "Command 'Convert to Single-Channel"' on page 2738

\section*{Managing trace}

Use menu commands to load and save traces in various formats.
Menu commands
- "Trace \(\rightarrow\) Save Trace"
- "Trace \(\rightarrow\) Load Trace"
- "Trace \(\rightarrow\) Export Symbolic Trace Config"

See also
- \({ }^{4}\) Chapter 6.4.1.21.3.22.15 "Command 'Save Trace"" on page 2741
- 乡y Chapter 6.4.1.21.3.22.8 "Command 'Load Trace"' on page 2737
- \(\sum^{\Perp}\) Chapter 6.4.1.21.3.22.7 "Command 'Export Symbolic Trace Config'" on page 2735

\section*{Showing statistics}

CODESYS evaluates and displays the recorded data with an option of saving the data to the clipboard. Click "Trace \(\rightarrow\) Statistics".

See also
- \({ }^{\circ}\) Chapter 6.4.1.21.3.22.20 "Command 'Statistics'" on page 2742

\subsection*{6.4.1.13.4 Data Recording with Trend}

\section*{General}

When you want to monitor the development of data over a long period of time for the purpose of reading a trend, you can save the data with "Trend Recording". You can configure any number of variables or parameters to save their values in a persistent database. This database is located on the PLC and is populated continually at runtime.
```

- Application
{1] Library Manager
㖓 PLC_PRG (PRG)
Task Configuration
MainTask
田 PLC_PRG
-% TrendRecordingTask
转 VisuTrendStorageAccess,GlobalInstances,g_TrendRecordingManager.CyclicCall
- % VISU_TASK
田园 VisuElems.Visu_Prg
Trend Recording Manager (2
每
TrendRecording
_A 3
Visualization Manager
Trend_A
Trend recording comprises the following objects：
－（1）：＂Trend recording task＂of type＂Task＂
－（2）：Object of type＂Trend Recording Manager＂
－（3）：Object of type＂Trend Recording＂

```

\section*{NOTICE！}

\section*{Timeout for trend recording}

During a trend recording，it can happen that the application task triggers a timeout that is caught with an exception when transitioning from＂Running＂to ＂Stop＂．Causes can be that file operations with the SQLite database are taking too long or that too many variables are being recorded．This usually happens on a target device with weak performance

You can avoid the occurrence of an exception：
－Configure the trend recording with less memory demand so that the amount of data that is stored is adapted to the target system．
－Reduce the number of variables．

To display the collected data，you design a visualization with a＂Trend＂element． This kind of visualization accesses the database for visualizing the data．

See also
－＊y Chapter 6．4．1．21．2．32＂Object＇Trend Recording Manager＇＂on page 2545
－\(\geqslant\) Chapter 6．4．1．21．2．33＂Object＇Trend Recording＇＂on page 2545
－\({ }^{4}\) Chapter 6．4．1．21．2．34＂Object＇Trend Recording Task＇＂on page 2548

\section*{Getting started with trend recording}

To execute trend recording on a runtime system，you need an application with a＂Trend Recording Manager＂object that contains at least one＂Trend Recording＂object．Then you can configure a database on the runtime system and the data buffering．
1. Add a "Trend Recording Manager" object below your application.
2. Select the "Trend Recording Manager" object and click "Add Object \(\rightarrow\) Trend Recording". Type a name in the "Add Trend Recording" dialog box.
\(\Rightarrow\) CODESYS creates the object. The editor opens.
3. Type a task in "Record Settings".
4. Click "Add Variable".
\(\Rightarrow\) CODESYS adds another variables. The blank settings open in the "Variable Settings" to the right of the tree view.
5. Select a valid IEC variable from the "Variable" field.
\(\Rightarrow\) The IEC variable is configured for trend recording.
6. Build the application.
7. Download the application to the controller and click "Start".
\(\Rightarrow\) The application records data in runtime mode and saves it to a database.

See also
- \({ }^{*}\) Chapter 6.4.1.21.2.33 "Object 'Trend Recording"' on page 2545
- \(\&\) Chapter 6.4.5.13.2 "Getting Started with Trend Visualization" on page 2928

\section*{Configuring trend recording}

You can configure a database on the runtime system and the data buffering.

\section*{NOTICE!}

\section*{Timeout for trend recording}

During a trend recording, it can happen that the application task triggers a timeout that is caught with an exception when transitioning from "Running" to "Stop". Causes can be that file operations with the SQLite database are taking too long or that too many variables are being recorded. This usually happens on a target device with weak performance.

You can avoid the occurrence of an exception:
- Configure the trend recording with less memory demand so that the amount of data that is stored is adapted to the target system.
- Reduce the number of variables.

Assigning tasks In this task, the runtime system records the trend.

1. Double-click a "Trend Recording" object in the device tree.
\(\Rightarrow\) The respective editor opens. In the tree view of the trend configuration, the top entry is selected, and on the right you see the current configuration in "Record Settings".
2. Click the "arrow down" symbol ( \(\boldsymbol{*}\) ) in the "Task" drop-down list.
\(\Rightarrow\) The drop-down list opens with all tasks that are available throughout the application.
3. Select a task for trend recording.

See also
- \({ }^{*}\) Chapter 6.4.1.21.2.33 "Object 'Trend Recording'" on page 2545

Adding IEC variables

\section*{NOTICE!}

The number of variables is limited for trend recording. You can change this number in the "Trend storage" dialog.
1. Double-click a "Trend Recording" object in the device tree.
\(\Rightarrow\) The respective editor opens. In the tree view of the trend configuration, the top entry is selected, and on the right you see the current configuration in "Record Settings".
2. Right-click an entry in the tree view.
3. Click "Add Variable".
\(\Rightarrow\) CODESYS adds another variables. The blank settings open in the "Variable Settings" to the right of the tree view.
4. Select a valid IEC variable from the "Variable" field.
\(\Rightarrow\) The IEC variable is configured for trend recording.
5. Configure how the variable is displayed in the trend diagram.
6. Configure how the alert color is displayed in the trend diagram.

See also
- \(\quad \stackrel{y}{c}\) Chapter 6.4.1.21.2.33 "Object 'Trend Recording'" on page 2545
- \({ }^{*}\) • Chapter 6.4.1.21.4.17 "Dialog Box 'Trend storage"" on page 2809

\section*{Removing variables from the configuration}
1. Double-click a "Trend Recording" object in the device tree.
2. Click a variable in the tree view of the configuration.
3. Click "Delete Variable" or press [Del].

Starting condi- You can configure conditional trend recording for execution. Configuration is not possible when tional trend recording
depending on triggering. For that you need a "Trace" object.
1. Double-click a "Trend Recording" object in the device tree.
2. Click the top node in the tree view of the trend configuration.
\(\Rightarrow\) The name of the trend configuration is selected and on the right you see the current configuration in "Record Settings".
3. Assign a Boolean variable, an access to a bit, or a property to the "Record condition" field.
\(\Rightarrow\) When the application is in runtime mode, data is recorded only if the value is TRUE.

See also
- 㘯 Chapter 6.4.1.13.3 "Data Recording with Trace" on page 2007

\section*{Adding param-} eter
1. Double-click a "Trend Recording" object in the device tree.
\(\Rightarrow\) The respective editor opens. In the tree view of the trend configuration, the top entry is selected, and on the right you see the current configuration in "Record Settings".
2. Right-click an entry in the tree view.
3. Click "Add Variable".
\(\Rightarrow\) CODESYS inserts a new variable. The blank settings open in the "Variable Settings" to the right of the tree view.
4. Click the "down" symbol ( \(\boldsymbol{\nabla}\) ) to the right of the "Variable" label.
5. Select "Parameter" from the drop-down list.
6. Click ... and select a parameter from the "Input Assistant" dialog.
7. Configure how the parameter is displayed in the trend diagram.
8. Configure how the alert color is displayed in the trend diagram.

See also
- \({ }^{4}\) Chapter 6.4.1.21.2.33 "Object 'Trend Recording"' on page 2545

\section*{Configuring} data buffering on the RTS
1. Double-click a "Trend Recording" object in the device tree.
\(\Rightarrow\) The respective editor opens. In the tree view of the trend configuration, the top entry is selected, and on the right you see the current configuration in "Record Settings".
2. Click "Trend Storage".
\(\Rightarrow\) The "Trend Storage" dialog opens.
3. Now you can change the settings.

See also
- \({ }^{4}\) Chapter 6.4.1.21.4.17 "Dialog Box 'Trend storage'" on page 2809

\section*{Configuring additional buffering}
1. Double-click a "Trend Recording" object in the device tree.
\(\Rightarrow\) The respective editor opens. In the tree view of the trend configuration, the top entry is selected, and on the right you see the current configuration in "Record Settings".
2. Click "Advanced".
\(\Rightarrow\) The "Advanced Trace Settings" dialog opens.
3. Now you can change the settings.

See also
- \({ }^{\text {² }}\), Chapter 6.4.1.21.4.18 "Dialog Box 'Advanced Trend Settings'" on page 2810

See also
- \({ }^{\bullet}\) Chapter 6.4.1.13.3.3 "Creating trace configuration" on page 2010

\subsection*{6.4.1.13.5 Monitoring tasks}

In online mode, you can display some statistical values of the tasks in the runtime system. This information is very useful for testing clock cycles or solving problems in the runtime performance.
1. Switch to online mode.
2. Select the "Task Configuration" object in the device tree.

Click "Project \(\rightarrow\) Edit Object".
\(\Rightarrow\) The task configuration opens in the editor.
3. Click the "Monitor" tab.

See also
- \(\quad\) Chapter 6.4.1.21.2.29.4 "Tab 'Monitor"' on page 2536

\subsection*{6.4.1.13.6 Reading the PLC log}

CODESYS provides the capability to display the events and error messages logged in the controller.

See also
- ※ Chapter 6.4.1.21.2.8.9 "Tab 'Log'" on page 2437

Reading the log Requirement: The controller is running.
1. Select the controller in the device tree.
2. Choose the command "Project \(\rightarrow\) Edit Object".
\(\Rightarrow\) The device editor opens.
3. Choose the tab "Log".
4. Click on to update the view.
\(\Rightarrow\) A connection to the controller is established. The controller in the device tree is highlighted in green.

All controller log information are displayed.
5. Click on \(X\) to delete the current list.
6. Filter the view by clicking on the desired category (for example "Information").
7. Save the log entries. Click on and choose a file name.

See also
- \({ }^{〔}\) Chapter 6.4.1.21.2.8.9 "Tab 'Log'" on page 2437

\subsection*{6.4.1.13.7 Using PLC shell for requesting information}

The "PLC shell" in CODESYS is a text-based control monitor (terminal) on a tab of the device editor. There you can enter commands for the request of specific information from the controller, as well as execute actions like starting, stopping or downloading applications. Also a description on the meaning and syntax of the possible commands you can get directly via the PLC shell.

Requesting
information about the application on the controller

Requirement: Your project is connected with a controller; Example: CODESYS Control Win V3, on which an application App1 is running.
1. Open the device editor double-clicking on the object CODESYS Control Win V3 in the device tree, and activate tab "PLC Shell".
\(\Rightarrow\) The tab shows an empty output data window. Below there is an entry field for a command.
2. Click button ....
\(\Rightarrow\) The "Insert Standard Command" dialog appears with a list of commands.
3. Choose command "?" and click button "Execute".
\(\Rightarrow\) The dialog closes and in the output data window you see a list of the supported commands and their possible parameters. Each the syntax for how to enter the command is displayed.
4. Click again ... and choose command "pid". In the input assistant supplement the command as follows: pid App1. Press the Enter key.
\(\Rightarrow\) In the output data window the following gets displayed (the GUIDs are just exmples):
```

pid App1
Project Identification
Application: App1
Code GUID:0x08a893c0
Data GUID:0x762d0e90

```
5. Click button \(\boldsymbol{v}\) in the command line.
\(\Rightarrow\) Command pid App1 is added to the history of already entered commands.

See also
- «y Chapter 6.4.1.21.2.8.11 "Tab 'PLC Shell"' on page 2441

\subsection*{6.4.1.13.8 PLC operation control via system variables}

\section*{CAUTION!}

You are responsible for runtime system services being enabled under safe application conditions and disabled only under critical conditions.

At runtime, the state of an application or facility can become sensitive and disruptive actions can endanger the entire machine or facility. However, in this state you can suppress certain commands and prevent dangerous actions. The "PlcOperationControl" function block and "Component Manager" library are provided for this purpose.

Examples of CODESYS commands that can suppress operations when executed:
- "Online Change", "Download"
- "Enable Breakpoint"
- "Reset Application", "Stop Application"
- "Transmit Data"
- "Force Values", "Write Values"

In order that a backup solution is always in place, you are not permitted to suppress the "Reset origin" and "Delete" commands.
CODESYS will notify you if a currently disabled runtime system service is required when the application is in runtime mode. Then, you can respond with an appropriate countermeasure.

Function block This function block is used for enabling and disabling operations. PlcOperationC
ontrol for
operation con-
trol
Table 386: Property (PROPERTY)
\begin{tabular}{|l|l|l|l|}
\hline Name & Data Type & Initial value & Description \\
\hline \begin{tabular}{l} 
xDisableApplic \\
ationOnlineCha \\
nge
\end{tabular} & BOOL & FALSE & TRUE: Online change is suppressed. \\
\hline \begin{tabular}{l} 
xDisableApplic \\
ationDownload
\end{tabular} & BOOL & FALSE & TRUE: Download is suppressed. \\
\hline \begin{tabular}{l} 
xDisableApplic \\
ationStop
\end{tabular} & BOOL & FALSE & TRUE: Application stop is suppressed. \\
\hline \begin{tabular}{l} 
xDisableApplic \\
ationBP
\end{tabular} & BOOL & FALSE & TRUE: Setting breakpoints is suppressed. \\
\hline \begin{tabular}{l} 
xDisableApplic \\
ationWrite
\end{tabular} & BOOL & FALSE & \begin{tabular}{l} 
TRUE: Writing variables is suppressed. \\
This can also be suppressed via PLCHandler/lec- \\
VarAccess.
\end{tabular} \\
\hline \begin{tabular}{l} 
xDisableApplic \\
ationForce
\end{tabular} & BOOL & FALSE & TRUE: Forcing variables is suppressed. \\
\hline \begin{tabular}{l} 
xDisableApplic \\
ationReset
\end{tabular} & BOOL & FALSE & \begin{tabular}{l} 
TRUE: Resetting the application (not "Reset \\
origin") is suppressed.
\end{tabular} \\
\hline xDisableAll & BOOL & TRUE: All operations are suppressed. \\
\hline
\end{tabular}

Implementing operation control

\section*{Requirement}
- Compiler version >=3.4.3.0
- In the device description, the PLC operation control is enabled by system variables.
1. Declare an instance of the PlcOperationControl function block (for example,

PlcOpCtrl_Inst).
\(\Rightarrow\) PlcOpCtrl_Inst : PlcOperationControl;
2. Suppress a command by assigning the respective TRUE property (for example, "Stop Application".
```

A PlcOpCtrl_Inst.xDIsableApplicationStop := TRUE;

```

\section*{See also}
- \(\quad\) Chapter 6.4.1.21.3.7.6 "Command 'Online Change'" on page 2629
- \(\quad>\) Chapter 6.4.1.21.3.8.16 "Command 'Force Values'" on page 2649
* \(>\) Chapter 6.4.1.12.5 "Forcing and Writing of Variables" on page 1987

\subsection*{6.4.1.13.9 Backup and restore}

CODESYS and the CODESYS standard runtime systems (with version 3.5.8.0 and later) support backing up application-specific files on the PLC. You can execute the required actions in the "Backup and Restore" tab of the generic device editor.
A backup consists of creating and saving a file in zip archive that contains the applicationrelated files and an information file meta.info. This backup file has the extension TBF (="Target Backup File") and can be saved in the local file system or on the PLC.
The following applies when restoring the software status from the backup file:
- A dialog opens with a list of affected files on the PLC, and you can deactivate optional components.
- We highly recommend to set the application to STOP mode for backup or restore. A dialog prompt will open to warn you about this.
- The user interface is blocked when restoring to the PLC.
- Existing files are overwritten without warning.
- Existing boot applications are deactivated as soon as at least one new boot application is part of the restore.
See also
- \(\Leftrightarrow\) Chapter 6.4.1.21.2.8.6 "Tab 'Backup and Restore'" on page 2435

Creating backup Requirement: A project is open with an application that is running on the required device. In files addition, for this example an external file myExternalFile.txt is inserted as an object below the application. This file is downloaded to the PLC implicitly when downloading the application.
1. Open the device editor by double-clicking the device entry in the device tree. Click the "Backup and Restore" tab.
\(\Rightarrow\) The tab opens with a menu bar including the "Backup" and "Restore" menus.
2. In the "Backup" menu, select the "Read Backup Information from Device" item.
\(\Rightarrow\) If the PLC is not connected at the moment, then CODESYS opens a temporary connection to the device and reads the relevant files from the \(\$ P l c L o g i c \$\) directory of the PLC into a table in the lower part of the tabbed page. In this example, at least the following files will be listed: \$PlcLogic\$/Application/Application.app, \$PlcLogic\$/Application/ Application.crc, and \$PlcLogic\$/Application/myExternalFile.txt. In addition, other external, project-dependent files are listed, which have been inserted below the application in the device tree. Furthermore, the source code archive file \(\$ \mathrm{PlcLogic} /\) /Archive.prj is listed if you have set the project setting for this ("Implicitly at program download and online change") as the loading time.
3. In the table, clear the check box for the \$PlcLogic\$/Application// myExternalFile.txt file in the "Active" column.
4. Select "Save Backup File to Device" in the "Backup" menu.
\(\Rightarrow\) The "Save as" dialog opens. The file type is predefine as "Backup files (*.tbf)".
5. Select a location for the backup file and click "Save".

See also
- \(\quad\) Chapter 6.4.1.21.4.12.6 "Dialog 'Project Settings' - 'Source Download'" on page 2770

Restoring from Requirement: A project is open with a PLC device connected. A backup file is saved to the local backup files file system as described above.
1. Open the device editor by double-clicking the device entry in the device tree. Click the "Backup and Restore" tab.
Click "Restore \(\rightarrow\) Load Backup File from Disc".
\(\Rightarrow\) The default prompt opens for selecting a backup file tbf in the local file system.
2. Select the backup file and click "Open".
\(\Rightarrow\) The files are read from the backup file and shown in the table of the dialog below. The file \$PlcLogic\$/Application/myExternalFile.txt that was excluded in the backup is missing.
3. Click "Restore \(\rightarrow\) Restore Backup to Device".
\(\Rightarrow\) A dialog prompt opens with information about the actions when restoring.
4. Click "OK" to start restoring the files to the PLC file system.
\(\Rightarrow\) When restore is complete, you are prompted to restart the PLC in order to activate the loaded application.

\subsection*{6.4.1.14 Updating an Application on the PLC}

\subsection*{6.4.1.14.1 General}

CODESYS basically provides two options to transfer a modified application to the controller: download and online change.
A download results in a recompilation of the application. In that time, a syntax check is performed and application code is also created and downloaded to the controller. This leads to the running program being stopped. A download is the recommended method of data transfer because a defined starting state is always created due to the program stop and the reinitialization.

In the case of an online change, only the modified parts are downloaded again to the controller. A running program is not stopped for this. You should perform an online change only in the case of minor changes to the application. For extensive changes, the behavior of a program cannot be safely predicted. For more information, read the notes in the description of the "Online Change" command.
See also
- \(\quad\) Chapter 6.4.1.14.2 "Executing the online change" on page 2025
- \(\quad 4\) Chapter 6.4.1.14.3 "Execution of a download" on page 2026
- \(\quad\) Chapter 6.4.1.21.3.7.5 "Command 'Load'" on page 2628
- \(\quad \stackrel{y}{c}\) Chapter 6.4.1.21.3.7.6 "Command 'Online Change'" on page 2629

\subsection*{6.4.1.14.2 Executing the online change}

CODESYS automatically offers you an online change if you log in with an application that is already present on the controller, but has been changed since the last download in the programming system. With this procedure only the modified parts are reloaded to the controller. A running program on the controller is not stopped during the online change.
In the view "Memory reserve for online change", you can configure memory reserves for the online change for function blocks of a project. In this way, instance variables do not have to be moved to the memory after changes are made to a function block for an online change.

\section*{NOTICE!}

When carrying out the online change, pay attention to the notes in the description of the "Online Change" command.

Executing the online change when logging in

See also
- \(\quad\) Chapter 6.4.1.21.3.7.6 "Command 'Online Change'" on page 2629
- \(\quad\) Chapter 6.4.1.21.3.7.2 "Command 'Login'" on page 2624
- \(\quad \Rightarrow\) Chapter 6.4.1.11.5 "Generating Application Code" on page 1976

Requirement: The connection settings of the controller are correctly set. The applications in the project and on the controller are identical. The project on the controller is running. The application is logged out.
1. Change your application.
2. Click "Online \(\rightarrow\) Login".
\(\Rightarrow\) A dialog appears with the information that the application has been changed since the last download.
3. Click the "Details..." button
4. Check the details in the "Application information" tab.

If you have not generated any code since the last change, the command "Application is not up to date. Generate code now?" appears at the bottom edge of the dialog. In this case click this command.
\(\Rightarrow\) You are shown a comparison view of the objects (objects marked red are different).
5. Close the dialog.
6. Select the option "Login with Online Change" and click "OK".
\(\Rightarrow\) The change is loaded to the controller. The running program on the controller is not stopped while doing this. The application is logged in.

See also
- 㜽 "View 'Project Comparison' - 'Differences'" on page 2607

\section*{Execute online change in the logged-in state (online operation)}

Requirement: The connection settings of the controller are correctly set. The applications in the project and on the controller are identical. The project on the controller is running. The application is logged in.
1. Select an object in the device tree. It is best to select a POU or a GVL here.
2. Click "Project \(\rightarrow\) Edit Object (Offline)".
\(\Rightarrow\) The object opens in the editor.
3. Change the object. For example, you can declare a new variable or change a value assignment here.
4. Click "Online \(\rightarrow\) Online Change".
\(\Rightarrow\) A query will appear, asking whether you really want to execute the online change.
5. Click "Yes" to confirm the dialog.
\(\Rightarrow\) The change is loaded to the controller.

\subsection*{6.4.1.14.3 Execution of a download}

A download of the application causes a compilation of the active application. In the process, a syntax check is performed and application code is also created and loaded to the controller. A program running on the controller is stopped during the download.

\section*{NOTICE!}

During the download, pay attention to the notes in the description of the "Download" command.

See also
- \({ }^{\circledR}\) Chapter 6.4.1.21.3.7.5 "Command 'Load'" on page 2628
- \(\underbrace{*}\) Chapter 6.4.1.21.3.7.2 "Command 'Login'" on page 2624
- \(\Leftrightarrow\) Chapter 6.4.1.11.5 "Generating Application Code" on page 1976

Downloading Requirement: the connection settings of the controller are correctly set. The applications in when logging in
the project and on the controller are identical. The project on the controller is running. The application is logged out.
1. Change your application.
2. Select the command "Online \(\rightarrow\) Login"
\(\Rightarrow\) A dialog box appears with the information that the application has been changed since the last download.
3. Select the option "Login with download" and click on "OK".
\(\Rightarrow\) The running program on the controller is stopped and the change is loaded to the controller. The application is logged in.

Downloading in Requirement: the connection settings of the controller are correctly set. The applications in the logged-in state (online mode)
the project and on the controller are identical. The project on the controller is running. The application is logged in.
1. Select an object in the device tree. It is best to select a POU or a GVL here.
2. Select the command "Project \(\rightarrow\) Edit Object (Offline)"
\(\Rightarrow\) The object opens in the editor.
3. Change the object. For example, you can declare a new variable or change a value assignment here.
4. Select the command "Online \(\rightarrow\) Download".
\(\Rightarrow\) A query will appear, asking whether you really want to execute the download.
5. Confirm the dialog box with "Yes".
\(\Rightarrow\) The running program on the controller is stopped and the change is loaded to the controller.

\subsection*{6.4.1.15 Copying files to/from PLC}

In the generic "Files" tab of the device editor, you can copy files to and from the local file system and the controller.

Requirement: The vendor has unlocked the tab. In the device tree, the connection to the controller is configured. The device is running.
1. Double-click the PLC device object in the device tree to open the device editor.
2. Click the "Files" tab.
3. In "Host" | "Location" on the left part of the view, set the path in the local file system where files will be copied to and from. Example: D: \FileTransferWithPLC. If necessary, create a new directory by clicking the folder symbol ( \(\square\) ).
\(\Rightarrow\) The files and directories are shown like in a file manager. Click the refresh symbol ( \(\%\) ) to update the display.
4. In "Runtime" on the right side of the view, set the required directory for the data transfer in the same way.
\(\Rightarrow\) CODESYS shows the files on the controller.
5. Select the required files from the file system tree for the file transfer (multiple selection is possible). You can also select a directory for transferring all files in a folder.
6. Click the left and right arrow symbols ( \(\square \ll, \square>\) ) between the two parts of the view.
\(\Rightarrow\) CODESYS copies the selected files to the other file system immediately. If a file is not already available in the target directory, then it is created. If it is already available and not write-protected, then it is overwritten. Otherwise a message is shown.

See also
- \(\quad\) Chapter 6.4.1.21.2.8.8 "Tab 'Files'" on page 2437

\subsection*{6.4.1.16}

\section*{Syntax:}

\section*{Using the Command-Line Interface}

You can start the command line with the following options and arguments.
<folder>Automation Builder.exe --<option>

Paths or option parameters must be written inside straight quotation marks when they contain spaces, dashes, or slash marks.
```

Option -- CODESYS is started in the specified language.
culture (lan-
guage of the
user interface)
Syntax: --culture=<culture>
<Culture>: Typical language codes are as follows: de, en, fr, it, es, zh-CHS.

```
Example Starting CODESYS with the user interface in English:
    Automation Builder.exe --culture=en

\section*{See also}
- \({ }^{\circledR}\) Chapter 6.4.1.21.4.14.14 "Dialog 'Options' - 'International Settings'" on page 2790
\begin{tabular}{ll} 
Option -- & CODESYS is started directly with the specified profile. When you start CODESYS without this \\
profile & option, the "Select Profile" opens. \\
(CODESYS pro-
\end{tabular}\(\quad\)\begin{tabular}{ll} 
file) & \\
Syntax: & \\
& \\
& <-profile="<profile name>" \\
& \(\rightarrow\) About" name>: You have to specify the profile name exactly as it is displayed in the "Help
\end{tabular}

\section*{Example}
```

Automation Builder.exe --culture=de --profile="Automation Builder
2.5"

```

Option -- After CODESYS is started, the comparison of two CODESYS projects is begun immediately. compare (start Type the path of the project file as arguments after the option and then the path of the reference project compar- project. CODESYS starts and opens the "Project Comparison - Differences" view.
ison)
Syntax: --compare="<path of project file>" "<path of reference project file>"

\section*{Example}
```

Automation Builder.exe --compare "D:\proj\project1.project"
"D:\proj\project2.project"

```

See also
- \({ }^{〔}\) Chapter 6.4.1.21.3.5.21 "Command 'Compare"' on page 2607
\begin{tabular}{ll} 
Option -- & CODESYS is started and the specified project is opened. \\
project (open \\
CODESYS \\
project)
\end{tabular}\(\quad\)\begin{tabular}{ll} 
Syntax: & \begin{tabular}{l}
--project="<path of project file>" \\
\end{tabular}
\end{tabular}
Example \(\quad\) Open the test project:
Automation Builder.exe --culture=de --
project="D: \(\backslash\) projects \(\backslash t e s t\). project"

\section*{See also}
- \({ }^{4}\) Chapter 6.4.1.21.3.2.2 "Command 'Open Project"' on page 2553

Option -- CODESYS is started, the specified project archive is extracted, and the project is opened.
```

Example Extract the test.projectarchive and open the project in the development system:
Automation Builder.exe --
projectarchive="D:\projects\test.projectarchive"

```

\section*{See also}
- \(\quad\) Chapter 6.4.1.21.3.2.9 "Command 'Extract Archive"' on page 2558
Option -- \(\quad\) The specified script file is run by CODESYS.
runscript (run
script)

Table 387: Command-line options for--runscript
\begin{tabular}{|c|c|}
\hline --runscript="<scriptfile>.py" & CODESYS runs the <scriptfile>.py script file at startup. You have to provide the complete path of the script file. \\
\hline --scriptargs:'<arg1> <arg2> ... <argn>' & Use this option with the --runscript option. As a result, the arguments <arg1> ... <argn> are passed to the script. The arguments are passed to the Python variable sys.argv. \\
\hline --noUI & \begin{tabular}{l}
Use this option with the --runscript option. \\
The CODESYS user interface is not opened. CODESYS prints all errors, warnings, compiler reports, and command-line messages generated from the script. The script messages (1: Severity Text) can be separated from other messages (2: Severity FatalError, Error, Warning, Information) with the ">" operator.
\end{tabular} \\
\hline --enablescripttracing & Use this option with the --runscript option. As a result, each command of the script file is shown in the output. \\
\hline --textPrompts & \begin{tabular}{l}
Use this option with the --noUI option. As a result, message service methods and default dialogs are output in the command line for user input. \\
If you do not specify --textPrompts, then all message service prompts are confirmed automatically with default values.
\end{tabular} \\
\hline scriptdebugger \{="<debugger>"\} & \begin{tabular}{l}
Use this option with the --runscript option. It sets IronPython in debug mode so that external debuggers can be used to debug Python scripts. The following values are defined for <debugger> (uppercase/lowercase is irrelevant). \\
- auto: Automatically detects if a debugger is included in every script for the current process. At this time, only .NET-based debuggers can be detected automatically. A detected debugger overwrites the --enablescripttracing flag. \\
. NET: Activates debugging for .NET-based debuggers, such as "Python Tools for Visual Studio" (PTVS) and SharpDevelop. With this option, a debugger can also be included in running scripts, as opposed to "auto". \\
Note: This is currently the default value when -scriptdebugger is used without providing a value. \\
disabled: Deactivates debugging and automatic detection. \\
script: Switches the IronPython script engine to debug mode for activating the debugging for set-trace debuggers. The script itself must connect to and disconnect from the debugger. \\
- tracing: Activates the simple integrated script tracing mode and deactivates the automatic detection (same as the option -scripttracing). \\
- \$absolute_path.py\$: Absolute path to a Python script that initiates the connection to a Python-based debugger. The IronPython script engine is switched to debug mode for allowing the debugging for set-trace debuggers. This script is run one time during the initialization and should define the following non-parameterized functions: \\
scriptdebuggersetup is run immediately before running the user script to establish the connection to the debugger. \\
scriptdebuggershutdown is called immediately after running the user script or when the script engine is downloaded and should close the connection to the debugger.
\end{tabular} \\
\hline
\end{tabular}

\section*{Examples of using transfer parameters in script files with 'sys.argv'}
```

start /b /wait Automation Builder.exe
--runscript="D:\Script\ArgvAnd__main__Test.py"
--scriptargs:'username password 3.14 "path=\"C:\temp<br>"'

```

Script file: ArgvAnd__main__Test.py
from __future__ import print_function
import sys
print("sys.argv: ", len(sys.argv), " elements:")
for arg in sys.argv:
    print(" - ", arg)
print()
print ("__name__:
\(\qquad\)
Output result: stdout:
```

sys.argv: 6 elements:

```
- D:\TestScripts \(\backslash\) ArgvAnd__main__Test.py
- username
- password
- 3.14
- path= "C:temp"
name__ main

For more information about the \(\qquad\) name global variable, see the Python documentation.

\section*{Examples of the message output}
```

start /b /wait Automation Builder.exe --
runscript="D:\Script\AmpelTest.py" --noUI 1>ScriptMessages.txt

```

CODESYS passes all messages that are generated by the script to the ScriptMessages.txt file. Other messages are printed to the command line.
start /b /wait Automation Builder.exe --
runscript="D:\Script\AmpelTest.py" --noUI \(2>\) NUL
CODESYS suppresses all messages, except for script messages. The script messages are printed to the command line.
```

Example of The following initdebug.py script was tested successfully with pydevd-based debuggers,
option --script-
debugger command line:

```
```

--profile="Fanta Development Build" --

```
--profile="Fanta Development Build" --
scriptdebugger="D:\test\charmdebug\initdebug.py"
scriptdebugger="D:\test\charmdebug\initdebug.py"
File: initdebug.py:
```

```
from _future_ import print_function
```

from _future_ import print_function
from _future_ import unicode_literals
from _future_ import unicode_literals
import sys
import sys
sys.path.append(r"D:\test\Env2\Lib\site-packages\pycharm-debug.egg")
sys.path.append(r"D:\test\Env2\Lib\site-packages\pycharm-debug.egg")
import pydevd
import pydevd
def scriptdebuggersetup():
def scriptdebuggersetup():
pydevd.settrace('localhost', port=51234, stdoutToServer=True,
pydevd.settrace('localhost', port=51234, stdoutToServer=True,
stderrToServer=True)
stderrToServer=True)
def scriptdebuggershutdown():
def scriptdebuggershutdown():
pydevd.stoptrace()

```
pydevd.stoptrace()
```


## See also

- http://docs.python.org/tutorial/modules.html

Option -- If you add this option after the option --compare <project1> <project2>, then whiteignorewhitesp space is ignored in the project comparison. Note that semantically relevant spaces, for example ace (ignore whitespace in project comparison) in STRING literals, are still taken into account.

Syntax
--compare="<path of project file>" "<path of reference project file>"
--ignorewhitespace="true"|"false"

## Example

```
Automation Builder.exe --compare "D:\proj\project1.project"
```

"D: \proj\project2.project" --ignorewhitespace="true"

See also

-     * Chapter 6.4.1.21.3.5.21 "Command 'Compare"' on page 2607

Option -- If you add this option after the option --compare <project1> <project2>, then comments ignorecomment are ignored in the project comparison.
$s$ (ignore com-
ments in project
comparison)
Syntax: --compare="<path of project file>" "<path of reference project file>"
--ignorecomments="true"|"false"

## Example

```
Automation Builder.exe --compare "D:\proj\project1.project"
"D:\proj\project2.project" --ignorecomments="true"
```

See also

- « Chapter 6.4.1.21.3.5.21 "Command 'Compare"' on page 2607

```
Option -- If you add this option after the option--compare <project1> <project2>, then object
ignorepropert properties (permissions, compile settings, directories, bitmaps, etc.) are ignored in the project
ies (ignore
object proper-
ties in project
comparison)
Syntax: --compare="<path of project file>" "<path of reference project file>"
--ignoreproperties="true"|"false"
```


## Example

```
Automation Builder.exe --compare "D:\proj\project1.project"
"D:\proj\project2.project" --ignoreproperties="true"
```

See also

- $\quad$ \& Chapter 6.4.1.21.3.5.21 "Command 'Compare'" on page 2607

Option --
skipunlicense dplugins (do not load components without a license)
Example Automation Builder.exe --skipunlicensedplugins

Option -signaturethum bprint (thumbprint of the certificate which is used for signing compiled libraries)

Syntax: --signaturethumbprint="<thumbprint of digital signature>"

## Example

> Automation Builder.exe --project="D:\projects\test.project" signaturethumbprint="3E96C9B61010CBDC3186021A1CAA64946DDCAAF3"
If you add this option after the option --project="<path of project file>", then the project is opened and the thumbprint of the certificate for signing compiled libraries is passed.

## See also

- ⓨ Chapter 6.4.1.21.3.4.18 "Command 'Security Screen'" on page 2592

Option -- If you add this option after the option --project="<path of project file>", then the enforcesigned "Enforce signing of compiled libraries" option is enabled in the project in the "Security Screen" compiledlibra ries (enforce signing of compiled libraries) on the "User" tab.
NOTICE!
When the "Security Screen" is opened and closed, the current settings are
applied in the user options, even when no active changes have been made.

Syntax: --enforcesignedcompiledlibraries="true"|"false"

## Example

```
Automation Builder.exe --project="D:\projects\test.library" --
enforcesignedcompiledlibraries="true"
```


## See also

- ${ }^{4}$ Chapter 6.4.1.21.3.4.18 "Command 'Security Screen'" on page 2592

| Option -- | If you add this option after the option --project="<path of project file>", then the |
| :--- | :--- |
| timestampings | Internet address of the RFC-3161 time stamp server ("Timestamping server") is set in the <br> erverurl (set <br> the time stamp <br> project in the "Security Screen" on the "User" tab. |
| server address) |  | | NOTICE! |
| :--- |
| When the "Security Screen" is opened and closed, the current settings are <br> applied in the user options, even when no active changes have been made. |

## Syntax: --timestampingserverurl="<URL of RFC-3161 timestamping server>"

## Example

```
Automation Builder.exe --timestampingserverurl="http://
timestamp.comodoca.com/rfc3161"
```


## See also

- $\quad$ Chapter 6.4.1.21.3.4.18 "Command 'Security Screen'" on page 2592


### 6.4.1.17 Using Libraries

$\qquad$
6.4.1.17.2 Library Manager............................................................................ 2034
6.4.1.17.3 Information for Library Developers............................................... 2035
6.4.1.17.4 Adding a Library to the Application............................................... 2036
6.4.1.17.5 Adding a library to the repository.................................................. 2037
6.4.1.17.6 Exporting library files.................................................................... 2037

### 6.4.1.17.1 Library repository

The library repository is the storage location on the development system for libraries and associated metadata. You can link any installed the libraries into your project by means of a library manager. Moreover, the libraries are installed with version management for easy library updates.
You can create and edit more repositories in addition to the preinstalled System repository.
̌) Chapter 6.4.1.21.3.9.5 "Command 'Library Repository'" on page 2657

### 6.4.1.17.2 Library Manager

In order to be able to use POUs, which are provided in a library POU, in the application, the library has to be integrated in the Library Manager in the project. The requirement for this is the installation of the library in the library repository.
The Library Manager displays all integrated libraries according to their library type and the respective properties. In the Library Manager, you can add more libraries from the library repository, remove libraries, and edit library properties.

The Library Manager can be inserted into the "POUs" view or the "Devices" view. As a result, a project can have a Library Manager for each application, as well as a Library Manager in the "POUs" view for use across all applications. The library POUs of the integrated libraries in the "POUs" view can be called regardless of the application. The library POUs of the integrated libraries in the "Devices" view can be called in the respective application code only. Furthermore, placeholder libraries behave differently when downloading depending on their positions.
Libraries that are integrated to a specific version in the project also have a placeholder for that version (placeholder library). You can define special placeholder resolutions. You can also use the placeholder resolution that is defined for a device in the device description or that is stored in the library repository for a library. The Library Manager notifies you about the actual placeholder resolution and shows the version that will be loaded when an application is downloaded (effective version).
When a Library Manager in the "POUs" view is integrated across all applications, you can access its contents globally. If placeholder libraries are integrated, then only the placeholder resolutions in the device description or library repository are checked.

A Library Manager is usually integrated in the "Devices" view. Then only the application code below it calls library POUs from it. Moreover, the special placeholder resolutions are checked first for placeholder libraries. Only after that are the placeholder resolutions checked that are in the device description or that originate from the library repository.
« ${ }^{\mu}$ Chapter 6.4.1.21.2.16 "Object 'Library Manager'"' on page 2469
« Chapter 6.4.1.9.9 "Using Library POUs" on page 1890
6.4.1.17.3 Information for Library Developers

In order to avoid consistency problems and to adequately support the user, be sure to adhere to certain rules for the creation, referencing, encryption, protection, and documentation of libraries.

The following description provides only an overview of the library development possibilities. For a more detailed description of these topics, see the "LibDevSummary" guidelines for library development.

## See also

- ${ }^{4}$ Chapter 6.4.1.17 "Using Libraries" on page 2034


## General

- You can define categories for libraries. The libraries are then displayed in the library repository below these categories.
- You can define a namespace for a library in order to enable unambiguous access to the integrated objects. The access becomes unambiguous by adding the namespace in front of the POU name:
<namespace>.<variable name>
Example: AC.Module
- You can open the POUs of unencrypted libraries (*. library) by double-clicking the respective entry in the Library Manager.
- You can create the following library types:
- *. library: Implementation library (source code of the library)
_ *.compiled-library, *.compiled-library-v3: Protected library; source code no longer accessible.
- *_Itfs.library: Interface library; contains only objects that are used for the interface definition of a component (for example, constants, structures, or interfaces) and do not generate any code.
- *_Cnt. library: Container library; does not contain any POUs; instead contains ex̄clusively other libraries; therefore used to conveniently integrate an entire set of libraries whose POUs are published on the top level of the container library.
- You can integrate external libraries into the application. External libraries are programmed outside of CODESYS in a different programming language, for example C.

Protection of libraries

- Source code protection:

When a library is prepared in "compiled-library" format, the source code of the library POUs is no longer visible after the library is integrated into a project.

- Signing:

In CODESYS V3 SP15 and higher, a certificate is always used for the signing of library projects (*. compiled-library-v3). The signing can be enforced by means of a setting in the security screen. Then for generating a compiled library, you need a certificate suitable for code signing in your user profile.
For library projects that have to be compatible with CODESYS < V3 SP15 (*. compiledlibrary), only the less safe signing is possible with a private key and a corresponding token. These deprecated methods should only be used for reasons of compatibility. Settings are configured in the "Project Information" on the "Signing" tab.
Note: For signing libraries, you should use compiler version 3.5.15.0 or higher because a better storage format is used.

- Licensing:

You can protect libraries by means of a license (dongle or soft container). License-protected libraries can be installed in the library repository. However, for use in the project, the valid license has to exist on the computer. Licenses are managed in the License Manager.
Library versions

- You can have several versions of a library installed on the system at the same time.
- You can have several versions of a library integrated into your project at the same time. However, we do not recommend doing this. In this case, each of the libraries must be assigned a unique namespace and access to the symbols must be qualified. Examples: V1. SendBlob, V2.SendBlob


## Referenced libraries

- You can integrate a library into other libraries (referenced libraries). The nesting can be of any depth.
- You can define whether referenced libraries should be visible in the Library Manager.
- You can integrate referenced libraries via library placeholders. This way you avoid the problems that could occur due to version dependencies or the necessity to use vendor-specific libraries.

See also

- \#y Chapter 6.4.1.3.4.2 "Retrieving and Editing Project Information" on page 1813


### 6.4.1.17.4 Adding a Library to the Application

The following instructions describe how to integrate for example the library Util into your application in order to use its library POUs.

1. Select the Library Manager and click "Project $\rightarrow$ Edit Object" to open it in the editor. $\Rightarrow$ The Library Manager opens in the editor.
2. Click "Library $\rightarrow$ Add Library". $\Rightarrow$ The "Add Library" dialog opens.
3. Type the string "util" into the input field above to search the library.
$\Rightarrow$ The library Util is displayed in the library view.
4. Select the library Util and click "OK" to close the dialog.
$\Rightarrow$ The library Util is added to the Library Manager.

See also

- ${ }^{4}$ Chapter 6.4.1.9.9 "Using Library POUs" on page 1890
- ⓨ Chapter 6.4.1.21.3.15.1 "Command 'Add Library'" on page 2712
- 出 Chapter 6.4.1.17.5 "Adding a library to the repository" on page 2037


### 6.4.1.17.5 Adding a library to the repository

The following instructions describe how to install a library in the library repository.

1. Select the command "Tools $\rightarrow$ Library Repository".
$\Rightarrow$ The dialog box "Library Repository" opens.
2. Click on the "Install" button.
3. Select the library that you wish to install. You can set a file filter here.

Click on "Open".
$\Rightarrow$ The library is added to the repository. The library can now be added in the Library Manager.

See also

- ${ }^{\circ}$ Chapter 6.4.1.17.4 "Adding a Library to the Application" on page 2036


### 6.4.1.17.6 Exporting library files

You can export a library from the library manager of a project or from the library repository and then save it as a file to the hard disk.

## Export from the

 library manager1. Open a library manager of an application in a project.
2. Select a library in the library manager.
3. Click the export command in the context menu.
$\Rightarrow$ The "Export Library" dialog box opens.
4. If the selected library is linked in the project not only as a compiled library, but also in source format, then both file types are in the drop-down list for "File type". Otherwise, the filter automatically shows the available type: "*. library or *. compiled-library.
5. Select the file type and storage location and click "Save".

Export from the library manager

1. Open the CODESYS library repository ("Tools" menu).
2. Select a library version in the window of the installed libraries.
3. Click the "Export" button.
$\Rightarrow$ The "Export Library" dialog box opens.
4. As step 4 and 5 for "Export from the library manager".

See also

- $\geqslant$ Chapter 6.4.1.21.2.16 "Object 'Library Manager"' on page 2469
- \& Chapter 6.4.1.21.3.9.5 "Command 'Library Repository"' on page 2657


### 6.4.1.18 Managing devices

### 6.4.1.18.1 General

CODESYS manages the installed devices in the device repository. A device repository is a defined location in the file system. In the default CODESYS installation, it is defined with an absolute path as the system repository. You install or uninstall devices in the "Device Repository" dialog. The system installs a device by reading the device description file. The properties of a device are defined in these files regarding configurability, programmability, and possible connections to other devices.
You can use the devices provided in the device repository by adding them to the device tree of your project.
See also

- \& Chapter 6.4.1.21.3.9.8 "Command 'Device Repository'" on page 2663
- © Chapter 6.4.1.18.2 "Installing devices" on page 2038


### 6.4.1.18.2 Installing devices

Install a device in the device repository in order to include it in your project.

1. Click "Tools $\rightarrow$ Device Repository". $\Rightarrow$ The "Device Repository" dialog box opens.
2. Select the install location. "System Repository" is set by default.
3. Click "Install". $\Rightarrow$ The "Install Device Description" dialog box opens.
4. Select the file path of the device description.
5. Select the file type filter of the required device description.
$\Rightarrow$ All device descriptions of the selected file type are listed.
6. Select the required device description and click "Open".
$\Rightarrow$ CODESYS adds the device description to the matching category of your device repository.
If errors occur during installation (for example, missing files that are referenced by the device description), then CODESYS displays them in the lower part of the device repository dialog box.

See also

- «y Chapter 6.4.1.21.3.9.8 "Command 'Device Repository"' on page 2663


### 6.4.1.19 Security

6.4.1.19.1 General ..... 2039
6.4.1.19.2 General Information ..... 2039
6.4.1.19.3 Security for the development system. ..... 2041
6.4.1.19.4 Security for the Runtime/PLC. ..... 2041
6.4.1.19.5 Security for CODESYS WebVisu ..... 2042
6.4.1.19.6 FAQ. ..... 2043

### 6.4.1.19.1 General

Due to the increased networking of controllers and plants, potential threats are also quickly rising. Therefore, you should carefully consider all possible security measures.
Security measures are absolutely necessary to protect data and communication channels from unauthorized access.

On the following help pages, you can learn more about the safety functions of CODESYS and the controller.

### 6.4.1.19.2 General Information

The following provides some general information about safety functions (security measures). This information applies regardless of the usage in CODESYS or one with a connection controller.

## Access protection with user management

As a means of protecting against unauthorized access to data, it is necessary to configure user accounts with specific access rights. Only a user with the credentials has access to the data or functions.

Creating passwords according to the general recommendations for achieving a high password strength is a tremendous contribution to security.

The following types of user management are roughly distinguished as follows:

- Simple user management:

To access data, only a password or the valid combination of user name and password has to be entered. This means that access can be only granted or denied. Graduated permissions cannot be configured.

- Group-based user management:

The access rights are assigned to user groups. Users who belong to a group can access the data or functions after entering the credentials with precisely these assigned and different permissions.

## Encryption, sig- Encryption: <br> nature

Encryption of data means the following: Data is converted into an unreadable form and can only be made readable again with a matching key. In the simplest case, the key is a password or a key pair.

There are two types of encryption methods:

- Symmetric method: (the only type of encryption until the mid-1970s)

Characteristic: Use of a secret key
Advantages: Fast, simple encoding
Disadvantages: The key has to be shared secretly.

- Asymmetric method:

Characteristic: Use of a key pair (private/secret key and public key)
Advantages: The public key can be made accessible to anyone, and authentication possible with it.
Disadvantages: Slow (approx. 1,000 to 10,000 times slower than symmetric methods); complex encoding; long key lengths

Key exchange is usually performed by asymmetric methods; encryption and decryption by symmetric methods.
Signature:
In order for the irrefutable ownership and integrity of a message to be verifiable, it should be provided with a signature. These are usually the steps involved:

- Sender: Determines a unique hash value over the data (H)
- Sender: Encrypts the hash value with private key (He)
- Recipient: Also calculates the hash value and decrypts the He with the public key and compares the two values. This allows the sender to be identified uniquely and verifies that the sender owns the private key.
In the case of asymmetric encryption, a public key contained in a certificate is first exchanged between the sender and the recipient. In addition, each participant needs a private key with which they can decrypt the data if they have the certificate. So if you want to access a certificate, you need a certificate AND a private key.
Hash methods are necessary for this:
- Hash method:

Characteristic: Unique thumbprint of the data (for example, checksum of the data) As low a collision as possible (it is very difficult to find / construct two different data for a single hash value)

## Certificates <br> In order to assign the public key to an identity, it is usually embedded in a certificate.

In certificate-based systems, each user receives a digital certificate. The certificate is used for digital identification. It contains information about the identity and the public key of the user. Each certificate is authenticated by an issuing authority, which in turn may be authenticated by higher authorities. The trust system of this PKI (Public Key Infrastructure) is strictly hierarchical. The common trust anchor is a root certificate.

Contents of a certificate:

- Version
- Serial number
- Algorithm ID
- Issuer (authority or company)
- Validity from (not before) to (not after)
- Certificate owner (subject)
- Certificate owner key information (subject public key)
- Public key algorithm
- Public key of the certificate owner
- Unique ID of the issuer (optional)
- Unique ID of the owner (optional) The owner possess a private key matching the public key.
- Extensions
- Purpose (extended key usage)
- ...

The certificate consists of 2 parts/files:

- Public X. 509 certificate (can be issued to anyone)
- Private key that matches the certificate or its public key only (has to be kept secret).

To manage the certificates in your local "Windows Certificate Store", see the following help page:

- ${ }^{4}$ Chapter 6.4.1.6 "Protecting and Saving Projects" on page 1819


### 6.4.1.19.3 Security for the development system

In CODESYS, you can apply access protection to projects, libraries, as well as individual applications. In addition to a simple write protection for a project, a user management (credentials, access rights) and encryption using certificates should be used.
See current help:

- ${ }^{4}$ Chapter 6.4.1.6 "Protecting and Saving Projects" on page 1819
- $y^{*}$ Chapter 6.4.1.9.18 "Protecting an application" on page 1915


### 6.4.1.19.4 Security for the Runtime/PLC

Communication with the controller connected in the CODESYS project should be protected against unauthorized access in the following ways:

- Enabling user management: simple or group-based
- Certificate-based encryption of communication with the controller

Enabling the First switch the communication to encryption so that you do not reveal any credentials to other security features participants in the network when transferring the user management.

Enforcing encrypted communication

- On the controller:
- Runtime version >=3.5 SP14: Encryption can be enabled for "Communication Policy" and enforced for all clients.
- In CODESYS:
- Encrypted communication can be selected as an option in the device editor on the "Communication Settings" tab (command or "Change Communication Policy" dialog) or in the "Security Screen" view. See the current help regarding this: \& Chapter 6.4.1.11.3 "Encrypting Communication, Changing Security Settings" on page 1967
If the CODESYS Security Agent is installed, then see the help for CODESYS Security Agent.
Enforcing a user management
- On the controller:
- Runtime version >= 3.5 SP17: User management is enforced by default for "Communication Policy".
Note: For enabling the user management, at least a CODESYS development system V3.5 SP16 is necessary. This means that, in the case of enforced user management which has not been enabled yet, you cannot connect to an older development system.
- In CODESYS:
- See the current help regarding this:
y Chapter 6.4.1.11.3 "Encrypting Communication, Changing Security Settings" on page 1967
« Chapter 6.4.1.11.4 "Handling of Device User Management" on page 1971


### 6.4.1.19.5 Security for CODESYS WebVisu

Protect the connection between the web server of the controller and the visualization client with the following measures against unwanted access:

- Configure an HTTPS connection (encryption with SSL/TSL) between the visualization client and the web server.
- Restrict access to the visualization and configure a visualization user management.

Configure an encrypted connection.

An HTTPS connection between the web server and the visualization client requires authentication of the web server by means of a certificate. You can create a self-signed certificate in the "Security Screen".

1. Click "View $\boldsymbol{\rightarrow}$ Security Screen".
2. Create a certificate for the web server on your controller.
$\Rightarrow$ The certificate data for the web server is displayed.
3. Stop your controller.
4. Restart the controller.
$\Rightarrow$ The new certificate is active.
5. Download your application to the controller.
6. Open your browser and specify the URL address of your web server.

The URL of a secure connection corresponds to the following format:
https:// <IP address/URL> :443/ <name of HTM file> .htm.
The HTML file name has to match the configured name as it is set in the "Visualization Manager" object below the WebVisu variant. You will find the IP address of the controller in the device editor when the a connection is active.
$\Rightarrow$ Example: https://localhost:443/webvisu.htm
The browser establishes a connection. If the certificate is not rated as trusted, then a security notice appears.
7. Confirm that you know the risk and want to proceed.
$\Rightarrow$ You have created self-signed certificate and confirmed it as trusted.
Now start the web application with the visualization. The lock symbol in the browser indicates secure communication.

See the chapter "Run as CODESYS WebVisu", which describes in detail how you use certificates in the security screen.

See also

- Chapter "Run as CODESYS WebVisu"
- Chapter "User management of the visualization"
- © Chapter 6.4.1.11.3 "Encrypting Communication, Changing Security Settings" on page 1967
- $\Leftrightarrow$ Chapter 6.4.1.11.4 "Handling of Device User Management" on page 1971

FAQ
6.4.1.19.6.1 Certificate expired ..... 2043
6.4.1.19.6.2 New certificate (while the current one is still valid) ..... 2043
6.4.1.19.6.3 Client does not support security feature, ..... 2043
6.4.1.19.6.4 CA-signed certificates preferred (PLC shell). ..... 2044
6.4.1.19.6.5 Problems at login ..... 2045
6.4.1.19.6.6 Disabling User Management ..... 2045
6.4.1.19.6.7 Permitting encrypted communication again ..... 2046

## Certificate expired

If the certificate from the controller for encrypted communication has expired ( valid from "not before" until "not after"), you get a prompt with a corresponding message in CODESYS when you attempt to access the controller. For example, to renew the expired certificate, you can accept the expired certificate and connect to the controller.


You will see this message again every time you try to login until a valid certificate is installed on the controller.

If you have created or imported a new certificate on the controller, then this new certificate will be available for you to accept the next time you login.
See also

- y Chapter 6.4.1.11.3 "Encrypting Communication, Changing Security Settings" on page 1967

Other clients that communicate encrypted with the controller (for example, PLCHandler) will typically not accept an expired certificate. This means that no connection can be established here.

## New certificate (while the current one is still valid)

A new certificate can be issued before the existing certificate expires. This makes it possible for the encrypted communication to continue seamlessly. As soon as a new certificate is available on the controller parallel to the one currently used, the new certificate will be offered by the controller at the next login attempt. All you have to do is accept it.
See also

- $\Leftrightarrow$ "Installing a controller certificate for encrypted communication via the PLC shell of the device editor" on page 1969


## Client does not support security feature

User manage- The following CODESYS clients do not support user management yet: ment

- WebServer < V3.5.14.0

In order for these clients to be able to establish a connection to the controller, the user management must not be enabled.

Encrypted com- The following CODESYS clients do not support encrypted communication yet: munication

- Data servers with compiler version $=<\mathrm{V} 3.5 .9 .0$
- WebVisu < V3.5.14.0 or in the case of enabled file transfer
- WebServer < V3.5.14.0
- Remote TargetVisu
- Data source ApplicationV3
- OPC Server V3
- PLCHandler < V3.5.14.0

In order for these clients to be able to establish a connection to the controller, the encrypted communication can be set as optional. Therefore clients can establish either an encrypted or an unencrypted connection.

Do not use the same user or password for encrypted and unencrypted communication.

See also

- © Chapter 6.4.1.11.3 "Encrypting Communication, Changing Security Settings" on page 1967


## CA-signed certificates preferred (PLC shell)

Using CA-signed certificates is not conveniently supported yet in CODESYS. However, you can still request and use these types of certificates. In the device editor, on the "PLC Shell" tab, you export the required CSR files to the file system and import from there the CER files sent from the certification authority.

## Requesting and providing a CAsigned certificate

$\checkmark$ You are connected to the controller.

1. First you generate certificate signing requests (CSR) of all server certificates.

For this purpose, click the "PLC Shell" tab of the controller and type the command certcreatecsr in the input line.
2. Click the "Log" tab and then the refresh button (*).
$\Rightarrow$ In the log entries, you can see that the CSR files were generated.
3. Click the "Files" tab and open the file path cert/export in the right side of the "Runtime" dialog.
$\Rightarrow$ The export folder contains the generated CSR files, for example
0_CmpsecureChannl.csr, 1_CmpApp.csr, 2_CmpWebServer.csr.
4. Select a file path where you wish to insert the CSR files in the left side of the "Host" dialog, mark the CSR files in the right side of the dialog, and click $\boxed{<}$.
$\Rightarrow$ The CSR files are copied to the required folder.
5. These requests can be signed for certification signing by a certificate authority (CA), and then you receive a signed certificate from the certification authority.
6. In the steps that follow, you import these signed server certificates to your controller.

## NOTICE!

Caution: Self-signed certificates of the server must be deleted before importing the CA-signed certificates.
7. Select the "Path" cert/import in the right side of the "Runtime" dialog.
8. In the left side of the "Host" dialog, select the path in the file system where you saved the signed certificates and selected the certificates.
9. Click $\qquad$
$\Rightarrow$ The certificates are copied to the cert/import folder.
10. Click the "PLC Shell" tab.
11. Type the command cert-import own <file name of the certificate.cer> in the input line of the tab and press the [Enter] key.
$\Rightarrow$ The signed certificates are available to the runtime system servers.

See also

- $\wedge^{\wedge}$ Chapter 6.4.1.21.2.8.11 "Tab 'PLC Shell"' on page 2441
- $\boldsymbol{y y}^{\text {• }}$ Chapter 6.4.1.21.2.8.9 "Tab 'Log'" on page 2437
- ${ }^{*}$ Chapter 6.4.1.21.2.8.8 "Tab 'Files'" on page 2437


## Problems at login

If you have entered an incorrect password when logging in to the user management of the controller, then the login dialog reappears immediately afterwards. After three incorrect attempts, the controller is locked for a defined period of time. However, stricter policies on the controller can lead to the user being locked out and only authorized again by an administrator.

See also

- ${ }^{*}$ Chapter 6.4.1.21.3.7.2 "Command 'Login'" on page 2624


## Disabling User Management

## NOTICE!

After disabling the user management, your controller is accessible again for everyone in the network of the controller. Therefore, you should only do this in justified exceptional cases or if the clients used do not support any user management.

For enabling the user management, at least a CODESYS development system V3.5 SP16 is necessary. This means that, in the case of enforced user management which has not been enabled yet, you cannot connect to an older development system.

1. If the security policy for device user management is set to "Enforced", first set it back to "Optional".
2. Execute the "Reset Origin Device" command. This deletes the user management and you can then reconnect to the controller without having to enter user credentials. Note: In CODESYS Version 3.5 SP16 Patch 20 and higher, you can exclude the boot application from the delete operation when you execute "Reset Origin Device".

See also

- © "Changing the communication policy (encryption, user management)" on page 1970 * Chapter 6.4.1.21.3.7.13 "Command 'Reset Origin Device"' on page 2636
- ${ }^{4}$ Chapter 6.4.1.11.4 "Handling of Device User Management" on page 1971
- ${ }^{4}$ Chapter 6.4.1.6 "Protecting and Saving Projects" on page 1819


## Permitting encrypted communication again

Remember that not every controller supports the deactivation of encrypted communication.

## NOTICE!

We strongly advise against disabling encrypted communication. Especially in connection with an enabled user management, encrypted communication should be enabled so that credentials do not fall into the wrong hands.

To disable encrypted communication with the controller again, proceed as follows:

1. If the communication policy for encrypted communication is set to "Enforced", first set it back to "Optional".
2. In the device editor, on the "Communication Settings" tab in the "Device" menu, disable "Encrypted communication". If you have installed the CODESYS Security Agent, then you can also change the setting in the "Security Screen".
$\Rightarrow$ CODESYS establishes unencrypted communication again with the controller. Other clients can also communicate again without encryption.

See also

- ${ }^{\text {® }}$ Chapter 6.4.1.11.3 "Encrypting Communication, Changing Security Settings" on page 1967
- ② Chapter 6.4.1.21.3.4.18 "Command 'Security Screen'" on page 2592
6.4.1.20.1 Programming Languages and Editors. ..... 2047
6.4.1.20.2 Variables ..... 2113
6.4.1.20.3 Operators ..... 2129
6.4.1.20.4 Operands ..... 2219
6.4.1.20.5 Data Types ..... 2234
6.4.1.20.6 Pragmas. ..... 2269
6.4.1.20.7 Identifiers. ..... 2327
6.4.1.20.8 Shadowing Rules ..... 2333
6.4.1.20.9 Keywords. ..... 2335
6.4.1.20.10 Methods 'FB_Init', 'FB_Reinit', and 'FB_Exit'. ..... 2336
6.4.1.20.11 Error Messages and Warnings. ..... 2341
6.4.1.20.1.1 Introduction. ..... 2047
6.4.1.20.1.2 Declaration Editor. ..... 2047
6.4.1.20.1.3 Common functions in graphical editors. ..... 2048
6.4.1.20.1.4 Structured Text and Extended Structured Text (ExST). ..... 2049
6.4.1.20.1.5 Sequential Function Chart (SFC). ..... 2062
6.4.1.20.1.6 Function Block Diagram / Ladder Diagram / Instruction List (FBD/LD/IL) ..... 2082
6.4.1.20.1.7 Continuous Function Chart (CFC) and Page-Oriented CFC. ..... 2097
6.4.1.20.1 Programming Languages and Editors


## Introduction

You program a POU in each case in the editor for the implementation language that you selected when creating the POU. CODESYS offers a text editor for ST and graphic editors for SFC, FBD/LD/IL and CFC.

The editor opens with a double-click on the POU in the device tree or in the "POUs" view.
Each of the programming language editors consists of two sub-windows:

- In the upper part you make declarations in the "declaration editor", in text or tabular form depending on the setting.
- In the lower part you insert the implementation code in the respective language.

You can configure the display and the behavior of each editor project-wide on the associated tab of the CODESYS options.

## Declaration Editor

In the declaration editor, you declare variables in variable lists and POUs.
If the declaration editor is used with an implementation language editor, then is opens in a view above the implementation language editor.
The declaration editor offers two possible views: textual 穽 and tabular $⿴$. In the "Tools
$\rightarrow$ Options $\rightarrow$ Declaration Editor" dialog, you define whether only the textual view or only the tabular view is available, or whether you can switch between both views by means of the buttons on the right side of the editor view.
A rectangle selection is possible in the textual view of the declaration editor. The key combinations for the rectangle selection are located on the help page for the ST editor.
See also

- ${ }^{4}$ Chapter 6.4.1.9.4.2 "Using the declaration editor" on page 1851
- ${ }^{\bullet}>$ Chapter 6.4.1.9.4 "Declaration of Variables " on page 1847
- ${ }^{4}$ Chapter 6.4.1.20.1 "Programming Languages and Editors" on page 2047
- ${ }^{4}$ Chapter 6.4.1.21.4.14.5 "Dialog 'Options' - 'Declaration Editor'" on page 2785
- ${ }^{*}$ Chapter 6.4.1.20.1.4.1 "ST Editor" on page 2049


## Declaration editor in online mode

In online mode, you see the tabular view of the editor. The header always contains the current object path:<device name>. <application name>. <object name>. In contrast to offline mode, the table also contains the "Value" and "Prepared Value" columns.
The "Value" column shows the actual value on the PLC, offering monitoring functionality. If the expression is an array with more than 1,000 elements, then you can define the range of the array indices to monitor. To do this, double-click in the "Data Type" column to open the "Monitoring Area" dialog. In this dialog, the declared array range is specified as the "Valid area" for monitoring. A maximum of 20,000 elements can be monitored per array. You define the range of the array indices to be monitored by specifying the "Start" and "End" indices. In order to move this area more easily while maintaining the same size, the available scrollbars can be used coupled. Tot toggle between coupled $\sqrt{3}$ and not coupled $[8$, click the symbol on the right of the bar. In non-coupled state, you can increase or decrease the size of the area to be monitored as desired.

The "Prepared Value" column contains the value that you prepared for forcing or writing.
If you double-click a "Prepared value" field, then you can specify a value explicitly for writing or forcing. In the case of enumerations, a combo box opens from which you can select a value. In the case of a Boolean variable you can toggle the prepared value with the help of the [Enter] key or the [Space] bar. If an expression (variable) is of a structured data type, for example the instance of a function block or an array variable, then a plus or a minus sign is placed in front.

You can customize the format of the representation of floating-point values in the options for monitoring.
See also

- $\quad 4$ Chapter 6.4.1.9.4.2 "Using the declaration editor" on page 1851
- ⓨ Chapter 6.4.1.21.4.8 "Dialog Box 'Prepare Value'" on page 2749
- 乡y "Forcing in the declaration part" on page 1988
- $\boldsymbol{y y}^{\boldsymbol{y}}$ Chapter 6.4.1.21.4.14.19 "Dialog 'Options' - 'Monitoring"' on page 2792


## Common functions in graphical editors

The implementation part of the graphical editors for FBD, LD, CFC, and SFC contains a toolbar in the lower right corner.

| $\downarrow$ | Return to normal editing mode: The mouse pointer changes back to the shape of the default arrow. You can select and edit elements in the editor view. |
| :---: | :---: |
| $\pm$ | Panning tool: The mouse pointer changes to the shape of two crossed arrows. You can click and drag anywhere in the editor view to move the visible area of the FBD/LD/IL editor or also pivot a CFC chart. |
| Q | Magnification tool: A magnified window opens in the lower right corner of the editor view and the mouse pointer changes to the shape of a cross. As you move the mouse pointer over your diagram, the magnification tool shows the area of the diagram under the cross at $100 \%$ magnification. Note: If you click in the view, then the magnification tool closes and the part of the diagram that the tool contained is displayed at $100 \%$ magnification. If you want to retain the set zoom factor, then you should use the default arrow ( $\dagger$ ) for returning to the default editing mode. |
| 図 | Zooming tool: This opens a drop-down list with a selection of zoom factors. Clicking more selections (. . . ) will open the "Zoom" dialog for typing other values. The current zoom factor is always shown to the left of the symbol. |

Zooming with the scroll wheel: By holding down the [Ctrl] key and moving the scroll wheel, you can change the zoom factor in steps of $10 \%$.
Every graphical editor has its own "ToolBox" view that is located on the right of the editor view by default. The toolbox contains elements that you can drag to insertion points in the editor view. CODESYS highlights the insertion points with gray position flags in the shape of diamonds, triangles, or arrows. These flags are green when you move the mouse pointer over them. When you release the mouse button, CODESYS inserts the element at the selected position.
It is also possible to use the mouse for moving elements in the editor.
You can drag function block declarations in the FBD, LD, and CFC graphical editors to the editor view. To do this, select the full declaration (variable name and data type) and drag it to a suitable position in the editor view. In the ladder diagram, you can also drag Boolean declarations to the editor and insert them as contacts.
See also

- ${ }^{\mu}$ Chapter 6.4.1.20.1.5.1 "SFC editor" on page 2062
- ${ }^{4}$, Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082
- ${ }^{*}$ Chapter 6.4.1.20.1.7.2 "CFC Editor" on page 2098


## Structured Text and Extended Structured Text (ExST)

6.4.1.20.1.4.1 ST Editor. ..... 2049
6.4.1.20.1.4.2 ST editor in online mode. ..... 2050
6.4.1.20.1.4.3 ST expressions. ..... 2050
6.4.1.20.1.4.4 Assignments ..... 2051
6.4.1.20.1.4.5 Statements. ..... 2055

## ST Editor

The ST editor is a textual editor used for the implementation of code in Structured Text (ST) and Extended Structured Text (ExST).
The line numbering is displayed on the left side of the editor. When inputing programming elements, the "List components" functionality (activated in the CODESYS options, "SmartCoding" category) and the Input Assistant [F2] are also useful. When the cursor is placed over a variable, CODESYS shows a tooltip with information for declaring variables.
The box selection can be made with the following key combinations:

- [Shift]+[Alt]+[Arrow Right]: The selected area is extended one position to the right.
- [Shift]+[Alt]+[Arrow Left]: The selected area is extended one position to the left.
- [Shift]+[Alt]+[Arrow Up]: The selected area is extended one position up.
- [Shift]+[Alt]+[Arrow Down]: The selected area is extended one position down.

The behavior (for example parentheses, mouse actions, tabs) and appearance of the editor are configured in the CODESYS options in the "Text Editor" category.
For an incremental search for strings in the editor, open an input field at the bottom edge of the editor by means of the key combination [Ctrl]+[Shift]+[i]. As soon as you start typing in characters, the corresponding search locations are highlighted in color in the editor. The number of found matches is shown to the right of the input field. You can set the cursor at the search location by using the arrow keys or the key combinations [AIt]+[Page Up] or [AIt]+[Page Down].
When you place the cursor on a symbol name, all occurrence locations of the symbol within the editor are highlighted in color. The search locations correspond to the hits in the cross-reference list. For very large projects, this can cause input delays. In this case, you can disable the function in the options of the text editor.
CODESYS identifies syntax errors already when inputing in the editor and shows the corresponding messages in the message view ("Precompile" category). If the corresponding option is selected in the CODESYS options ("SmartCoding" category), then the error locations in the text are also underlined with a wavy red line.

The "Format Document" command provides an automatic formatting of syntactically correct ST code.
See also

- $\sum^{4}$ Chapter 6.4.1.9.5.4.2 "Programming structured text (ST)" on page 1879
-     * Chapter 6.4.1.20.1.4.3 "ST expressions" on page 2050
- $\Leftrightarrow$ Chapter 6.4.1.20.1.4.5.11 "ST - Comments" on page 2061
- ${ }^{\text {• }}$, Chapter 6.4.1.20.1.4 "Structured Text and Extended Structured Text (ExST)" on page 2049
- ${ }^{\circ}>$ Chapter 6.4.1.21.4.14.26 "Dialog 'Options' - 'Text Editor'" on page 2798
- $\Leftrightarrow$ Chapter 6.4.1.21.3.4.5 "Command 'Messages"' on page 2583
- ${ }^{〔}$, Chapter 6.4.1.21.3.3.45 "Command 'Advanced' - 'Format Document'" on page 2581


## ST editor in online mode

In online mode CODESYS displays the variables and expressions in the ST editor. The writing and forcing of the variables and expressions as well as debugging functions (breakpoints, single step execution) are also possible.
If you use assignments as expressions in ST programming, no further breakpoint positions are created within a line.
See also

- $\Downarrow$ Chapter 6.4.1.13.2.2 "Calling of monitoring in programming objects" on page 1996
- ${ }^{\text {• }}$ Chapter 6.4.1.12.5 "Forcing and Writing of Variables" on page 1987
- ${ }^{\bullet}$, Chapter 6.4.1.13.2.3 "Using watch lists" on page 2002
- $\&$ Chapter 6.4.1.12.3 "Using Breakpoints" on page 1981
- $\quad$ Chapter 6.4.1.12 "Testing and Debugging" on page 1980
- « Chapter 6.4.1.12 "Testing and Debugging" on page 1980
- ${ }^{\bullet}$ - Chapter 6.4.1.12.4 "Stepping Through a Program" on page 1985
- ② Chapter 6.4.1.12.7 "Flow Control" on page 1992
- ${ }^{\bullet}$ Chapter 6.4.1.12.8 "Determining the current processing position with the call stack" on page 1994


## ST expressions

An expression is a construct that returns a value following its evaluation.
Expressions are composed of operators and operands. In Extended Structured Text (ExST) you can also use assignments as expressions. An operand can be a constant, a variable, a function call or a further expression.

## Examples

| 2014 | $(*$ Constant *) |
| :--- | :--- |
| ivar | $(*$ Variable *) |
| fct $(a, b)$ | $(*$ Function call *) |
| $\left(x^{\star} y\right) / z$ | $(*$ Expression *) |
| real_var2 $:=$ int.var; | $(*$ in ExST: Assignment *) *) |

## See also

- ⓨ "ExST - Extended structured text" on page 1879

Evaluation of expressions

The evaluation of an expression takes place by processing the operators according to certain rules of binding. CODESYS processes the operator with the strongest binding first. Operators with the same binding strength are processed from left to right.

| Operation | Symbol | Binding strength |
| :--- | :--- | :--- |
| Parenthesize | (Expression) | Function name (parameter list) <br> all operators with syntax: <operator> <br> () |
| Function Call | EXPT |  |
| Exponentiate | - |  |
| Negate <br> Complementation | NOT |  |
| Multiplication <br> Division <br> Modulo | 夫 |  |
| Addition | MOD |  |
| Subtraction | + |  |
| Comparison | - |  |
| Equality | <,>,<=,>= |  |
| Inequality | <> |  |
| Bool AND | AND <br> AND_THEN | Weakest binding |
| Bool XOR | OR |  |
| Bool OR | OR_ELSE |  |

See also

- Hapter 6.4.1.20.3 "Operators" on page 2129


## Assignments

6.4.1.20.1.4.4.1 ST assignment operator. ..... 2051
6.4.1.20.1.4.4.2 ST assignment operator for outputs ..... 2052
6.4.1.20.1.4.4.3 ExST assignment 'S=' ..... 2052
6.4.1.20.1.4.4.4 ExST assignment 'R=' ..... 2053
6.4.1.20.1.4.4.5 ExST - Assignment as expression. ..... 2054
6.4.1.20.1.4.4.6 Assignment Operator 'REF='. ..... 2054

## ST assignment operator

## Syntax:

```
<operand> := <expression>
```

This assignment operator executes the same function as the MOVE operator.
See also

- $\quad$ Chapter 6.4.1.20.3.7 "Operator 'MOVE"' on page 2137


## ST assignment operator for outputs

The assignment operator => assigns the output of a function, a function block, or a method to a variable. The position on the right side of the operator can also be blank.

## Syntax

```
<output> => <variable>
```


## Example

```
FBcomp_Output1 => bVar1;
FBcomp_Output2 => ;
```

FBcom_Output1 and FB_Output2 are outputs of a function block. The value of FBcom_Output1 is assigned to the variable bVar1.

## ExST assignment 'S='

When the operand of the Set assignment switches to TRUE, then TRUE is assigned to the variable to the left of the operator. The variable is set.
<variable name> $S=$ <operand name> ;
The variables and the operand have the data type BOOL.

## Example

```
PROGRAM PLC_PRG
VAR
    xOperand: BOOL := FALSE;
    xSetVariable: BOOL := FALSE;
END_VAR
xSetVariable S= xOperand;
```

When the operand xOperand switches from FALSE to TRUE, then TRUE is also assigned to the variable $x$ SetVariable. But then the variable keeps this state, even if the operand continues to change its state.

## Multiple assignments

## NOTICE!

In the case of multiple assignments within a code line, the individual assignments are not processed from right to left, but all assignments refer to the operands at the end of the code line.

```
Example
```

$x$ ResetVariable gets the $R=$ assignment of the return value of funcompute.

```
    FUNCTION funCompute : BOOL
```

    FUNCTION funCompute : BOOL
    VAR_INPUT
    VAR_INPUT
        xIn : BOOL;
        xIn : BOOL;
    END_VAR
    END_VAR
    IF \overline{xIn = TRUE THEN}
    IF \overline{xIn = TRUE THEN}
        funCompute := TRUE;
        funCompute := TRUE;
        RETURN;
        RETURN;
    END_IF
END_IF
PROGRAM PLC_PRG
PROGRAM PLC_PRG
VAR
VAR
xSetVariable: BOOL;
xSetVariable: BOOL;
xResetVariable: BOOL := TRUE;
xResetVariable: BOOL := TRUE;
xVar: BOOL;
xVar: BOOL;
END_VAR
END_VAR
xSe\overline{tVariable S= xResetVariable R= funCompute(xIn := xVar);}
xSe\overline{tVariable S= xResetVariable R= funCompute(xIn := xVar);}
xSetVariable gets the S= assignment of ht return value of funCompute, but not from
xSetVariable gets the S= assignment of ht return value of funCompute, but not from
xResetVariable.

```
xResetVariable.
```

See also

- \#, "ExST - Extended structured text" on page 1879
- ${ }^{\circ}$ Chapter 6.4.1.20.1.4.4.4 "ExST assignment ' $R=$ "' on page 2053


## ExST assignment 'R='

When the operand of the Reset assignment switches to TRUE, then FALSE is assigned to the variable to the left of the operator. The variable is reset.

```
<variable name> R= <operand name> ;
```

The variables and the operand have the data type BooL.

## Example

```
VAR
    xOperand: BOOL := FALSE;
            xResetVariable: BOOL := TRUE;
END_VAR
xResetVariable R= xOperand;
```

When the operand xOperand switches from FALSE to TRUE, then FALSE is also assigned to the variable xResetVariable. But then the variable keeps its state, even if the operand continues to change its state.

## Multiple assignments

## NOTICE!

In the case of multiple assignments within a code line, the individual assignments are not processed from right to left, but all assignments refer to the operands at the end of the code line.

```
Example
```

```
FUNCTION funCompute : BOOL
```

FUNCTION funCompute : BOOL
VAR_INPUT
VAR_INPUT
xIn : BOOL;
xIn : BOOL;
END VAR
END VAR
IF xIn = TRUE THEN
IF xIn = TRUE THEN
funCompute := TRUE;
funCompute := TRUE;
RETURN;
RETURN;
END_IF
END_IF
PROGRAM PLC PRG
PROGRAM PLC PRG
VAR
VAR
xSetVariable: BOOL;
xSetVariable: BOOL;
xResetVariable: BOOL := TRUE;
xResetVariable: BOOL := TRUE;
xVar: BOOL;
xVar: BOOL;
END_VAR
END_VAR
xSe\overline{tVariable S= xResetVariable R= funCompute(xIn := xVar);}
xSe\overline{tVariable S= xResetVariable R= funCompute(xIn := xVar);}
xResetVariable gets the R= assignment of the return value of funCompute.
xResetVariable gets the R= assignment of the return value of funCompute.
xSetVariable gets the S= assignment of ht return value of funCompute, but not from
xSetVariable gets the S= assignment of ht return value of funCompute, but not from
xResetVariable.

```
xResetVariable.
```

See also

- \#, "ExST - Extended structured text" on page 1879
- ${ }^{*}$, Chapter 6.4.1.20.1.4.4.3 "ExST assignment 'S="' on page 2052


## ExST - Assignment as expression

In ExST, as an extension to the IEC 61131-3 standard, CODESYS permits the use of assignments as expressions.

## Examples

| int_var1 $:=$ int_var2 $:=$ int_var3 + 9; | $\left(\begin{array}{l}* \\ \text { int_var1 and int_var2 receive the value of } \\ \left.\text { int_var3 }+9{ }^{*}\right)\end{array}\right.$ <br> real_var1 $:=$ real_var2 $:=$ int_var; <br> int_var $:=$ real_var1 $:=$ int_var; <br> $\left({ }^{*}\right.$ real_var1 and real_var2 receive the value of <br> int_var *) <br> IF b $:=(i=1)$ THEN <br> i $:=i+1 ;$ <br> END_IF$\left({ }^{*}\right.$ incorrect assignment, the data types do not corre- <br> spond! *) |
| :--- | :--- |

See also

- 出 "ExST - Extended structured text" on page 1879


## Assignment Operator 'REF='

The operator generates a reference (pointer) to a value.

## Syntax:

<variable name> REF= <variable name> ;

## Example

```
refA : REFERENCE TO DUT;
B : DUT;
C : DUT;
A REF= B; // corresponds to A := ADR(B);
A := C; // corresponds to A^ := C;
```


## See also

-     * Chapter 6.4.1.20.5.14 "Reference" on page 2245
- ${ }^{2} \Rightarrow$ Chapter 6.4.1.21.3.13. ${ }^{\text {"Command }}$ 'REF= (Reference Assignment)"' on page 2687


## Statements

6.4.1.20.1.4.5.1 $S T$ statement ' $I F$ '. ..... 2055
6.4.1.20.1.4.5.2 ST instruction 'FOR'. ..... 2056
6.4.1.20.1.4.5.3 ST instruction 'CASE'. ..... 2057
6.4.1.20.1.4.5.4 ST instruction 'WHILE'. ..... 2058
6.4.1.20.1.4.5.5 ST Statement 'REPEAT' ..... 2058
6.4.1.20.1.4.5.6 ST statement 'RETURN'. ..... 2059
6.4.1.20.1.4.5.7 ST instruction 'JMP'. ..... 2059
6.4.1.20.1.4.5.8 ST instruction 'EXIT' ..... 2060
6.4.1.20.1.4.5.9 EXST Statement 'CONTINUE'. ..... 2060
6.4.1.20.1.4.5.10 ST function block call. ..... 2060
6.4.1.20.1.4.5.11 ST - Comments ..... 2061

## ST statement 'IF'

The IF statement is used for checking a condition and, depending on this condition, for executing the subsequent statements.

A condition is coded as an expression that returns a Boolean value. If the expression returns TRUE, then the condition is fulfilled and the corresponding statements after THEN are executed. If the expression returns FALSE, then the following conditions, which are identified with ELSIF, are evaluated. If an ELSIF condition returns TRUE, then the statements are executed after the corresponding THEN. If all conditions return FALSE, then the statements after ELSE are executed.
Therefore, at most one branch of the IF statement is executed. ELSIF branches and the ELSE branch are optional.

## Syntax

```
IF <condition> THEN
    <statements>
( ELSIF <condition> THEN
        <statements> )*
( ELSE
        <statements> )?
END IF;
// ( ... )* None, once or several times
// ( ... )? Optional
```


## Example

```
PROGRAM PLC_PRG
VAR
    iTemp: INT;
    xHeatingOn: BOOL;
    xOpenWindow: BOOL;
END_VAR
IF iTemp < 17 THEN
    xHeatingOn := TRUE;
ELSIF iTemp > 25 THEN
    xOpenWindow := TRUE;
ELSE xHeatingOn := FALSE;
END_IF;
```

The program is run as follows at runtime:
For the evaluation of the expression iTemp < 17 = TRUE, the subsequent statement is executed and the heating is switched on. For the evaluation of the expression iTemp < 17 $=$ FALSE, the subsequent ELSIF condition iTemp > 25 is evaluated. If this is true, then the statements in ELSIF are executed and the view is opened. If all conditions are FALSE, then the statement in ELSE is executed and the heating is switched off.

See also

- ${ }^{*} \boldsymbol{y}$ Chapter 6.4.1.20.1.4.3 "ST expressions" on page 2050


## ST instruction 'FOR'

The FOR loop is used to execute instructions with a certain number of repetitions.

## Syntax:

```
FOR <counter> := <start value> TO <end value> {BY <increment> } DO
<instructions>
END_FOR;
```

The section inside the curly parentheses $\}$ is optional.
CODESYS executes the <instructions> as long as the <counter> is not greater, or - in case of negative increment - is not smaller than the <end value>. This is checked before the execution of the <instructions>.

Every time the instructions <instructions> have been executed, the counter <counter> is automatically increased by the increment <increment>. The increment <increment> can have any integral value. If you do not specify an increment, the standard increment is 1 .

## Example

```
FOR iCounter := 1 TO 5 BY 1 DO
iVar1 := iVar1*2;
END_FOR;
Erg}\mp@subsup{}{}{-}:= iVar1
```

If you have pre-configured iVar1 with 1 , iVar1 has the value 32 after the FOR loop.

## CAUTION!

The end value <end value> may not attain the same value as the upper limit of the data type of the counter.

If the end value of the counter is equal to the upper limit of the data type of the counter, an endless loop results. For example, an endless loop results in the above example if iCounter is of the data type SINT and the <end value> equals 127, since the data type SINT has the upper limit 127.

As an extension to the IEC 61131-3 standard you can use the CONTINUE instruction within the FOR loop.

See also

- ${ }^{4}$ Chapter 6.4.1.20.5.3 "Integer data types" on page 2235
- シy Chapter 6.4.1.20.1.4.5.9 "EXST Statement 'CONTINUE"' on page 2060


## ST instruction 'CASE'

Use this dialog box for pooling several conditional instructions containing the same condition variable into a construct.

## Syntax:

```
CASE <Var1> OF
<value1>:<instruction1>
<value2>:<instruction2>
<value3, value4, value5>:<instruction3>
<value6 ... value10>:<instruction4>
<value n>:<instruction n>
{ELSE <ELSE-instruction>}
END_CASE;
```

The section within the curly brackets $\}$ is optional.
Processing scheme of a CASE instruction.

- If the value of the variable <Var1> is <value i>, then the instruction <instruction i> is executed.
- If the variable $<\operatorname{Var} 1>$ has non of the given values, then the $<$ ELSE-instruction $>$ is executed.
- If the same instruction is executed for several values of the variable, then you can write the values in sequence, seperated by commas.


## Example

```
CASE iVar OF
1, 5: bVar1 := TRUE;
        bVar3 := FALSE;
2: bVar2 := FALSE;
    bVar3 := TRUE;
10..20: bVar1 := TRUE;
            bVar3= TRUE;
ELSE
    bVar1 := NOT bVar1;
    bVar2 := bVar1 OR bVar2;
END_CASE;
```


## ST instruction 'WHILE'

The WHILE loop is used like the FOR loop in order to execute instructions several times until the abort condition occurs. The abort condition of a WHILE loop is a boolean expression.

## Syntax:

```
WHILE <boolean expression> DO
    <instructions>
    END_WHILE;
```

CODESYS repeatedly executes the <instructions> for as long as the <boolean expression> returns TRUE. If the boolean expression is already FALSE at the first evaluation, then CODESYS never executes the instructions. If the boolean expression never adopts the value FALSE, then the instructions are repeated endlessly, as a result of which a runtime error results.

## Example

```
WHILE iCounter <> 0 DO
Var1 := Var1*2
iCounter := iCounter-1;
END_WHILE;
```


## NOTICE!

You must ensure by programming means that no endless loops are caused.

In a certain sense the WHILE and REPEAT loops are more powerful than the FOR loop, since you don't need to already know the number of executions of the loop before its execution. In some cases it is thus only possible to work with these two kinds of loop. If the number of executions of the loop is clear, however, then a FOR loop is preferable in order to avoid endless loops.

As an extension to the IEC 61131-3 standard you can use the CONTINUE instruction within the WHILE loop.

See also

- ${ }^{4}$ Chapter 6.4.1.20.1.4.5.2 "ST instruction 'FOR'" on page 2056
- ⓨ Chapter 6.4.1.20.1.4.5.9 "EXST Statement 'CONTINUE"' on page 2060


## ST Statement 'REPEAT'

The REPEAT loop is used like the WHILE loop, but with the difference that CODESYS only checks the abort condition after the execution of the loop. The consequence of this behavior is that the REPEAT loop is executed at least once, regardless of the abort condition.

## Syntax:

```
REPEAT
<instructions>
    UNTIL <boolean expression>
    END_REPEAT;
```

CODESYS executes the <instructions> until the <boolean expression> returns TRUE.
If the boolean expression already returns TRUE at the first evaluation, CODESYS executes the instructions precisely once. If the boolean expression never adopts the value TRUE, then the instructions are repeated endlessly, as a result of which a runtime error results.

## Example

```
REPEAT
Var1 := Var1*2;
iCounter := iCounter-1;
UNTIL
iCounter = 0
END_REPEAT;
```

In a certain sense the WHILE and REPEAT loops are more powerful than the FOR loop, since the number of executions of the loop doesn't already need to be known before its execution. In some cases you can only work with these two kinds of loop. If the number of executions of the loop is clear, however, then a FOR loop is preferable in order to avoid endless loops.
As an extension to the IEC 61131-3 standard you can use the CONTINUE instruction within the WHILE loop.
See also

- ※ Chapter 6.4.1.20.1.4.5.4 "ST instruction 'WHILE"' on page 2058
- ${ }^{4}$ Chapter 6.4.1.20.1.4.5.2 "ST instruction 'FOR"' on page 2056



## ST statement 'RETURN'

Use the RETURN statement in order to exit from a function block. You can make this dependent on a condition, for example.

## Example

```
IF xIsDone = TRUE THEN
    RETURN;
END_IF;
iCounter := iCounter + 1;
```

If the value of xIsDone is equal to TRUE, then the function block is exited immediately and the statement iCounter : $=$ iCounter +1 ; is not executed.

## See also

- ${ }^{4}$, Chapter 6.4.1.20.1.4.5.1 "ST statement 'IF"" on page 2055


## ST instruction 'JMP'

The JMP instruction is used to execute an unconditional jump to a program line that is marked by a jump label.

## Syntax:

```
<label>: <instructions>
JMP <label>;
```

The jump label <label> is any unique identifier that you place at the beginning of a program line. On reaching the JMP instruction, a return to the program line with the <label> takes place.

## Example

```
iVar1 := 0;
label1: iVar1 := iVar1+1;
(*instructions*)
IF (iVar1 < 10) THEN
JMP _label1;
END_İF;
```


## ST instruction 'EXIT'

The EXIT instruction is used in a FOR, WHILE or REPEAT loop in order to end the loop regardless of other abort conditions

See also

- ${ }^{4}$, Chapter 6.4.1.20.1.4.5.2 "ST instruction 'FOR'" on page 2056
- ${ }^{4}$ Chapter 6.4.1.20.1.4.5.4 "ST instruction 'WHILE"' on page 2058
- \& Chapter 6.4.1.20.1.4.5.5 "ST Statement 'REPEAT'" on page 2058


## EXST Statement 'CONTINUE'

CONTINUE is an instruction of the Extended Structured Text (ExST).
The instruction is used inside FOR, WHILE and REPEAT loops in order to jump to the beginning of the next execution of the loop.

## Example

```
FOR Counter:=1 TO 5 BY 1 DO
INT1:=INT1/2;
    IF INT1=0 THEN
        CONTINUE; (* to avoid a division by zero *)
        END IF
Var1:=Var1/INT1; (* executed, if INT1 is not 0 *)
END_FOR;
Erg:=Var1;
```


## See also

- ${ }^{\circledR}$ Chapter 6.4.1.20.1.4.5.2 "ST instruction 'FOR"" on page 2056
- ${ }^{*}$, Chapter 6.4.1.20.1.4.5.4 "ST instruction 'WHILE"' on page 2058
- $\Leftrightarrow$ Chapter 6.4.1.20.1.4.5.5 "ST Statement 'REPEAT"" on page 2058


## ST function block call

## Syntax

```
<FB-instance>(<FB input variable>:=<value or address>|, <other FB input
variables>);
```


## Example

```
TMR:TON;
```

```
TMR (IN:=%OX5, PT:=T#300ms);
```

TMR (IN:=%OX5, PT:=T\#300ms);
varA:=TMR.Q;

```
varA:=TMR.Q;
```

The timer function block TON is instanced in TMR: TON and called with assignments for the parameters IN and PT.

The output $Q$ is addressed with TMR. $Q$ and assigned to the variable varA.

## See also

- ${ }^{*}$ Chapter 6.4.1.21.2.21.3 "Object 'Function Block'" on page 2479


## ST - Comments

| Comment | Description | Example |
| :---: | :---: | :---: |
| Single-line | There are two ways of marking: <br> - Starts with / / and ends at the end of the line <br> - Starts with / / / and ends at the end of the line <br> In CODESYS, these comments are handled the same way. <br> However, if library documentation is created using the LibDoc Scripting Collection, the following applies: <br> - When the property LibDocContent = DocsOnly is entered in the project information, only comments marked with / // are processed into library documentation. See the example for this below the table. <br> - When LibDocContent = CommentsAndDocs (default setting) is defined, all comments are processed into library documentation. | /// This is a comment. <br> /// This is a comment. |
| Multiline | Starts with (* and ends with *). | ```(* This is a multiline comment *)``` |
| Nested | Starts with (* and ends with *). Additional comments (*....*) can be contained within this comment. | ( * a:=inst.out; (* comment 1 *) $\mathrm{b}:=\mathrm{b}+1$; (* comment 2 *) *) |

```
Comments for
tooltips and
POU documen-
tation
```

```
A tooltip in the header of a POU is defined by the following
```

A tooltip in the header of a POU is defined by the following
comment:
comment:
// tooltip text - line 1
// tooltip text - line 1
// tooltip text - line 2
// tooltip text - line 2
// tooltip text - line 3
// tooltip text - line 3
Afterwards the documentation is defined as follows:
Afterwards the documentation is defined as follows:
/// reStructuredText

```
        /// reStructuredText
```

Note: It is not recommended to mix the different comment types because this can cause unwanted side-effects when the documentation is generated.

## Sequential Function Chart (SFC)

$$
\text { 6.4.1.20.1.5.1 SFC editor............................................................................... } 2062
$$

6.4.1.20.1.5.2 SFC Editor in Online Mode....................................................... 2063
6.4.1.20.1.5.3 Processing order in SFC........................................................ 2064
6.4.1.20.1.5.4 Qualifiers for Actions in SFC................................................... 2065
6.4.1.20.1.5.5 Implicit variables..................................................................... 2066
6.4.1.20.1.5.6 SFC Flags.............................................................................. 2067
6.4.1.20.1.5.7 Library "Analyzation"............................................................... 2072
6.4.1.20.1.5.8 Elements............................................................................... 2073

## SFC editor

The SFC editor is graphical editor. A new SFC POU includes an Init step and a subsequent transition.

In the SFC editor, you can insert individual elements into the diagram by means of commands in the "SFC" menu, the context menu, or the "ToolBox" view.
When inserting by means of a menu command, the elements that can be inserted at the currently selected position are available.

Before inserting branches parallel to multiple actions and transitions, you must highlight these actions and transitions in a multiple selection.

You can also drag SFC elements from the "ToolBox" view to the diagram. When you drag an element over the editor, CODESYS marks all possible insertion points with gray boxes. If you move the mouse over a gray box, then the color of the box changes to green. When you release the mouse button, the object is inserted at that location.

If you drag a branch into the diagram, then you must set the beginning and the end of the branch using the mouse pointer. You set the beginning of the branch by releasing the mouse button at an insertion point. The color of the box then changes to red. You set the end of the branch by clicking the second insertion point. Then CODESYS inserts a branch around the objects between the beginning and end markers.

For copying step and transition elements that call action objects or transition objects, two different duplication modes can be set. Either the references are copied at the same time, or the referenced objects are embedded and duplicated when copying.

You define the look and feel of the editor in the CODESYS options ("SFC Editor").
See also

- $\quad$ \& Chapter 6.4.1.20.1.3 "Common functions in graphical editors" on page 2048
- $\&$ Chapter 6.4.1.20.1.5 "Sequential Function Chart (SFC)" on page 2062
- ${ }^{4}$ Chapter 6.4.1.9.5.5.2 "Programming in SFC" on page 1880
- ${ }^{\mu}$ Chapter 6.4.1.21.4.14.23 " Dialog 'Options' - 'SFC Editor'" on page 2795


## SFC Editor in Online Mode

In the SFC editor, the variables and expressions in use on the controller can be displayed at runtime. You can also write and force variables and expressions. Debugging functions, such as breakpoints and step-by-step execution, are not available yet.

In the SFC editor options, you can set the online representation of the SFC elements and attributes.

In the case that you have declared SFC flags explicitly, then they are displayed in the declaration part in online mode. They are not displayed in offline mode.

Note the processing order of elements in an SFC diagram.

In online mode, CODESYS displays active steps in blue.


See also

- ${ }^{4}$ Chapter 6.4.1.20.1.5.5 "Implicit variables" on page 2066
- \#y Chapter 6.4.1.9.4.2 "Using the declaration editor" on page 1851
- ${ }^{\circ}$, Chapter 6.4.1.20.1.5.3 "Processing order in SFC" on page 2064
- \& Chapter 6.4.1.21.4.14.23 "Dialog 'Options' - 'SFC Editor'" on page 2795


## Processing order in SFC

## Basic element behavior

- Active step: An active step includes actions currently being executed. In online mode, CODESYS displays active steps in blue.
- Initial step: In the first cycle after calling a POU in SFC, the initial step is activated automatically and the step action is executed.
- CODESYS executes IEC actions at least two times: the first time is when the step is activated, and the second time when the step is deactivated (but not until the next cycle).
- Alternative branches: If the step before the branch is active, then CODESYS passes the first transition of each alternative branch line from left to right. CODESYS activates the subsequent step in the first branch line with a transition yielding TRUE.
- Parallel branches: If the step before the branch (horizontal double line) is active and the transition before the branch yields TRUE, then CODESYS activates the first steps in every branch line. The branch lines are then processed at the same time. The step after the end of the branch is activated when every last step in each branch line is active and the transition after the double line yields TRUE.

Processing order

1. Reset IEC actions

CODESYS resets the internal action control flags of the action qualifiers ( $\mathrm{N}, \mathrm{R}, \mathrm{S}, \mathrm{L}, \mathrm{D}, \mathrm{P}$, SD, DS, SL). These flags control IEC actions. However, flags are not reset when they are called within actions.
2. Execute exit actions

CODESYS verifies whether all steps fulfill the condition for executing the exit action for each step. The order of verification follows the layout in the SFC diagram, from top to bottom and from left to right.
CODESYS executes an exit action when the step is deactivated (after any entry and step actions have been executed in the preceding cycle and the condition for the subsequent step yields TRUE).
3. Execute entry actions

CODESYS verifies whether all steps fulfill the condition for executing the entry action for each step. The order of verification follows the layout in the SFC diagram, from top to bottom and from left to right. If the conditions are fulfilled, then CODESYS executes the entry actions.
CODESYS executes an entry action as soon as the transition of the preceding step has been processed and yields TRUE, thus indicating that the step has been activated.
4. Time check / Execute step actions

CODESYS performs the following check for each step in the order of the SFC layout:

- CODESYS copies the elapsed time of the active step to the respective implicit step variable <step name>.t. (not yet implemented)
- If a timeout occurs, then CODESYS sets the respective error flags. (not yet implemented)
- For non-IEC steps: CODESYS executes the step action.

5. Execute IEC actions

CODESYS executes the IEC actions in alphabetical order, passing through the list of actions two times. In the first pass, CODESYS executes the IEC actions for each step that was deactivated in the preceding cycle. In the second pass, the IEC actions are executed for each active step.
6. Transition check / Activate next steps

The transitions are passed as follows: If a step is active in the current cycle and the subsequent transition yields TRUE and any defined minimum time of the step has elapsed, then the subsequent step is activated.

## NOTICE!

Please note when executing actions:
An action can be executed multiple times within the same cycle if you use it in multiple SFC diagrams. For example, if a sequential function chart includes two IEC actions A and B, both of which are programmed in SFC and call an IEC action $C$, then the IEC action $C$ is called two times.
If you use the same IEC action at the same time in different levels of an SFC diagram, then this can lead to unpredictable results when processing. For this reason, CODESYS issues a corresponding error message. This error message can appear for projects that have been created in an earlier version of the development system.

Please note: It is possible to use implicit variables to monitor the processing status of steps and actions and to control processing.

See also

- Chapter 6.4.1.20.1.5.5 "Implicit variables" on page 2066
- ${ }^{\text {• }}$ Chapter 6.4.1.20.1.5.4 "Qualifiers for Actions in SFC" on page 2065


## Qualifiers for Actions in SFC

You assign qualifiers to IEC steps. Qualifiers describe how a step action is processed.
Qualifiers are processed by the SFCActionControl function block in the library IecSfc.library. The library is automatically integrated into the project by the SFC plug-in.

Table 388: Available qualifiers

| N | Non-stored | The action is active as long as the step. |
| :--- | :--- | :--- |
| R | overriding Reset | The action is deactivated. |
| S | Set (Stored) | CODESYS executes this action as soon as the step is active. The action <br> execution is continued even when the step has been deactivated until it <br> gets a reset. |
| L | time Limited | CODESYS executes this action as soon as the step is active. The action is <br> executed until the step is deactivated or the given time span has elapsed. |
| D | time Delayed | CODESYS begins executing the action only after the given delay time has <br> elapsed following step activation and the step is still active. The action is <br> executed until the step is deactivated. |
| P | Stored and time <br> Delayed | CODESYS executes the action exactly two times: one time when the step <br> is activated and one time when the step is deactivated. |
| CODESYS begins executing the action only after the given delay time has |  |  |
| elapsed following step activation. The action is executed until it gets a |  |  |
| reset. |  |  |

You have to specify the times for the L, D, SD, DS, and SL qualifiers in the format of a TIME constant.

When an IEC action is deactivated, it is executed one more time. This means that CODESYS executes this kind of action at least two times. This also applies to actions with the $P$ qualifier.

See also

- $\quad$ \& Chapter 6.4.1.9.5.5.2 "Programming in SFC" on page 1880


## Implicit variables

Every SFC object supplies implicit variables for you to monitor the status of steps and IEC actions at runtime. These implicit variables are declared automatically by CODESYS for each step and each IEC action.
The implicit variables are structure instances of the type SFCStepType for steps and type SFCActionType for actions. The variables have the same names as their elements, for example "step1" variable name for "step1" step name. The structure members describe the status of a step or action or the currently elapsed time in an active step.

In the element properties, you can define whether CODESYS should export a symbol definition for this flag to the symbol configuration.

## See also

- 出 Chapter 6.4.1.20.1.5.8.6 "SFC element properties" on page 2080

Step and action Syntax for the implicit variable declaration: status

```
<step name>:SFCStepType;
    _<action name>:SFCActionType;
```

Table 389: The following implicit variables are available for step or IEC action status:

| Step |  |
| :---: | :---: |
| <step name>.x | Shows the activation status in the current cycle. <br> When <step name>. $x=$ TRUE, CODESYS processes the step in the current cycle. |
| <step name>._x | Shows the activation status for the next cycle. <br> When <step name>. $x=$ TRUE and <step name>. $x=$ FALSE, CODESYS processes the step in the next cycle. This means that <step name>._x is copied to <step name>. $x$ at the beginning of a cycle. |
| <step name>.t | The flag t yields the current elapsed time since the step was activated. This applies only to steps, regardless of whether a minimum time has been defined or not in the step properties. <br> Also see SFC flag SFCError. |
| <step name>._t | For internal use only |
| IEC action |  |
| _<action name>.x | TRUE when the action is being executed. |
| _<action name>._x | TRUE when the action is active. |

## NOTICE!

You can use the above variables to force a specific status value to a step (activate a step). However, note that this can cause an unstable status in the SFC.

See also

- « ${ }^{*}$ Chapter 6.4.1.20.1.5.6 "SFC Flags" on page 2067

Access to
implicit varia-
bles
bles

## Syntax for access:

Assign the implicit variable directly in the POU: <variable name>:=<step
name>.<implicit variable> or <variable name>:=_<action name>.<implicit variable>
Example status:=step1._x;

From another POU, with the POU name: <variable name>:=<POU name>.<step name>.<implicit variable> or <variable name>:=<POU name>._<action name>.<implicit variable>

```
Example status:=SFC_prog.step1._x;
```

Symbol genera- In the element properties of a step or action, you define whether CODESYS should add a tion symbol definition for the step or action flag. In the "Properties" view, you have to select the necessary access rights in the "Symbol" column.
See also

- ${ }^{\text {® }}$ Chapter 6.4.1.20.1.5.8.6 "SFC element properties" on page 2080

SFC Flags
SFC flags are implicitly generated variables with predefined names. You can use them to influence the processing of an SFC diagram. You can use these flags, for example, to display timeouts or reset step chains. In addition, you can activate jogging mode specifically to activate transitions. You have to declare and activate these variables in order to have access to them.

SFC flags

| Name | Data Type | Description |
| :--- | :--- | :--- |
| SFCInit | Bool | TRUE: CODESYS resets the sequence to the initial step. The other SFC flags are <br> also reset (initialization). While the variable is TRUE, the initial step remains set <br> (active), but its actions are not executed. Only when you set SFCInit again to <br> FALSE is the POU further processed normally. |
| SFCReset | Bool | This function behaves similar to SFCInit. However, CODESYS continues pro- <br> cessing after the initialization of the initial step. For example, in the initial step, you <br> could immediately reset the SFCReset flag to FALSE. |
| SFCError | Bool | TRUE if a timeout occurs in an SFC diagram. If second timeout occurs in the <br> program, it is not registered unless you previously reset the variable SFCError. <br> The declaration of SFCError is a requirement for other flag variables to func- <br> tion for controlling the chronological sequence ("SFCErrorStep", SFCErrorPOU, <br> SFCQuitError). |


| Name | Data Type | Description |
| :---: | :---: | :---: |
| SFCEnableLim it | Bool | Used specifically for activating (TRUE) and deactivating (FALSE) the timeout control in steps using SFCError. If you declare and activate this variable (SFC settings), then you must set it to TRUE for SFCError to work. If you do not, then the timeouts are ignored. The is useful, for example, at start-up or in manual operation. If you do not declare the variable, then SFCError will work automatically. <br> The requirement is the declaration of SFCError. |
| SFCErrorStep | String | Stores the name of the step that caused a timeout, which was registered by SFCError. The name is kept until the registered step error is reset by means of SFCQuitError (FALSE -> TRUE). <br> The requirement is the declaration of SFCError. |
| SFCErrorPOU | String | Stores the name of the block in which a timeout occurred and was registered by SFCError. The name is saved until the timeout is reset by SFCQuitError. <br> The requirement is the declaration of SFCError. |
| SFCQuitError | Bool | As long as this Boolean variable is TRUE, CODESYS pauses the processing of the SFC diagram and any timeout, saved in the variable SFCError, is reset. If you reset the variable to FALSE, then all previous times in the active steps are reset. <br> The requirement is the declaration of SFCError. |
| SFCPause | Bool | As long as this variable is TRUE, CODESYS pauses the processing of the SFC diagram. |
| SFCTrans | Bool | TRUE if a transition is active. |
| SFCCurrentSt ep | String | Shows the name of the active step, regardless of the time monitoring. In parallel branches, the name of the step of the rightmost branch line is always stored. |
| SFCTip, SFCTipMode | Bool | Controls the jogging mode of the SFC block. <br> If you enable this flag with SFCTipMode=TRUE, then you can activate the next step only by setting SFCTip to TRUE. While SFCTipMode is set to FALSE, transitions can also be used to continue activation. |
| SFCErrorAnal yzation, |  | Contains as string all variables that contribute to the total value TRUE of SFCError (timeout in one step). SFCError needs to be activated for this. SFCErrorAnalyzation implicitly uses the function of the POU AnalyzeExpression of the library Analyzation. |
| SFCErrorAnal yzationTable, |  | Contains in a table all variables that contribute to the total value TRUE of SFCError (timeout in one step). SFCError needs to be activated for this. SFCErrorAnalyzationTable implicitly uses the function of the POU AnalyzeExpressionTable of the library Analyzation. |

Implicit generation of SFC flags

CODESYS declares SFC flags automatically when you activate the respective options. You can set this option in the "SFC Settings" tab of the properties dialog for each POU, or in the "SFC" project settings dialog for each SFC POU in the project.


The SFC settings for the SFC flags of individual POUs are effective only if you have not selected the "Use defaults" option. When you select this option, the settings apply that were defined in the project settings.

SFC flags that you declare in the SFC settings dialog are visible only in the online view of the SFC block.

See also

- 会 "Flag" on page 2763

Explicit genera- Manual declaration, which was necessary in CoDeSys V2.3, is now only required to enable tion of SFC write access from another block. In this case, you should note that when you declare the flag in flags a global variable list, you must deactivate its "Declare" setting in the SFC settings dialog. If you do not do this, then a local SFC flag is implicitly declared that CODESYS uses instead of the global variable.

## Application example for SFCError

## Example You have created an SFC block named sfc1, which contains the s1 step. You have defined

 timeouts in the step properties. (See "Online view of SFC block sfc1" below.)If for any reason the s1 step remains active longer than its time properties have permitted (timeout), then CODESYS sets the SFCError flag to permit access by the application.

To permit access, you have to declare and activate the SFC flag in the SFC settings. If you have only declared it, then the SFC flag is only displayed in the online view of sfc1 in the declaration part, but it has no function.


Now the SFC flag can be referenced within the POU, for example in an action (2) or outside of the block (1).


Online view of the SFC block sfc1


SFCError is TRUE as soon as a timeout occurs within sfc2.
Note that you can use the flags SFCErrorAnalyzation and SFCErrorAnalyzationTable to determine the components of the expression that contributes to the value TRUE of the SFCError.

## See also

- ⓨ Chapter 6.4.1.20.1.5.7 "Library "Analyzation"" on page 2072


## Access to the Syntax for access:

You assign the flag directly within the POU: <variable name>:=<SFC flag>

```
Example checkerror:=SFCerror;
```

From another POU with POU name: <variable name>:=<POU name>.<SFC flag>

```
Example: checkerror:=SFC_prog.SFCerror;
```

If you need write access from another block, then you also have to declare the SFC flag explicitly as a VAR_INPUT variable in the SFC block or globally in a GVL.

## Example Local declaration:

```
PROGRAM SFC_prog
VAR INPUT
    S\overline{FCinit:BOOL;}
END_VAR
```

Global declaration in a global variable list:

```
VAR GLOBAL
    S\overline{FCinit:BOOL;}
END VAR
PROGRAM PLC_PRG
VAR
    setinit: BOOL;
END_VAR
SFC_prog.SFCinit:=setinit; // write access to SFCinit in SFC_prog
```


## See also

- « Chapter 6.4.1.20.1.5.7 "Library "Analyzation"" on page 2072


## Library "Analyzation"

This library contains POUs for the analysis of expressions. When a composite expression has the total value of FALSE, those of its components that contribute to this result can be determined. In the SFC editor, the flags SFCErrorAnalyzation and SFCErrorAnalyzationTable use these functions implicitly to examine the transition expressions. Then the flags provide the identifiers of the variables that contributed to a timeout error. They keep this information until they are reset explicitly by means of the SFC flag SFCQuitError.


An analysis POU cannot be called by means of a pointer. This kind of call is ignored. Call the POU as a single instance.

For a description of the library POUs and an example of how the SFC flags display the analysis results in CODESYS, see the documentation for the library (online help or directly in the Library Manager).

See also

- ${ }^{\text {® }}$, Chapter 6.4.1.9.5.5.2 "Programming in SFC" on page 1880
- $\Leftrightarrow$ Chapter 6.4.1.20.1.5.6 "SFC Flags" on page 2067


## Elements

6.4.1.20.1.5.8.1 SFC elements 'Step' and 'Transition'. ..... 2073
6.4.1.20.1.5.8.2 SFC Element 'Action' ..... 2075
6.4.1.20.1.5.8.3 SFC element 'Branch' ..... 2078
6.4.1.20.1.5.8.4 SFC element 'Jump'. ..... 2079
6.4.1.20.1.5.8.5 SFC element 'Macro' ..... 2079
6.4.1.20.1.5.8.6 SFC element properties. ..... 2080

## SFC elements 'Step' and 'Transition'

Step symbol †; Transition symbol +
As a rule, CODESYS inserts steps and transitions as combinations. Inserting a step without a transition or a transition without a step causes an error when compiling. You can modify this by double-clicking the name.


## NOTICE!

Step names must be unique within the scope of the parent block. Consider this especially when using actions that were also programmed in SFC.

Please note that you can convert a step into an initial step by clicking "Init step" or by setting the respective property in the SFC properties.
All steps are defined by the step properties, which you can display and edit in the "Properties" view, depending on the set options.

You have to add those actions to the step which are to be executed when the step is active. A distinction is made between IEC actions and step actions. Details for this are found in the chapter about the SFC element "Action".
A transition must include the condition for the subsequent step to be active as soon as the value of the condition yields TRUE. Therefore, a transition condition must yield TRUE or FALSE. It can be defined in one of two ways:

- (1) Inline condition (direct): You replace the default transition name with either the name of a Boolean variable, a Boolean address, a Boolean constant, or a statement with a Boolean result, for example (i<100) AND b. You cannot specify programs, function blocks, or assignments here.
- (2) Multi-use condition (separate transition or property object): You replace the default transition name with the name of a transition or property object (㖘, 䝑). You create these objects by clicking "Project $\rightarrow$ Add Object". This allows multiple use of transitions, for example "condition_xy" in the figures below. Like an inline condition, the object can contain a Boolean variable, Boolean address, Boolean constant, or an statement with a Boolean result. In addition, it can also contain multiple statements with any code.



## - NOTICE!

The user is responsible for assigning the required expression to a transition
variable if the transition includes multiple instructions.

Transitions that reference a transition or property object are marked with a small triangle in the upper right corner of the transition box.


As opposed to CoDeSys V2.3, now CODESYS treats a transition condition like a method call. The entry has the following syntax:

```
<transition name>:=<transition condition>
```

(for example trans1:=a=100)
or only
<transition condition>
(for example $a=100$ )
You will find an example (condition_xy) in the figure above.
See also

- ${ }^{*}>$ Chapter 6.4.1.9.5.5.2 "Programming in SFC" on page 1880
- $\quad \stackrel{y}{ }$ Chapter 6.4.1.21.3.12.6 "Command 'Insert Step-Transition'" on page 2677
- $\xi^{*}$ Chapter 6.4.1.20.1.5.8.2 "SFC Element 'Action'" on page 2075
- $\quad 4$ Chapter 6.4.1.21.3.12.1 "Command 'Init Step'" on page 2675
- $\Leftrightarrow$ Chapter 6.4.1.20.1.5.8.6 "SFC element properties" on page 2080
- $\Rightarrow$ Chapter 6.4.1.9.22.5 "Calling methods" on page 1933


## SFC Element 'Action'

Symbol: 気
An action includes one or more statements in one of the valid implementation languages. You can assign an action to a step.
Actions that you use in SFC steps have to be created as POUs in the project.


Exception: In the case of IEC actions, which you add to a step as action association, you can also specify a Boolean variable instead of an action object. The value of these variables is switched between FALSE and TRUE each time the action is executed.

## NOTICE!

You have to define unique step names within the scope of the parent block. An action written in SFC must not contain a step with a name identical to the step to which the action is assigned.

A distinction is made between IEC actions and step actions:

1. IEC actions IEC actions comply with the IEC 61131-3 standard. They are executed according to their qualifiers.
IEC actions are executed two times: first when the step is activated and second when the step is deactivated. If you assign multiple actions to one step, then the action list is processed from top to bottom.
Each action box includes the qualifier in the first column and the action name in the second column. Both can be edited directly.


## NOTICE!

When the same global Boolean variable is associated as an IEC action in different SFC POUs, unwanted overwriting can result.

In contrast to step actions, you can use different qualifiers for IEC actions. Moreover, each IEC action is provided with a control flag. This directs CODESYS to execute an action only one time at any moment, even if the action is called by another step at the same time. This cannot be guaranteed for step actions.

You assign IEC actions to steps by clicking "SFC $\rightarrow$ Insert Action Association".
See also

- ② Chapter 6.4.1.21.3.12.14 "Command 'Insert Action Association'" on page 2680
- $\Leftrightarrow$ Chapter 6.4.1.20.1.5.4 "Qualifiers for Actions in SFC" on page 2065

2. Step actions These are actions that you can use to extend the IEC standard.

- Entry action:

CODESYS executes this action after the step is activated and before the main action is executed.
You reference a new action, or an action created below the SFC object, from a step by means of the "Entry action" element property (2). You can also add a new action to the step by means of the "Add Entry Action" command. The entry action is marked with an E in the lower left corner of the step box.

- Main action:

CODESYS executes this action when the step is active and any entry actions have already been processed. However, in contrast to IEC actions (see above), these step actions are not executed a second time when the step is deactivated. Moreover, you cannot use qualifiers here.
You add an existing action to a step by means of the "Main action" element property (1). You can create and add a new action by clicking the step element. A main action is marked with a filled triangle in the upper right corner of the step box.

## - Exit action:

CODESYS executes this action one time when the step is deactivated. However, note that an exit action is not executed in the same cycle, but at the beginning of the next cycle. You reference a new action, or an action created below the SFC object, from a step by means of the "Exit action" element property (3). You can also add a new action to the step by means of the "Insert Exit Action" command. The exit action is marked with an x in the lower right corner of the step box.


See also

- ${ }^{4}$ Chapter 6.4.1.20.1.5.8.6 "SFC element properties" on page 2080

Difference between IEC actions and step actions

The main difference between step actions and IEC actions with a qualifier $N$ is that an IEC action is always executed two times: when the step is activated and when the step is deacti-


You have attached the Action_AS1 action to the AS1 step as a step action (left) and as an IEC action with qualifier $N$ (right). Because two transitions are activated in each case, the time to reach the initial step again is two PLC cycles. This is true as long as the iCounter counter variable was initialized at 0 and then incremented in the Action_AS1 action. After the Init step is reactivated, iCounter returns a value of 1 in the example on the left. In the example on the right, a value of 2 is returned because the IEC action is executed a second time due to the deactivation of AS1.

Another difference: Step actions can be pseudo-embedded. In this case, they can be called only from the related step. If you copy this step, CODESYS creates new action objects automatically and copies the respective implementation code. You define whether or not a step action is embedded, either when the first action is inserted into the step, or later in the "Duplicate when copying" element property. In general, this behavior can also be preset in the SFC options.
Moreover, for IEC actions, a Boolean variable can be specified instead of an action object. This is not possible for step actions.

## SFC element 'Branch'

## Symbol 占

Use branches to program parallel or alternative sequences in the sequential function chart.
For alternative branches, CODESYS processes just one of the branch lines at a time, depending on the preceding transition condition. Parallel branches are processed at the same time.
See also

- \# Chapter 6.4.1.20.1.5.3 "Processing order in SFC" on page 2064
- \&Chapter 6.4.1.9.5.5.2 "Programming in SFC" on page 1880
- $\xi^{*}$ Chapter 6.4.1.21.3.12.13 "Command Insert Branch Right"' on page 2679

Parallel branch
For parallel branches, the branch lines must begin and end with steps. Parallel branch lines can contain additional branches.

The horizontal lines before and after the branch are double lines.


Processing in online mode: If the preceding transition (t2 in the example) yields TRUE, then the first steps in all parallel branch lines are active (Step11 and Step21). CODESYS processes the individual branch lines at the same time and the subsequent transition is passed afterwards ( t 3 ).
The "Branch<n>" jump marker is added automatically to the horizontal line that indicates the beginning of a branch. You can define this marker as the jump destination.
Please note that you can convert a parallel branch into an alternative branch by clicking "Alternative".
See also

- ② Chapter 6.4.1.21.3.12.11 "Command 'Alternative'" on page 2679

[^32]

If the step before the branch is active, then CODESYS passes the first transition of each alternative branch line from left to right. For the first transition that yields TRUE, the associated branch line opens, thus activating the step following the transition.

Please note that you can convert an alternative branch into a parallel branch by clicking "Parallel".

See also

- $\quad 4$ Chapter 6.4.1.21.3.12.10 "Command 'Parallel"' on page 2678


## SFC element 'Jump'

## Symbol $\downarrow$

Use a jump to define which actions in a step should be executed next as soon as the transition preceding the jump is TRUE. Jumps may become necessary, as execution paths cannot cross or lead upwards.
Excluding the required jump at the end of a diagram, you can generally insert jumps only at the end of a branch.

The destination of a jump is defined by the added text string, which you can edit directly. The jump destination can be a step name or the marker for a parallel branch.


See also

- ※ Chapter 6.4.1.9.5.5.2 "Programming in SFC" on page 1880
- ⓨ Chapter 6.4.1.21.3.12.16 "Command 'Insert Jump'" on page 2681


## SFC element 'Macro'

Symbol 罞

A macro includes part of the SFC diagram, but it is not displayed in detail in the main view of the editor.
Using macros does not influence the processing flow. Macros are used for hiding specific parts of the diagram, for example to increase overall clarity.

You open the macro editor by double-clicking the macro box or by clicking "SFC
$\rightarrow$ Zoom Into Macro". You can program here just like in the main view of the SFC editor. To close the macro editor, click "SFC $\rightarrow$ Zoom Out of Macro".

(1) Main view in the SFC editor
(2) Macro editor view for Macro1

Macros can also include other macros. The caption of the macro editor always shows the path of the open macro within the diagram, for example:


See also

- $\xi^{*}$ Chapter 6.4.1.9.5.5.2 "Programming in SFC" on page 1880
- $\&$ Chapter 6.4.1.21.3.12.20 "Command 'Zoom Into Macro'" on page 2682
- $\quad 4$ Chapter 6.4.1.21.3.12.21 "Command 'Zoom Out of Macro"' on page 2682


## SFC element properties

You edit the properties of an SFC element in the "Properties" view. Click "View $\rightarrow$ Element Properties" to open this view. The properties to be displayed depend on the currently selected element.

The properties that are displayed in the SFC diagram next to the element depend on the settings in the "View" tab of the SFC editor options.

General

| Property | Value description |
| :--- | :--- |
| "Name" | Element name, by default "<element><consecutive number>", for example step <br> name "Step0", "Step1", branch name "Branch0", etc. |
| "Comment" | Element comment in text, for example "counter reset". You can insert line breaks <br> by pressing [Ctrr]+[Enter]. |
| "Symbol" | For each SFC element, CODESYS declares an implicit variable with the same <br> name as the element. <br> The configuration determines whether this flag variable should be exported to <br> the symbol configuration and which access rights for the symbol should be <br> applied in the PLC. <br> - "No access": The symbol is exported to the symbol configuration but cannot <br> be accessed from the PLC. <br> - "Read": The symbol is exported to the symbol configuration and can be read <br> from the PLC. |
|  | "Write": The symbol is exported to the symbol configuration and can be <br> - written from the PLC. <br> "Read/Write": Combination of read and write. <br> - Empty: A symbol is not exported to the symbol configuration. |

## Specific

| Property | Value description |
| :--- | :--- |
| "Init step" | V: This option is activated only for the defined initial step. By default, this is the <br> first step in an SFC diagram. <br> Note: If you activate this property for another step, then it must be deactivated in <br> the previous step to prevent compilation errors. |
| "Duplicate when copying" | This option is available for steps that contain a step action (entry action, main <br> action, or exit action), and for transitions that are linked to a transition object. <br> V: When copying the step or transition, a new object is created for each called <br> action or transition. It contains a copy of the implementation code of the copied <br> object. <br> $\square:$ When copying a step or transition, the link to the called object is retained for <br> the respective action or transition. No new objects are generated. The source <br> and the copies of the step or transition call the same action or transition. |


| Property | Value description |
| :---: | :---: |
| "Times" <br> - "Minimum active" <br> - "Maximum active" | Minimum time that the step is active, even when the subsequent transition is TRUE. <br> Maximum time that the step can be active. If this time is exceeded, then CODESYS sets the SFCError implicit variable to TRUE. <br> Times according to IEC syntax (for example t\#8s) or the TIME variable; default: t\#Os. |
| "Actions" <br> - "Entry action" <br> - "Step action" <br> - "Exit action" | - "Entry action": CODESYS executes these actions after activating the step. <br> - "Step action": CODESYS executes this action when the step is active and any entry actions have already been processed. <br> - "Exit action": CODESYS executes this action in the subsequent cycle when the step is deactivated. <br> Please note the processing sequence. |

When using the respective implicit SFC variables and flags, you receive information about the status of a step or an action or about timeouts.

## See also

- ${ }^{\mu}$ Chapter 6.4.1.21.4.14.23 " Dialog 'Options' - 'SFC Editor"' on page 2795
- ${ }^{4}$ Chapter 6.4.1.20.1.5.5 "Implicit variables" on page 2066
- $\wedge^{\circ}$ Chapter 6.4.1.20.1.5.8.2 "SFC Element 'Action"" on page 2075

Function Block Diagram / Ladder Diagram / Instruction List (FBD/LD/IL)
6.4.1.20.1.6.1 FBD/LD/IL Editor..................................................................... 2082
6.4.1.20.1.6.2 FBD/LD/IL editor in online mode............................................ 2086
6.4.1.20.1.6.3 Modifiers and operators in IL................................................... 2087
6.4.1.20.1.6.4 Elements................................................................................. 2091

## FBD/LD/IL Editor

The FBD/LD/IL editor is a combined editor of the programming languages FBD, LD and IL.

## If necessary, IL can be activated in the CODESYS options.

There is a common set of commands and elements and CODESYS automatically converts the 3 programming languages into one another internally.

The code in the implementation part is structured in all three languages with the aid of networks. The "FBD/LD/IL" menu provides the commands for working in the editor.
In offline and online modes, you can switch editors at any time by using the menu command in "View".

The behavior of the FBD/LD/IL editor is defined by the settings in "Tools $\rightarrow$ Options" (category "FBD, LD and IL").

## NOTICE!

There are some special elements that CODESYS cannot convert and thus it displays only in the applicable language. There are also constructs that are not clearly convertible between IL and FBD and are therefore 'normalized', i.e. nullified, when converted back to FBD. This concerns: negation of expressions and explicit/implicit assignment of function block inputs and outputs.

An error-free conversion between the languages requires syntactically correct code. Otherwise parts of the implementation can be lost.

## See also

- $y^{*}$ Chapter 6.4.1.9.5.2 "FBD/LD/IL" on page 1860 (programming)
-     * Chapter 6.4.1.21.3.14 "Menu 'FBD/LD/IL'" on page 2700 (commands)
- $\quad y^{\sharp}$ Chapter 6.4.1.21.4.14.10 "Dialog 'Options' - 'FBD, LD, and IL'" on page 2787
- « Chapter 6.4.1.20.1.3 "Common functions in graphical editors" on page 2048

FBD and LD Inserting and arranging elements
You can drag elements with the mouse from the view "Tools" (toolbox) into the implementation part of the editor. Alternatively you can use the commands of the context menu or the "FBD/LD/IL" menu.
Settings for the display and interface are defined in the CODESYS options, category "FBD/LD/ IL".

If you drag an element with the mouse over a network in the editor, all possible insertion positions are displayed with gray diamond-shaped, triangular or arrow-shaped position marks. As soon as the mouse pointer is located over one of these marks, the mark turns green. If the mouse button is now released, CODESYS inserts the element at this position.


If you drag a function block or an operator from the toolbox or a network at the left-hand side of the network onto one of the two arrows, then CODESYS automatically creates a new network and inserts the element there.

In order to replace an element, drag a suitable other element onto its position with the mouse. Elements that you can replace by the new element are marked by CODESYS in the editor with text fields, for example "Replace", "Attach input".

You can use the usual commands in the menu "Edit" for cutting, copying, pasting and deleting elements. Copying also works with drag-and-drop by holding down the [Ctrl] key.

## NOTICE!

The operators with EN/ENO functionality can only be inserted in the FBD and IL editors.

## Selecting elements

A box or a connecting line in the editor is selected by clicking on it with the mouse so that it has the focus. Multiple selection is possible by keeping the [Ctrl] key pressed. A selected element is shaded red.

## Tooltip

If the cursor points to certain elements, for example to a variable or to an input, a tooltip appears showing information about this element.
In the case of elements underlined with a wavy red line, the tooltip shows the pre-compile error message of the error that occurs with this element.

## Navigating in the editor

Table 390: Navigating in the editor

| With the help of the keys and commands described below, you can place the focus within the editor on a different cursor position. The change between the positions is also network-spanning in function. |  |
| :---: | :---: |
| $\begin{aligned} & {[\leftarrow]} \\ & {[\rightarrow]} \end{aligned}$ | Change to the neighboring cursor position, along the signal flow, i.e. from left to right and vice versa. |
| $\begin{aligned} & {[\uparrow][\downarrow]} \\ & {[\downarrow]} \end{aligned}$ | Change to the next cursor position above or below the current position, if this neighboring position belongs to the same logical group. For example, all connections of a box form a logical group. <br> If such a logical group does not exist: change to the first cursor position in the next higher or lower neighboring element. In the case of parallel-connected elements, navigation takes place along the first branch. |
| [Ctrl] + [Home] | Change to the first network; this will be selected. |
| [Ctrl] + [End] | Change to the last network; this will be selected. |
| [Page Up] | Scroll upwards by one page; the highest network on this page is selected. |
| [ Page Down] | Scroll downwards by one page; the lowest network on this page is selected. |
| Command "Go to..." | Change to a certain network. |

## Opening function blocks

If a function block is inserted in the editor, then you can open its implementation by a doubleclick or with the context menu command "Browse for symbol $\rightarrow$ Go to Definition".
See also

- « "Function block diagram (FBD)" on page 1860
- «乡 "Ladder diagram (LD)" on page 1860
- $\Rightarrow$ Chapter 6.4.1.9.5.2.2 "Programming function block diagrams (FBD)" on page 1862
- $\Leftrightarrow$ Chapter 6.4.1.9.5.2.3 "Programming ladder diagrams (LD)" on page 1864
- ${ }^{4}$ Chapter 6.4.1.20.1.6.4 "Elements" on page 2091
- $y^{*}>$ Chapter 6.4.1.21.4.14.10 "Dialog 'Options' - 'FBD, LD, and IL'" on page 2787
- $\Leftrightarrow$ Chapter 6.4.1.20.1.6.2 "FBD/LD/IL editor in online mode" on page 2086
- $\quad$ Chapter 6.4.1.21.3.14.44 "Command 'Go to"' on page 2712


## IL editor Inserting and arranging elements:

You can insert elements with the help of the commands of the menu "FBD/LD/IL" in the context menu. You can also drag a new network from the tool box into the implementation part of the editor by drag-and-drop.
You can use the usual commands in the menu "Edit" for cutting, copying, pasting and deleting elements. Copying also works with drag-and-drop by holding down the [Ctrl] key.

## NOTICE!

Please note that operators with EN/ENO functionality can only be inserted in the FBD and IL editors.

Each program line is entered in a table row.

Table 391: Structure of networks in the IL editor
1st line: Network title
Requirement: The option is activated in the CODESYS options.

2nd line: Network comment
Requirement: The option is activated in the CODESYS options.
3rd line and on:

| Column | Contents | Description |
| :--- | :--- | :--- |
| 1 | Operator | Contains the IL operator (LD, ST, CAL, AND, OR, <br> etc.) or a function name. If you call a function block, <br> you must additionally specify the corresponding <br> parameters here; in the preceding field you must <br> enter in this case $:=$ or $=>$. |
| 2 | Operand | Contains precisely one operand or the name of a <br> jump label. <br> In the case of several operands you must enter <br> them in several rows and when doing so insert a <br> comma directly behind the individual operands. (See <br> example below) |
| 3 | Address | Contains the address of the operand as defined in its <br> declaration | declaration.

non-editable
You can activate/deactivate the display via the option "Display symbol address". To do this, select the command "Tools $\rightarrow$ Options" and the "General" tab in the category "FBD, LD and IL".
Contains the comment that was specified for the operand if necessary in the declaration.
non-editable
You can activate/deactivate the display via the option "Display symbol comment" if you select the command "Tools $\rightarrow$ Options" and the "General" tab in the category "FBD, LD and IL".

Comment for the current program line.
You can activate/deactivate the display via the option "Operand comment" if you select the command "Tools $\rightarrow$ Options" and the "General" tab in the category "FBD, LD and IL".

Table 392: Navigating in the editor

| Key(s)/ <br> command | Cursor movement |
| :--- | :--- |
| $[\uparrow]$ <br> $[\downarrow]$ | Jumps to the field located above/below. |
| $[$ Tab] | Jumps one field to the right within the row. |
| $[$ Shift $]+[$ Tab] $]$ | Jumps to the left to the preceding field within the row |
| $[$ Space $]$ | Opens the editing frame for the selected field. Alternatively you can click <br> with the mouse on the field. If applicable the button for the input assistant <br> dialog box is available. |
| $[$ Ctrl] + [Enter] | Inserts a new row below the current row. |
| $[$ Del $]$ | Deletes the current row. |
| [Ctrl] + [Home] | Sets the focus at the start of the document and marks the first network. |
| [Ctrl] + [End] | Sets the focus at the end of the document and marks the last network. |
| [Page Down] | Scrolls up by one page and marks the top rectangle. |
| $[$ Page Up] | Scrolls down by one page and marks the top rectangle. |

See also

- « "Instruction list (IL)" on page 1861
- ${ }^{\bullet}$ Chapter 6.4.1.9.5.2.4 "Programming in instruction list (IL)" on page 1865
- $\&$ Chapter 6.4.1.20.1.6.3 "Modifiers and operators in IL" on page 2087
- $y^{*}>$ Chapter 6.4.1.21.4.14.10 "Dialog 'Options' - 'FBD, LD, and IL'" on page 2787
- $\Leftrightarrow$ Chapter 6.4.1.20.1.6.2 "FBD/LD/IL editor in online mode" on page 2086


## FBD/LD/IL editor in online mode

FBD/LD/IL editor In online mode the current value of each variable is displayed behind the variable in the editor. in online mode Writing/forcing and the setting of breakpoints is possible.

If the variable is presently forced, this is indicated directly in front of the forced value by $\mathbf{F}$. If a value has been prepared for writing or forcing, this value is displayed directly behind the current value in square brackets <value>.

## Example Forced variable:

## BVar1 ETRUE

Prepared value


In the online view of a ladder diagram (LD) the connecting lines are marked in color: connections with the value TRUE are displayed as a thick blue line, connections with the value FALSE as a thick black line. Conversely, connections with an unknown or analog value are displayed normally (thin black line).

## NOTICE!

Note that values of the connections are calculated from the monitored variables.
This is not a genuine flow control.

## Breakpoints

Possible positions for breakpoints are in principle the positions at which values of variables can change (instructions), at which the program branches or at which another box is called.
Possible breakpoint positions:

- On the entire network: causes the breakpoint to be set at the first possible position in the network.
- On a box, if the box contains an assignment. Not possible with operator boxes, for example ADD, DIV.
- On assignments.
- At the end of the box at the position of the return to the calling box. In online mode an empty network automatically appears here; it is marked by 'RET' in place of a network number.


## NOTICE!

At present you cannot directly set a breakpoint on the first box in the network. However, if you set a breakpoint on the entire network, this breakpoint marking is transferred automatically to the first box in online mode.

## NOTICE!

Breakpoints in methods: CODESYS automatically sets a breakpoint in all methods that can be called. Therefore, if a method managed by an interface is called, breakpoints are set in all methods that occur in function blocks that implement this interface as well as in all derived function blocks that use the method. If a method is called by a pointer to a function block, CODESYS sets the breakpoints in the method of the function block and in all derived function blocks that use the method.

See also

- ${ }^{\text {® }}$ Chapter 6.4.1.12.5 "Forcing and Writing of Variables" on page 1987
- $\quad 4$ Chapter 6.4.1.12.3 "Using Breakpoints" on page 1981


## Modifiers and operators in IL

Table 393: Modifiers

| Modifier | Combined with oper- <br> ator | Description |
| :--- | :--- | :--- |
| C | JMP, CAL, RET | The command is only executed if the result of <br> the preceding expression is TRUE. |
| N | JMPC, CALC, RETC | The command is only executed if the result of <br> the preceding expression is FALSE. |
| N | otherwise | negation of the operand (not of the accumu- <br> lator). |

Table 394: Operators with the possible modifiers

| Operator | N | Meaning | Example |
| :--- | :--- | :--- | :--- |
| LD | N | Loads the (negated) the value of the operand <br> into the accumulator. | LD ivar |
| ST | N | Stores the (negated) content of the accumu- <br> lator in the operand. | ST iErg |
| S | Sets the operand (type BOOL) to TRUE if the <br> content of the accumulator is TRUE. | S bVar1 |  |


| Operator | N | Meaning | Example |
| :---: | :---: | :---: | :---: |
| R |  | Sets the operand (type BOOL) to FALSE if the content of the accumulator is TRUE. | R bVar1 |
| AND | N, ( | Bitwise AND of the accumulator value and (negated) operand | And bVar2 |
| OR | N, ( | Bitwise OR of the accumulator value and (negated) operand | OR xVar |
| XOR | N, ( | Bitwise exclusive OR of the accumulator value and (negated) operand | $\begin{aligned} & \text { XOR N, } \\ & \text { (bVar1,bVa } \\ & \text { r2) } \end{aligned}$ |
| NOT |  | Bitwise negation of the accumulator value |  |
| ADD | 1 | Addition of the accumulator value and the operand; result is written into the accumulator. | ADD ivar1 |
| SUB | 1 | Subtraction of the operand from the accumulator value; result is written into the accumulator. | SUB iVar2 |
| MUL | 1 | Multiplication of accumulator value and operand; result is written into the accumulator. | MUL ivar2 |
| DIV | 1 | Division of the accumulator value by the operand; result is written into the accumulator. | DIV 44 |
| GT | 1 | Checks whether the accumulator value is greater than the operand value; result (BOOL) is written into the accumulator; > | GT 23 |
| GE | 1 | Checks whether the accumulator value is greater than or equal to the operand value; result (BOOL) is written into the accumulator. | GE iVar2 |
| EQ | 1 | Checks whether the accumulator value is equal to the operand value; result (BOOL) is written into the accumulator. | EQ iVar2 |
| NE | 1 | Checks whether the accumulator value is not equal to the operand value; result (BOOL) is written into the accumulator; | NE iVar1 |
| LE | 1 | Checks whether the accumulator value is smaller than or equal to the operand value; result (BOOL) is written into the accumulator. | LE 5 |
| LT | 1 | Checks whether the accumulator value is smaller than the operand value; result (BOOL) is written into the accumulator. | LT cVar1 |
| JMP | CN | Unconditional (conditional) jump to the specified jump label | JMPN next |
| CAL | CN | (Conditional) call of a program or a function block (if the accumulator value is TRUE) | CAL prog1 |
| RET |  | Exit the box and return to the calling box. | RET |
| RET | C | If the accumulator value is TRUE: exit the box and return to the calling box. | RetC |
| RET | CN | If the accumulator value is FALSE: exit the box and return to the calling box. | Retcn |
| $)$ |  | Evaluation of the reset operation |  |




| Application | Description | Examples |
| :---: | :---: | :---: |
| Function block call, program call | Column 1: Operator CAL or CALC <br> Column 2: Name of the function block instance or the program and opening parenthesis (. If no parameters follow, the closing parenthesis ) is entered here. <br> rows following that: <br> Column 1: parameter name followed by := for input parameter or => for output parameter <br> Column 2: parameter value followed by a comma, if further parameters follow. The closing parenthesis ) is input after the last parameter. <br> As a limitation according to the IEC standard, complex expressions cannot be used here. You must assign such constructs to the function block or the program before the call. |  |
| Function Call | Row 1: Column 1: LD <br> Column 2: input variable <br> Row 2: Column 1: Function name Column 2: further input parameters separated by commas. <br> CODESYS writes the return value into the accumulator. <br> Row 3: Column 1: ST Column 2: variable into which the return value is written | LD X7 <br> GeomAverage 25 <br> ST Ave |
| Action call | Like function block call or program call. <br> The action name is appended to the name of the FB instance or the program. | CAL POO1.Resetaction |


| Application | Description | Examples |  |
| :--- | :--- | :--- | :--- |
| Jump | Column 1: operator JMP or <br> JMPC. <br> Column 2: Name of the jump <br> label of the destination net- <br> work. <br> In the case of an uncondi- <br> tional jump, the preceding <br> instruction sequence must <br> end with one of the following <br> commands: ST, STN, S, R, <br> CAL, RET, JMP | LD | JMPC |
| In the case of a conditional <br> jump the execution of the |  |  |  |
| jump depends on the loaded |  |  |  |
| value. |  |  |  |$\quad$|  |
| :--- |

See also

- ${ }^{*}$ Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082
- ${ }^{\text {B }}$ "Instruction list (IL)" on page 1861
- ${ }^{\bullet}$, Chapter 6.4.1.9.5.2.4 "Programming in instruction list (IL)" on page 1865


## Elements

6.4.1.20.1.6.4.1 FBD/LD/IL element 'Network'. ..... 2091
6.4.1.20.1.6.4.2 FBD/LD/IL element 'Box'. ..... 2092
6.4.1.20.1.6.4.3 FBD/LD/IL element 'Assignment'. ..... 2092
6.4.1.20.1.6.4.4 FBD/LD/IL element 'Box with EN/ENO'. ..... 2092
6.4.1.20.1.6.4.5 FBD/LD/IL element 'Input'. ..... 2093
6.4.1.20.1.6.4.6 FBD/LD/IL element 'Label'. ..... 2093
6.4.1.20.1.6.4.7 FBD/LD/IL element 'Jump'. ..... 2093
6.4.1.20.1.6.4.8 FBD/LD/IL element 'Return'. ..... 2093
6.4.1.20.1.6.4.9 FBD/LD/IL element 'Branch'. ..... 2093
6.4.1.20.1.6.4.10 FBD/LD/IL element 'Execute' ..... 2094
6.4.1.20.1.6.4.11 LD element 'Contact' ..... 2094
6.4.1.20.1.6.4.12 LD element 'Coil'. ..... 2095
6.4.1.20.1.6.4.13 LD element 'Branch Start/End' ..... 2095
6.4.1.20.1.6.4.14 Closed branch. ..... 2096

## FBD/LD/IL element 'Network'

## Symbol [险

A network is the base unit of an FBD or LD program. In the FBD/LD/IL editor, the networks are arranged in a list. Each network is provided with a sequential network number on the left side and can include: logical and arithmetic expressions, program/function/function block calls, jumps, or return statements.
An IL program consists of at least one network. This network can include all IL statements of the program.
You can provide each network with a title, comment, or label. In the CODESYS options (category "FBD, LD, and IL", you can define whether network title, comment, and separator between individual networks are displayed in the editor.
Click the first line of the network to enter a network title. Click the second line of the network to enter a network comment.

See also

- ${ }^{*} \boldsymbol{y}$ Chapter 6.4.1.21.4.14.10 "Dialog 'Options' - 'FBD, LD, and IL'" on page 2787
- ② Chapter 6.4.1.21.3.14.1 "Command 'Insert Network'" on page 2700


## FBD/LD/IL element 'Box'

Symbol: 泎
A box and its call can represent additional functions, for example IEC function blocks, IEC functions, library function blocks, operators.
A box can have any number of inputs and outputs.
If the box also provides an image file, the box icon is displayed inside the box. The requirement is that the option "Show box symbol" is activated in the CODESYS options, category "FBD, $L D$ and $I L$ ".

If you have changed the box interfaces, you can update the box parameters with the command "FBD/LD/IL $\rightarrow$ Update Parameters" without having to re-insert the box.

See also

- ${ }^{*}$ Chapter 6.4.1.21.3.14.5 "Command 'Insert Box"' on page 2701
- ${ }^{*} \boldsymbol{y}$ Chapter 6.4.1.21.4.14.10 "Dialog 'Options' - 'FBD, LD, and IL'" on page 2787
- $\quad$ Chapter 6.4.1.21.3.14.38 "Command 'Update Parameters"' on page 2710


## FBD/LD/IL element 'Assignment'

Symbol: -van
The FBD editor shows a newly inserted assignment as a line with 3 question marks after it. The LD editor shows a newly inserted assignment as a coil with 3 question marks located above it.
After insertion you can replace the placeholder ? ? ? by the name of the variable to which the signal coming from the left is to be assigned. The input assistant is available to you for this.

In IL an assignment is programmed via the operators LD and ST.


- Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082
- $\quad \stackrel{y}{c}$ Chapter 6.4.1.21.3.14.4 "Command 'Insert Assignment"' on page 2701


## FBD/LD/IL element 'Box with EN/ENO'

Symbol: 監
The element is available only in the FBD and LD editors.
The box generally corresponds to the FBD/LD/IL element "Box"; however, this box additionally contains an EN input and an ENO output. EN and ENO have the data type BOOL.
Function of the EN input and ENO output: if the input EN has the value FALSE at the time of the calling the box, the operations defined in the box are not executed. Otherwise, i.e. if EN has the value TRUE, these operations are executed. The ENO output has the same value as the EN input.
See also

- $y^{2}$ Chapter 6.4.1.21.3.14.6 "Command 'Insert Box with EN/ENO'" on page 2702
- $y^{4}$ Chapter 6.4.1.20.1.6.4.2 "FBD/LD/IL element 'Box"' on page 2092


## FBD/LD/IL element 'Input'

Symbol: 涣
The maximum number of inputs depends on the type of box.
A newly added input is first marked with ? ? ?. You can replace the string ?? ? by a variable or a constant.

See also

- \& Chapter 6.4.1.21.3.14.13 "Command 'Insert Input"" on page 2703
- $\quad$ Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082


## FBD/LD/IL element 'Label'

The label is an optional identifier for a network in FBD and LD, which you can specify as a destination for a jump.

If you insert a jump label in a network, it will be added as an editable field "Label:" in the network.

See also

- $\&$ Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082
- $\Leftrightarrow$ Chapter 6.4.1.21.3.14.11 "Command 'Insert Label'" on page 2703


## FBD/LD/IL element 'Jump'

Symbol $\rightarrow$
In FBD or LD a jump is inserted either directly before an input, directly after an output or at the end of the network, depending on the current cursor position.
You enter a jump label as the jump destination directly behind the jump element.
In IL you program a jump with the instruction JMP.
See also

- ${ }^{4}$ Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082
- \& Chapter 6.4.1.21.3.14.10 "Command 'Insert Jump"' on page 2703
- ${ }^{\circ} \boldsymbol{y}$ Chapter 6.4.1.20.1.6.4.6 "FBD/LD/IL element 'Label"" on page 2093


## FBD/LD/IL element 'Return'

This element immediately interrupts the execution of the box if the input of the RETURN element goes TRUE.
In an FBD or LD network you can place the Return instruction parallel to or after the preceding elements.

In IL the RET instruction is available to you for this purpose.
See also

- ${ }^{\text {• }}$, Chapter 6.4.1.21.3.14.12 "Command 'Insert Return'" on page 2703
- $\Leftrightarrow$ Chapter 6.4.1.20.1.6.3 "Modifiers and operators in IL" on page 2087


## FBD/LD/IL element 'Branch'

Symbol: ᄂ

The element is available in the LD and FBD editor and represents an open line branch. A line branch splits the processing line from the current cursor position onwards into 2 subnetworks, which are executed in succession from top to bottom. You can branch each subnetwork further, as a result of which multiple branches are created within a network.

Each subnetwork is given a marker symbol (rectangle) at the branch point, which you can select in order to execute further commands.

The commands "Copy", "Cut" and "Paste" are not available for subnetworks.


In order to delete a subnetwork, you must first delete all elements of the network and then the marker symbol of the subnetwork.
See also

- ② Chapter 6.4.1.21.3.14.33 "Command 'Insert Branch"' on page 2709
- ${ }^{\wedge} \boldsymbol{y}$ Chapter 6.4.1.21.3.14.34 "Command 'Insert Branch Above'" on page 2709
- $\boldsymbol{y}^{\boldsymbol{y}}$ Chapter 6.4.1.21.3.14.35 "Command 'Insert Branch Below'" on page 2709


## FBD/LD/IL element 'Execute'

## Symbol:

The element is a box that enables you to directly enter ST code in the FBD and LD editors.
You can drag the "Execute" element with the mouse from the "Tools" view into the implementation part of your POU. If you click on "Enter ST code here...", an input field opens where you can input multiple-line ST code.

- $\quad$ Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082


## LD element 'Contact'

Symbol: $\sqrt[n r]{ }$, in the editor』』
The element is available only in the LD editor.

A contact passes on the signal TRUE (ON) or FALSE (OFF) from left to right until the signal finally reaches a coil in the right-hand part of the network. For this purpose a boolean variable containing the signal is assigned to the contact. To do this, replace the placeholder ? ? ? above the contact with the name of a boolean variable.

You can arrange several contacts both in series and in parallel. In the case of two parallel contacts, only one needs to obtain the value TRUE in order for ON to be passed on to the right. If contacts are connected in series, all of them must obtain the value TRUE in order for ON to be passed on to the right by the last contact in the series. Hence, you can program electrical parallel and series connections with LD.
A negated contact $\rrbracket / \rrbracket$ forwards the signal TRUE if the variable value is FALSE. You can negate an inserted contact with the help of the command "FBD/LD/IL $\rightarrow$ Negation" or insert a negated contact from the "Tools" view.

If you place the mouse pointer on a contact with the left mouse button pressed and with a network selected, the button "Convert to coil" appears in the network. If you now move the mouse pointer onto this button, still with the mouse button pressed, and then release the mouse button over this button, CODESYS converts the contact into a coil.

See also

- ${ }^{\text {• }}$ Chapter 6.4.1.21.3.14.17 "Command 'Insert Contact"' on page 2704
- $\&$ Chapter 6.4.1.21.3.14.22 "Command 'Insert Negated Contact"' on page 2706
- $\Leftrightarrow$ Chapter 6.4.1.21.3.14.18 "Command 'Insert Contact (Right)"' on page 2705
- $\quad \stackrel{y}{c}$ Chapter 6.4.1.21.3.14.20 "Command 'Insert Contact in Parallel (Above)"' on page 2705
- ${ }^{\text {• }} \boldsymbol{y}$ Chapter 6.4.1.21.3.14.19 "Command 'Insert Contact in Parallel (Below)"' on page 2705
- $\quad 4$ Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082


## LD element 'Coil'

Symbol:««, in the editor ©
The element is available only in the LD editor.
A coil adopts the value supplied from the left and saves it in the boolean variable assigned to the coil. Its input can have the value TRUE (ON) or FALSE (OFF).

Several coils in a network can only be arranged in parallel.
In a negated coil ( $/ \downarrow)$ the negated value of the incoming signal is stored in the boolean variable that is assigned to the coil.

## Set coil, Reset coil


Set coil: If the value TRUE arrives at a set coil, the coil retains the value TRUE. As long as the application is running, the value can no longer be overwritten here.
Reset coil: If the value TRUE arrives at a reset coil, the coil retains the value FALSE. As long as the application is running, the value can no longer be overwritten here.

You can define an inserted coil as a set or reset coil with the help of the command "FBD/LD/IL $\rightarrow$ Set/Reset" or insert it as an element "Set Coil" and "Reset Coil" from the "Tools" view.

See also

- $\quad$ © Chapter 6.4.1.21.3.14.14 "Command 'Insert Coil" on page 2704

- $\xi^{\bullet}$ Chapter 6.4.1.21.3.14.29 "Command 'Negation"' on page 2708
- ${ }^{4}$ Chapter 6.4.1.21.3.14.31 "Command 'Set/Reset"' on page 2708


## LD element 'Branch Start/End'

The element serves the closed line branch.
See also

- ⓨ Chapter 6.4.1.20.1.6.4.14 "Closed branch" on page 2096
- « Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082
- ${ }^{\bullet}$, Chapter 6.4.1.21.3.14.36 "Command 'Set Branch Start Point'" on page 2709
- ${ }^{\circ}$ C Chapter 6.4.1.21.3.14.37 "Command 'Set Branch End Point"' on page 2710


## Closed branch

A closed branch is available in LD only, and it contains a starting point and an end point. It is used for implementing parallel analyses of logical elements.
Inserting a closed branch

- Command "FBD/LD/IL $\rightarrow$ Insert Contact Parallel (Below)"
- Command "FBD/LD/IL $\rightarrow$ Insert Contact Parallel (Above) "
- Command "FBD/LD/IL $\rightarrow$ Set Branch Start/End Point"


#### Abstract

Closed branch When you select one or more contacts and then execute the command "Insert Contact in at a contact Parallel", a parallel branch is added with a single vertical line. For this kind of branching, the signal flow passes through both branches. This is an OR construct of both branches.


Closed branch at a block, OR evaluation, or short-circuit evaluation

New: When you select a box and execute the command "Insert Contact in Parallel", a parallel branch is inserted with a double vertical line. This indicates that a short-circuit evaluation (SCE) is implemented. SCE allows for the execution of a function block with a Boolean output to be bypassed if a specific condition is TRUE. The condition can be displayed in the LD editor as a branch connected parallel to the function block branch. The short circuit condition is defined by one or more contacts in this branch that are interconnected parallel or sequentially.

## Functional principle:

The branches that do not include the function block are processed first. If CODESYS detects the value TRUE for one of these branches, then the function block is not called in the parallel branch. In this case, the value at the input of the function block is sent directly to the output. If CODESYS determines FALSE for the SCE condition, then the box will be called and the Boolean result of its processing is passed on. If all branches contain function blocks, they are analyzed from top to bottom and their outputs are logically ORed. If there are no branches with function blocks, normal OR operations are performed.

Example The function block instance x 1 (TON) has a Boolean input and a Boolean output. The execution of $x 1$ is skipped if TRUE is determined for the condition in the parallel line branch. The condition value results from the OR and AND operations that connect contacts cond1, cond2 and cond3.
x 1 is executed if the condition value from the connection of the contacts cond1, cond2 and cond3 is FALSE.

(1) Indicates from the double vertical connections that it is a construct subject to an SCE.
(2) Indicates from the single vertical connections that it is an OR construct.

The given LD example is shown below as ST code. P_IN and P_OUT are the Boolean values at the input (split point) and output (reunification point) of the parallel line branch.

```
P_IN := b1 AND b2;
IF ((P_IN AND cond1) AND (cond2 OR cond3)) THEN
    P_OUT := P_IN;
ELSE
    x1(IN := P_IN, PT := {p 10}t#2s);
    tElapsed := x1.ET;
    P OUT := x1.Q;
END_IF
bRes := P_OUT AND b3;
```

See also

- ${ }^{\mu}$ Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082
- $\sum^{*}$ Chapter 6.4.1.21.3.14.36 "Command 'Set Branch Start Point"" on page 2709
- ${ }^{4}$ Chapter 6.4.1.21.3.14.37 "Command 'Set Branch End Point"' on page 2710
- ${ }^{\Perp}$ Chapter 6.4.1.21.3.14.20 "Command 'Insert Contact in Parallel (Above)"' on page 2705
- 㘯 Chapter 6.4.1.21.3.14.19 "Command 'Insert Contact in Parallel (Below)"' on page 2705- 告 Chapter 6.4.1.21.3.14.21 "Command 'Toggle Parallel Mode'" on page 2706


## Continuous Function Chart (CFC) and Page-Oriented CFC

6.4.1.20.1.7.1 General ..... 2098
6.4.1.20.1.7.2 CFC Editor. ..... 2098
6.4.1.20.1.7.3 CFC editor, page-oriented ..... 2101
6.4.1.20.1.7.4 Keyboard Shortcuts in the CFC Editors ..... 2102
6.4.1.20.1.7.5 CFC Editor in Online Mode. ..... 2103
6.4.1.20.1.7.6 Elements ..... 2109

## General

From an external point of view, a Function Block Diagrams consists of inputs and outputs, with data being processed between them. From an internal point of view, a Function Block Diagrams consists of POUs and their connections which represent data (signals) and act as assignment operators in ST. The overall behavior is composed of the behavior of the inserted POUs which call other POUs or library POUs.

Code in the "Continuous Function Chart (CFC)" implementation language mainly illustrates the data flow through the system. Therefore, a continuous function chart is also referred to as a "signal flow chart".
In the page-oriented CFC editor, you can wire POUs to each other and create well-structured Function Block Diagrams distributed over multiple pages. The page-oriented editor behaves like the CFC editor, but provides the following functionality:

The page-oriented editor behaves like the CFC editor, but provides additional functionality. The editor supports you with the following functions:

- Creating pages
- Setting the page size
- Copying and inserting pages in the page navigator
- Copying the implementation of a POU in the CFC implementation language and inserting into a page
- Well-structured and space-saving arranging of inputs, outputs, and connection marks in the border areas
- Connection over pages with connection marks


## CFC Editor

Configuring the You can configure the appearance, behavior, and printing for the entire project in the CODESYS editor options in the "CFC Editor" category. For example, on the "View" tab, you can configure the color of the connecting lines depending on the data type.

## Editing

| Cursor symbol: | Requirement: "Pointer" is selected in the "Toolbox" view. <br> The symbol indicates that you can edit in the editor. Select elements or connec- <br> tions to move them or to execute commands. |
| :--- | :--- |
| Cursor symbol: + | Requirement: An element is selected in the "Toolbox" view. <br> Clicking in the editor inserts the selected element. You can also drag an element <br> to the editor. |
| Dragging a function block <br> instance from the declaration <br> to the editor | Requirement: A line is selected in the declaration of the CFC. <br> The instance is inserted as a box with name, type, and all pins. |
| Dragging a variable from the <br> declaration to a box pin to the <br> editor | The variable is inserted as an input or output with a connection to the box pin in <br> focus. <br> Hint: The cursor indicates when your focused location is valid for a variable: |


| Dragging a variable from the declaration part to the editor | Requirement: The respective element is selected in the declaration. <br> - Function block instance: A POU with the corresponding data type is created. <br> - Declaration of VAR_INPUT or CONSTANT: An input element is inserted. <br> - Declaration of VAR_OUTPUT: An output element is inserted. <br> - Declaration of VAR, VAR_GLOBAL: A window opens at the insert location, where you can select whether an input element or output element should be inserted. <br> When a variable is dragged from the declaration part to an existing replaceable element, the existing element is replaced. |
| :---: | :---: |
| Dragging a function block or POU to the editor from the "Devices" view, "POUs" view, or from the Library Manager. | A box element with the corresponding type is inserted. <br> - When a box is dragged to an existing connecting line and both the input and output of the box are compatible with the data type of the line, the box is inserted in the line. Here its first matching input and output are connected to the elements that were previously connected by the connecting line. <br> - When a box is dragged to an existing box, the existing box is replaced. |
| Resorting the order of inputs and outputs within a function block by means of drag\&drop | Requirement: The text field of the input or output, which should be resorted to another location, is selected. |
| [Ctrl] + click in the programing area | Requirement: An element is selected in the "Toolbox" view. <br> As long as you hold down the [Ctrl] key, a selected element is created each time you click in the programming area. |
| [CtrI]+[Right Arrow] | Requirement: In the CFC program, exactly one output pin is selected for an element. <br> The selection is moved so that the input pin at the end of the connecting line is selected. In the case of multiple pins, all are selected. |



See also

- « Chapter 6.4.1.20.1.3 "Common functions in graphical editors" on page 2048

| Connecting | You can insert connecting lines between element connections. Connecting lines are inserted <br> by means of auto-routing so that connecting lines are automatically optimal and as short as <br> possible. The connecting lines are checked for collisions. |
| :--- | :--- |


| Dragging a pin to another | A connecting line is inserted between the two element pins. |
| :---: | :---: |
| Dragging a pin to a function block | Dropping can be done on a pin or on the text field of a pin. <br> In the case of extendable operators (example: ADD), dropping can also be done within the box. The following behavior applies for this: If there are still unconnected input pins, then the top free pin is connected. If there are no more unconnected input pins, then a new pin is automatically inserted below. |
| Command "Connect Selected Pins" | Requirement: Multiple pins are selected. The pins are marked in red. |
| Move an inserted element so that it touches the pin of another element. | Requirement: The "Enable AutoConnect" option is selected. The touching pins are connected automatically. |
| [. | The connection icon is located in the upper right corner of the editor. A green icon indicates collision-free connections. A red icon indicates collisions. Clicking the icon opens a menu with commands for collision processing, for example the "Show Next Collision" command. |
| $\rightarrow \mathrm{C}-1$ | Requirement: A connection is selected and the "Connection Mark" command is executed. <br> Instead of a long connecting line, a connection is represented by connection marks. |

See also

- ② Chapter 6.4.1.21.3.13.22 "Command 'Show Next Collision'" on page 2694


## Commands when editing

See also


See also

- " ${ }^{4}$ Chapter 6.4.1.20.1.3 "Common functions in graphical editors" on page 2048
- « Chapter 6.4.1.9.5.3.3 "Programming in the CFC editor" on page 1871
- \& Chapter 6.4.1.21.3.13 "Menu 'CFC'" on page 2685
- \& Chapter 6.4.1.21.4.11.14 "Dialog 'Properties' - 'CFC Execution Order"' on page 2761


## CFC editor, page-oriented

POUs generated in the "Continuous Function Chart (CFC) - page-oriented" cannot be converted into "Continuous Function Chart (CFC)" POUs or back.


- (1) Page navigator
- (2) Page header with name and description
- (3) Left border area reserved for inputs and sink connection marks
- (4) Program area
- (5) Right border area reserved for outputs and source connection marks

Editing You can drag a "Page" element from the "ToolBox" view to the page navigation. Then an additional page is inserted.

You can select existing pages in the page navigation and duplicated them by clicking "Edit $\rightarrow$ Copy" and "Edit $\rightarrow$ Paste".
The size of the page is changed by means of the "Edit Page Size" command.
Connections over multiple pages are established by means of the "Connection Mark - Source" and "Connection Mark - Sink" elements. When you drag a connecting line from an input pin or an output pin to the border area, a new connection mark is created automatically. The advantage is that the "List components" input assistance provides all previously defined connection mark sources.
If you have selected an element in the editor, then you can use the arrow keys to move the selection from one element to the next to navigate through the circuit. If you then select a connection mark and press another arrow key, even the corresponding connection mark of the next/previous page will be selected.

You can transfer networks from a CFC POU to the program area of a page-oriented CFC by clicking "Edit $\rightarrow$ Copy" and "Edit $\rightarrow$ Paste" (from the clipboard). You can also use drag\&drop.

Execution order The execution order is determined automatically according to the order of the pages as they are sorted in the page navigator of the editor. Within a page, a page-oriented CFC object behaves like a CFC object. Therefore, you can switch between "Auto Data Flow Mode" and "Explicit Execution Order Mode".

## Additional com- See also

 mands in "CFC page-oriented"- ${ }^{\leftrightarrows}$ Chapter 6.4.1.21.3.13.2 "Command 'Edit Page Size'" on page 2686
- $\&$ Chapter 6.4.1.21.3.13.1 "Command 'Edit Worksheet"' on page 2685

See also

- $\rightleftarrows^{\circledR}$ Chapter 6.4.1.20.1.3 "Common functions in graphical editors" on page 2048
- ${ }^{\bullet}$ Chapter 6.4.1.9.5.3.3 "Programming in the CFC editor" on page 1871
- ${ }^{*} \Rightarrow$ Chapter 6.4.1.9.5.3.2 "Automatic Execution Order by Data Flow" on page 1867
- ${ }^{\bullet}$ Chapter 6.4.1.21.3.13 "Menu 'CFC'" on page 2685
- $\quad$ Chapter 6.4.1.21.4.11.14 "Dialog 'Properties' - 'CFC Execution Order'" on page 2761


## Keyboard Shortcuts in the CFC Editors

| Keyboard shortcuts in the CFC editor and pageoriented CFC editor | Keyboard shortcuts | Command |
| :---: | :---: | :---: |
|  | [Ctrl]+[Shift]+[A] | Select All |
|  | Insert elements: |  |
|  | [Ctrl]+[B] | Insert Box <br> The "Input Assistant" dialog opens in order to select the box. |
|  | [Ctrl]+[Shift]+[B] | Insert Empty Box |
|  | [Ctrl]+[Shift]+[E] | Insert Box with EN/ENO <br> The "Input Assistant" dialog opens in order to select the box. |
|  | [Ctrl]+[Q] | Insert Input Inserts an input element |
|  | [Ctrl]+[A] | Insert Output <br> Inserts an output element |
|  | [Ctrl]+[L] | Insert Jump |
|  | Edit already inserted elements: |  |
|  | [Ctrl]+[N] | Negate |
|  | [Ctrl]+[M] | Toggle between Set, Reset, REF, and None |
|  | [Ctrl]+[U] | Reset Pins |

After inserting an element, the inserted element is selected in the editor.
See also

- $\quad$ ² Chapter 6.4.1.20.1.7.6.5 "CFC Element 'Box'" on page 2110
- $\boldsymbol{y y}^{\bullet}$ Chapter 6.4.1.20.1.7.6.3 "CFC Element 'Input"' on page 2109
- $\underbrace{\bullet}$ Chapter 6.4.1.20.1.7.6.4 "CFC Element 'Output'" on page 2109
- $\quad$ Chapter 6.4.1.20.1.7.6.6 "CFC element 'Jump'" on page 2110
- $\sum^{2}$ Chapter 6.4.1.21.3.13.3 "Command 'Negate'" on page 2686
- $\quad{ }^{*}$ Chapter 6.4.1.21.3.13.7 "Command 'S (Set)"' on page 2687
- ※ Chapter 6.4.1.21.3.13.6 "Command 'R (Reset)'" on page 2687
- ${ }^{\bullet}>$ Chapter 6.4.1.21.3.13.8 "Command 'REF= (Reference Assignment)"' on page 2687
- 宏 Chapter 6.4.1.21.3.13.5 "Command 'None"' on page 2687
- $\xi^{*}$ Chapter 6.4.1.21.3.13.24 "Command 'Reset Pins'" on page 2694


## CFC Editor in Online Mode

In online mode, you can monitor and change variable values of the controller. In addition, debugging features are provided such as breakpoints and stepping.

Monitoring As usual, you can monitor values in the declaration part as well as in the implementation part (with inline monitoring).
Inline monitoring of a function block is possible only when an instance of the function block is open. No values are displayed in the basic implementation view.

Monitoring a Boolean variable

The connections between Boolean variables are displayed in color according to their actual value: TRUE in blue and FALSE in black. The element pins are decorated with the actual value.


[^33]

Forcing and writing of variables

In online mode in the declaration editor, you can prepare a value for forcing or writing a monitored variable.

When you select the "Prepare values in implementation part" check box in the "CFC Editor" category of the CODESYS options, you can also prepare values in the implementation part.
To do this, open the "Prepare Value" dialog by double-clicking an element or the monitoring box next to an element. No dialog appears for Boolean variables. However, with each mouse click on the value displayed next to the variable, the values TRUE and FALSE are toggled.
Prepared values are displayed in angle brackets. After executing a write or a force, a red " F " is shown in the monitoring box.

Changing of constant input parameters of function block instances

You can write input parameters of function block instances of type VAR_INPUT CONSTANT in online mode and modify the parameters in this way. After logging out, you save these parameters by clicking "Save Prepared Parameters to Project".
$\checkmark$ Requirement: A CFC editor is active. An instantiated function block has VAR_INPUT CONSTANT variables in its declaration.

1. In the editor, open the POU by calling the function block instance.
```
=>
    FUNCTION_BLOCK FB_DoIt
    VAR_INPUT
    iAlfa : INT;
    iBravo: INT;
    sCharlie : STRING := 'Charlie';
    xItem : BOOL;
    iDelta : INT;
END_VAR
VAR_INPUT CONSTANT
    MAXIMUM : INT := 12;
END_VAR
VAR_OUTPUT
    iResult : INT;
    sResult : STRING;
    xResult : BOOL;
END_VAR
```

The declaration of FB_DOIt has been supplemented by the constant MAXIMUM.


The graphical representation of the function block instances contains the "Parameters" button.
2. Login to the controller.
3. Click the "Parameters" button of the function block instance.
$\Rightarrow$ The "Edit Parameters" dialog opens.
4. Click the "Value" column in an inline monitoring field of a parameter.
$\Rightarrow$ The "Prepare Value" dialog opens.
5. Type 20 in the "Prepare a new value for the next write or force operation" field.
6. Click "OK" to confirm the entry.
$\Rightarrow$ The prepared value is shown in angle brackets next to the current value (for example, <20>).

| Parameter | Type | Value | Initial value | Min | Max | Unit | Description |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| MAXIMUM | INT | 128203 | 12 |  |  |  |  |

7. Click "Debug $\rightarrow$ Write Values".
$\Rightarrow$ The prepared value is written. The parameter is changed and displayed in the project in brackets after the value.

| Parameter | Type | Value | Initial value | Min | Max | Unit | Description |
| :---: | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| MAXIMUM | INT | $[20]$ | 20 | 12 |  |  |  |

The difference between both values is shown by a red cross next to the parameter field of the function block instance.

8. Click "Edit Parameters" to close the dialog. Logout.
9. Click "CFC $\rightarrow$ Save Prepared Parameters to Project".
$\Rightarrow$ The change parameter values are saved to the project. The asterisk next to the parameter field disappears.

Breakpoint loca- Possible position of a breakpoint
tions

- Element "Output" Variables are described.
- Element "Box"

POUs are called.

- Element "RETURN" The program flow is branched.
- Element "Selector"

Structure elements are described.
Click "Debug $\rightarrow$ Toggle Breakpoint" to set a new breakpoint or delete an existing breakpoint. A red circle in the block diagram represents an active breakpoint.


## NOTICE!

A breakpoint is set automatically in all methods that can be called.
Therefore, if a method is called that is defined over an interface, then breakpoints are set in all methods of function blocks that implement this interface. This also applies to all inherited function blocks that define methods.

Stepping into a You can process a POU in steps in debug mode. A called POU is supplemented internally by a RETURN at the beginning before the element with the number 0 and at the end after the last element. When stepping, these are started automatically.

## Commands in

 online modeSee also

- \& Chapter 6.4.1.21.3.13.35 "Command 'Force Function Block Input"' on page 2697
- \& Chapter 6.4.1.21.3.13.34 "Command 'Prepare Box for Forcing"' on page 2697
- \& Chapter 6.4.1.21.3.13.18 "Command 'Edit Parameters"' on page 2692
- \& Chapter 6.4.1.21.3.13.19 "Command 'Save Prepared Parameters to Project"' on page 2693

See also

- $\#$ Chapter 6.4.1.13.2 "Monitoring of Values" on page 1995
- « Chapter 6.4.1.12.5 "Forcing and Writing of Variables" on page 1987
- "as "Forcing a function block input in CFC" on page 1989
- \& Chapter 6.4.1.12.3 "Using Breakpoints" on page 1981
- \& Chapter 6.4.1.12.4 "Stepping Through a Program" on page 1985


## Elements

6.4.1.20.1.7.6.1 CFC element 'Page'. ..... 2109
6.4.1.20.1.7.6.2 CFC element 'Control Point' ..... 2109
6.4.1.20.1.7.6.3 CFC Element 'Input'. ..... 2109
6.4.1.20.1.7.6.4 CFC Element 'Output' ..... 2109
6.4.1.20.1.7.6.5 CFC Element 'Box'. ..... 2110
6.4.1.20.1.7.6.6 CFC element 'Jump' ..... 2110
6.4.1.20.1.7.6.7 CFC element 'Label'. ..... 2111
6.4.1.20.1.7.6.8 CFC element 'Return'. ..... 2111
6.4.1.20.1.7.6.9 CFC element 'Composer'. ..... 2111
6.4.1.20.1.7.6.10 CFC element 'Selector'. ..... 2111
6.4.1.20.1.7.6.11 CFC element 'Comment'. ..... 2111
6.4.1.20.1.7.6.12 CFC element 'Connection Mark - Source/Sink'. ..... 2112
6.4.1.20.1.7.6.13 CFC element 'Input Pin' ..... 2112
6.4.1.20.1.7.6.14 CFC element 'Output Pin'. ..... 2112

## CFC element 'Page'

Symbol:
The element inserts a new page into the editor. It is available only in the page-oriented CFC editor. The number of the page is automatically assigned in accordance with its position. You can enter the name and the description of the page into the orange header. The page size is adapted with the "Edit Page Size" command.
See also

- $\sum^{\wedge}$ Chapter 6.4.1.21.3.13.2 "Command 'Edit Page Size'" on page 2686


## CFC element 'Control Point'

## Symbol: ${ }^{\dagger}$

Use a control point in order to fix points of a connection before you adapt the line routing. To do this, drag the element to the desired position on a connecting line. Connecting lines with control points are no longer routed automatically.
See also

- ${ }^{4}$ Chapter 6.4.1.9.5.3.3 " Programming in the CFC editor" on page 1871
- $\sum^{\bullet}$ Chapter 6.4.1.21.3.13.30 "Command 'Create Control Point'" on page 2696
- $\Leftrightarrow$ Chapter 6.4.1.21.3.13.29 "Command 'Remove Control Point"' on page 2695


## CFC Element 'Input'

Symbol: $\square-$
Keyboard shortcuts for inserting the element: [Ctrl]+[Q]
By default, CODESYS inserts an input element with the text "???". You can edit this input field directly by clicking it and typing in a constant value or a variable name. Alternatively, you could click $\qquad$ to open the Input Assistant to select a variable.

## CFC Element 'Output'

Symbol: -ص
Keyboard shortcuts for inserting the element: [Ctrl]+[A]

By default, CODESYS inserts an output element with the text "???". You can edit this input field directly by clicking it and typing in a constant value or a variable name. Alternatively, you could click ... to open the Input Assistant to select a variable.

## CFC Element 'Box'

Symbol:
Keyboard shortcuts for inserting the element

- [Ctrl]+[B]
- $[$ Ctrl $]+[$ Shift $]+[B]$ : Empty box
- [Ctrl]+[Shift]+[E]: Box with EN/ENO

You use the element in order to insert an operator, a function, a function block, or a program. By default, CODESYS inserts the element with the name "???". You can edit this field directly by clicking it and typing in a function block name. Alternatively, you could click ... to open the Input Assistant and select a function block
In the case of a function block, CODESYS also displays an input field ("???") above the function block symbol. You have to replace this name with the name of the function block instance. If you instantiate a function block with constant input parameters, then the function block element displays the "Parameter..." field in the bottom left corner. You click on this field to edit the parameters.

In order to replace an existing box, you replace only the currently inserted identifier with the new desired name. When you do this, note that CODESYS adapts the number of input and output pins according to the definition of the POU and that existing assignments may be deleted as a result.

## NOTICE!

Because feedback is allowed in CFC, implicit variables with the data type of the input variable are created at the output of a box (in the example: temp_USINT). If the result of the operation of a function block is a value which exceeds the number range of the data type of the input variable, then the overflow is written to the implicit variable. The actual output variable gets the value of the implicit variable, thus the overflow and not the actual result of the operation (see example).

## Example Implicitly generated variables at the box output:



Implicitly generated code:

```
temp_USINT := USINT1 * temp_USINT;
UDINT1 := temp_USINT;
```


## See also

- ${ }^{\circ}$, Chapter 6.4.1.21.3.13.18 "Command 'Edit Parameters"" on page 2692


## CFC element 'Jump'

Symbol: ص

You use the element in order to define a position at which program execution is to continue. You must define this target position by a label. To do this, enter the name of the mark in the input field "???". If you have already inserted the corresponding label, you can also select it via the input assistant (...).

See also

- ${ }^{4}$ Chapter 6.4.1.20.1.7.6.7 "CFC element 'Label"" on page 2111


## CFC element 'Label'

Symbol:
A label defines a position to which program execution jumps with the help of a jump element. In online mode CODESYS automatically inserts a RETURN flag at the end of a CFC function block.

See also

- ② Chapter 6.4.1.20.1.7.6.6 "CFC element 'Jump'" on page 2110


## CFC element 'Return'

Symbol: $=$
Use the element in order to exit the function block.
Please note that in online mode in the CFC editor a return element is automatically inserted before the first line and after the last element. In single-step execution CODESYS automatically jumps to the return element at the end before exiting the function block.

## CFC element 'Composer'

Symbol: 非
The composer element is for handling structural components. The individual components of a structure are made available to you as an input. For this purpose you must name the composer element like the structure concerned (replace the "???").

The composer element is the counterpart to the selector element.
See also

- ${ }^{*}$ Chapter 6.4.1.20.1.7.6.10 "CFC element 'Selector"' on page 2111


## CFC element 'Selector'

Symbol: 퍄
The selector element is for handling structural components. The individual components of a structure are made available to you as an output. For this purpose you must name the selector element like the structure concerned (replace the "???")
The selector element is the counterpart to the composer element.
See also

- ${ }^{4}$ Chapter 6.4.1.20.1.7.6.9 "CFC element 'Composer"" on page 2111


## CFC element 'Comment'

Symbol: ■

With this element you input a comment in the CFC editor. Replace the placeholder text in the element by the comment text. A line break can be inserted with the aid of the shortcut [Ctrl] + [Enter].

## CFC element 'Connection Mark - Source/Sink'

Symbol: = $=$
You can use connection marks instead of a connecting line between elements. That helps you to display complex diagrams more clearly.
For a valid connection you must connect an element "Connection Mark - Source" with the output of an element and an element "Connection Mark - Sink" with the input of another element. Both marks must bear the same name. The names are not case-sensitive.

Notes on naming

- The standard name for connection marks is $\mathrm{C}-\langle\mathrm{nr}\rangle .\langle\mathrm{nr}\rangle$ is a sequential number starting with 1.
- You can rename the standard name. In doing so, you must make sure that the connection mark - source and connection mark - sink have the same name.
- If you change the name of the connection mark - source, the destination name is automatically renamed.
- If you change the name of the connection mark - sink, the source name is retained.

Observe the command "Connection Mark" for the automatic transformation of an existing connection.

See also

- ${ }^{\wedge}$ Chapter 6.4.1.21.3.13.31 "Command 'Connection Mark"' on page 2696
- $\Leftrightarrow$ Chapter 6.4.1.20.1.7.2 "CFC Editor" on page 2098


## CFC element 'Input Pin'

Symbol:
Depending on the type of function block you can add further inputs to an inserted function block element. To do this you must select the function block element and drag the function block input element onto the body of the function block.
Please note: You can drag an input or output connection to another position on the function block with the [Ctrl] key pressed.
See also

- ${ }^{4}$, Chapter 6.4.1.20.1.7.6.14 "CFC element 'Output Pin"' on page 2112


## CFC element 'Output Pin'

## Symbol:

Depending on the type of function block you can add further outputs to an inserted function block element. To do this you must select the function block element and drag the function block output element onto the body of the function block.
Please note: You can drag an input or output connection to another position on the function block with the [Ctrl] key pressed.

See also

- ${ }^{\circ}$ Chapter 6.4.1.20.1.7.6.13 "CFC element 'Input Pin"" on page 2112
6.4.1.20.2.1 General ..... 2113
6.4.1.20.2.2 Local variables - VAR ..... 2113
6.4.1.20.2.3 Input variables - VAR_INPUT ..... 2113
6.4.1.20.2.4 Output variables - VAR_OUTPUT ..... 2114
6.4.1.20.2.5 Input/Output Variable (VAR_IN_OUT) ..... 2114
6.4.1.20.2.6 Global variables - VAR_GLOBAL ..... 2118
6.4.1.20.2.7 Temporary variable - VAR_TEMP ..... 2119
6.4.1.20.2.8 Static variables - VAR STAT ..... 2119
6.4.1.20.2.9 External variables - VAR_EXTERNAL ..... 2120
6.4.1.20.2.10 Instance variables - VAR_INST. ..... 2120
6.4.1.20.2.11 Configuration variables - VAR_CONFIG ..... 2121
6.4.1.20.2.12 Constant Variables - 'CONSTANT' ..... 2121
6.4.1.20.2.13 Persistent Variable - PERSISTENT ..... 2122
6.4.1.20.2.14 Retain Variable - RETAIN. ..... 2124
6.4.1.20.2.15 SUPER ..... 2125
6.4.1.20.2.16 THIS ..... 2126


## General

The scope of a variable defines how and where you can use a variable. You define the scope in the variable declaration.

## Local variables - VAR

Local variables are declared between the keywords VAR and END_VAR in the declaration part of programming objects.
You have read-only access to local variables by using the instance path.
You can extend local variables with an attribute keyword.

## Example

```
VAR
    iVar1 : INT;
END_VAR
```

See also

- ※ Chapter 6.4.1.9.19 "Data Persistence" on page 1920


## Input variables - VAR_INPUT

Input variables are used at the inputs of function blocks.
VAR_INPUT variables are declared between the keywords VAR_INPUT and END_VAR in the declaration part of programming objects.
You can extend input variables with an attribute keyword.

## Example

```
VAR_INPUT
    iIn1 : INT; (* 1st input variable *)
END_VAR
```

See also

- ${ }^{*} \boldsymbol{y}$ Chapter 6.4.1.9.19 "Data Persistence" on page 1920


## Output variables - VAR_OUTPUT

Output variables are used at the outputs of function blocks.
VAR OUTPUT variables are declared between the keywords VAR_OUTPUT and END VAR in the declaration part of programming objects. CODESYS returns the values of this variable to the calling POU. There you can retrieve the values and continue using them.
You can extend output variables with an attribute keyword.

## Example

```
VAR_OUPUT
    iOut1 : INT; (*1st output variable *)
END VAR
```


## See also

- $\boldsymbol{y}^{2}$ Chapter 6.4.1.9.19 "Data Persistence" on page 1920

Output variables According to the IEC 61131-3 standard, functions and methods have additional outputs. You in functions and have to assign these additional outputs when calling the function, as shown below. methods

## Example

```
fun(iIn1 := 1, iIn2 := 2, iOut1 => iLoc1, iOut2 => iLoc2);
```


## Input/Output Variable (VAR_IN_OUT)

A VAR_IN_OUT variable is an input/output variable, which is part of a POU interface and serves as a formal pass-by-reference parameter.

## Syntax declaration

```
<keyword> <POU name>
VAR_IN_OUT
    <variable name> : <data type> ( := <initialization value> )? ;
END VAR
<keywword> : FUNCTION | FUNCTION_BLOCK | METHOD | PRG
```

You can declare an input/output variable in the VAR_IN_OUT declaration section in the POUs PRG, FUNCTION_BLOCK, METHOD, or FUNCTION. As an option, a constant of the declared data type can be assigned as an initialization value. The VAR_IN_OUT variable can be read and written.

## Usage

- Call: When a POU is called, the formal VAR_IN_OUT variable receives the actual variable (pass-by-reference variable) as the argument. At runtime, no copies are generated when parameters are passed. Instead, the formal variable receives a reference to the actual variable passed remotely. The referential variables contain a memory address internally as a value to the actual value (pass as pointer, call-by reference). It is not possible to specify a constant (literal) or a bit variable directly as an argument.
- Read/write access within the POU: If the variable is written to within the POU, then this affects the passed variable. When the POU is exited, any performed changes are retained. This means that a POU uses its VAR_IN_OUT variables just like the calling POU uses its variables. Read access is always permitted.
- Read/write access remotely: VAR_IN_OUT variables cannot be directly read or written remotely via <function block instance name>.<variable name>. This works only for VAR_INPUT and VAR_OUTPUT variables.
- Passing string variables: If a string variable is passed as an argument, then the actual variable and the formal variable should have the same length. Otherwise the passed string can be manipulated unintentionally. This problem does not occur in the case of VAR_OUTPUT CONSTANT parameters.
- Passing bit variables: A bit variable cannot be passed directly to a VAR_IN_OUT variable because it needs an intermediate variable.
- Passing properties: Not permitted.


If a string is passed as a variable or a constant to a formal VAR_IN_OUT CONSTANT variable, then the string is automatically passed cō̄plētely. You do not have to check the string length.

See also

- Chapter " "Transfer variable VAR_IN_OUT CONSTANT" on page 2117

```
Example
Passing arrays
```

```
TYPE DUT A :
```

TYPE DUT A :
STRUCT
STRUCT
xA: BOOL;
xA: BOOL;
iB: INT;
iB: INT;
END_STRUCT
END_STRUCT
END_TYPE
END_TYPE
FUNCTION_BLOCK FB_SetArray
FUNCTION_BLOCK FB_SetArray
VAR_IN_OUT
VAR_IN_OUT
aDāta_A : ARRAY[0..1] OF DUT_A; // Formal variable
aDāta_A : ARRAY[0..1] OF DUT_A; // Formal variable
END_VAR
END_VAR
aData_A[0].xA := TRUE;
aData_A[0].xA := TRUE;
aData_A[0].iB := 100;
aData_A[0].iB := 100;
PROGRAM PLC_PRG
PROGRAM PLC_PRG
VAR
VAR
fbSetA : FB_SetArray;
fbSetA : FB_SetArray;
aSpecialData : ARRAY[0..1] OF DUT_A; // Actual variable
aSpecialData : ARRAY[0..1] OF DUT_A; // Actual variable
END_VAR
END_VAR
fbSetA(aData_A := aSpecialData);
fbSetA(aData_A := aSpecialData);

| Expression | Type | Value | Prepared value | Address | Comment |
| :---: | :---: | :---: | :---: | :---: | :---: |
| $\pm$ fbSetA | FB_SetArray |  |  |  |  |
| - aSpecialData | ARRAY [0..1] OF DUT_A |  |  |  | Actual variable |
| - aSpecialData[0] | DUT_A |  |  |  |  |
| - XA | BOOL | TRUE |  |  |  |
| - is | INT | 100 |  |  |  |
| - aSpecialData[1] | DUT_A |  |  |  |  |
| - XA | BOOL | FALSE |  |  |  |
| - is | INT | 0 |  |  |  |

Passing
strings
{attribute 'qualified_only'}
VAR_GLOBAL
g_sDEV_STATUS : STRING(25) := 'Device_A';
END_V\overline{A}R
FUNCTION_BLOCK FB_SetStatus
VAR_IN_OÜT
sDeviceStatus : STRING(25); // Formal parameter
END_VAR
sDeV̄iceStatus := CONCAT(sDeviceStatus, 'Activ');
PROGRAM PLC_PRG
VAR
fbDoB : FB_SetStatus;
END_VAR
fbD\overline{OB(sDeviceStatus := GVL.g_sDEV_STATUS); //Call with actual}
parameter
The variable sDeviceStatus is part of the POU interface of FB_B. When calling fbDoB, first a device name is assigned to the string and then the string is manipulated.

```

\section*{Passing bit variables}
```

VAR_GLOBAL

```
VAR_GLOBAL
    xBit0 AT %MX0.1 : BOOL;
    xBit0 AT %MX0.1 : BOOL;
    xTemp : BOOL;
    xTemp : BOOL;
END_VAR
END_VAR
FUNCTION_BLOCK FB_DoSomething
FUNCTION_BLOCK FB_DoSomething
VAR_INPUT
```

VAR_INPUT

```
```

    xIn : BOOL;
    END_VAR
VAR_IN_OUT
xInOut : BOOL;
END VAR
IF xIn THEN
xInOut := TRUE;
END_IF
PROGRAM PLC_PRG
VAR
xIn : BOOL;
DoSomething_1 : FB_DoSomething;
DoSomething_2 : FB_DoSomething;
END VAR
// The following line of code causes a compiler error:
// C0201: Typ 'BIT' is not equal to type 'BOOL' of VAR_IN_OUT
'xInOut'
DoSomething_1(xIn := xIn, xInOut := xBit0);
// Workaround
xTemp := xBit0;
DoSomething_2(xIn := xIn, xInOut := xTemp);
xBit0 := xTemp;

```

The program calls the function block instances DoSomething_1 and DoSomething_2. As a result of the direct assignment of the bit variable xBito to the VAR_IN_OUT input, \(\bar{a}\) compiler error is generated when the DoSomething_1 instance is called. In contrast, calling the DoSomething_2 instance with the assignment of an intermediate variable is correct code.

Transfer vari- A VAR_IN_OUT CONSTANT variable serves as a constant pass-by-reference parameter, to
able
VAR IN OUT CONSTANT

Syntax declaration which \(\overline{\mathrm{a}}\) STRING or WSTRING type variable or constant (literal) can be passed. The parameter can be read, but not written. Passing of properties is not permitted.
```

<keyword> <POU name>
VAR_IN OUT CONSTANT
<vāriable name> : <data type>; // formal parameter
END_VAR
<keyword> : FUNCTION | FUNCTION_BLOCK | METHOD | PRG

```

VAR IN OUT CONSTANT variables are declared without assigning an initialization value.

\section*{Usage}
- When calling the POU, a STRING or WSTRING constant variable or literal can be passed. Consequently, write access is not permitted.
- Passing parameters of a string constant: The string length of the constants can be any size, and the string length does not depend on the string length of the VAR_IN_OUT CONSTANT variables.

If the "Replace constants" option is selected in "Project \(\rightarrow\) Project Settings" in the "Compile Options" category, then passing the parameters of a constant with basic data type or a constant variable with basic data type generates a compiler error.

The variable is supported in compiler version >=3.5.2.0.

```

Example
Passing
parameters of
string con-
stants and
string varia-
bles

```
```

FUNCTION funManipulate : BOOL

```
FUNCTION funManipulate : BOOL
VAR_IN_OUT
VAR_IN_OUT
        sReadWrite : STRING(16); (* Can be read or written here in POU
        sReadWrite : STRING(16); (* Can be read or written here in POU
*)
*)
        dwVarReadWrite : DWORD; (* Can be read or written here in POU *)
        dwVarReadWrite : DWORD; (* Can be read or written here in POU *)
END VAR
END VAR
VAR - IN OUT CONSTANT
VAR - IN OUT CONSTANT
    c_sReadOnly : STRING(16); (* Constant string variable can only
    c_sReadOnly : STRING(16); (* Constant string variable can only
be read here in POU *)
be read here in POU *)
END_VAR
END_VAR
sReadWrite := 'String_from_POU';
sReadWrite := 'String_from_POU';
dwVarReadWrite := STRINNG_TO_DWORD(c_sReadOnly);
dwVarReadWrite := STRINNG_TO_DWORD(c_sReadOnly);
PROGRAM PRG_A
PROGRAM PRG_A
VAR
VAR
    sVarFits : STRING(16);
    sVarFits : STRING(16);
    sValFits : STRING(16) := '1234567890123456';
    sValFits : STRING(16) := '1234567890123456';
        dwVar: DWORD;
        dwVar: DWORD;
END_VAR
END_VAR
// The following line of code causes the compiler error c0417:
// The following line of code causes the compiler error c0417:
// C0417: VAR_IN_OUT parameter 'sReadWrite' needs a variable with
// C0417: VAR_IN_OUT parameter 'sReadWrite' needs a variable with
write access as input.
write access as input.
funManipulate(sReadWrite:='1234567890123456',
funManipulate(sReadWrite:='1234567890123456',
c_sReadOnly:='1234567890123456', dwVarReadWrite := dwVar);
c_sReadOnly:='1234567890123456', dwVarReadWrite := dwVar);
// Correct code
// Correct code
funManipulate(sReadWrite := sValFits, c_sReadOnly := '23',
funManipulate(sReadWrite := sValFits, c_sReadOnly := '23',
dwVarReadWrite := dwVar);
dwVarReadWrite := dwVar);
funManipulate(sReadWrite := sVarFits, c_sReadOnly := sValFits,
funManipulate(sReadWrite := sVarFits, c_sReadOnly := sValFits,
dwVarReadWrite := dwVar);
dwVarReadWrite := dwVar);
In the code, strings are passed to the funManipulate function via different VAR_IN_OUT variables. When passing a string literal, a compiler error is output to a VAR_IN_OUT variable. When passing a constant variable to a VAR_IN_OUT CONSTANT variable, correct code is generated even for passing string variables.
```

See also

- $\xi^{\xi}$ Chapter 6.4.1.9.4 "Declaration of Variables" on page 1847
- $\Longleftrightarrow$ Chapter 6.4.1.21.4.12.4 "Dialog Box 'Project Settings' - 'Compileoptions'" on page 2769
- $y^{*}$ Chapter 6.4.1.21.2.21.4 "Object 'Function'" on page 2482
- $\Leftrightarrow$ Chapter 6.4.1.21.2.21.3 "Object 'Function Block'" on page 2479
- $\Leftrightarrow$ Chapter 6.4.1.21.2.21.6 "Object 'Method'" on page 2485
- ${ }^{*} \Rightarrow$ Chapter 6.4.1.21.2.21.5 "Object 'Interface"' on page 2484
- « Chapter 6.4.1.21.2.21.7 "Object 'Interface Method'" on page 2490
- $y_{y}$ Chapter 6.4.1.20.2.12 "Constant Variables - 'CONSTANT"' on page 2121


## Global variables - VAR_GLOBAL

Global variables are ordinary variables, constants, external or remanent variables that are recognized within the entire project.
You declare global variables in global variable lists or in the declaration section of programming objects between the keywords VAR_GLOBAL and END_VAR.

The system recognizes a global variable when you prepend the variable name with a dot (for example, .iGlobVar1).

## NOTICE!

If a local variable that is declared in a block has the same name as a global variable, then it has precedence within the block.

## NOTICE!

For compiler version 3.2.0.0 and later, CODESYS always initializes global variables before the local POU variables.

## Example

```
VAR_GLOBAL
    iVarGlob1 : INT;
END_VAR
```


## See also

- $\Rightarrow$ Chapter 6.4.1.21.2.12 "Object 'GVL' - Global Variable List" on page 2465
- $\&$ Chapter 6.4.1.20.3.70 "Operator - Global namespace" on page 2216


## Temporary variable - VAR_TEMP

This function is an extension of the IEC 61131-3 standard.
You declare temporary variables locally between the keywords VAR_TEMP and END_VAR.
VAR_TEMP declarations are possible only in program blocks and function blocks.
CODESYS initializes temporary variables each time the block is called.
The application can access the temporary variables only in the implementation section of a program block or a function block.

## Example

```
VAR_TEMP
    iVarTmp1 : INT; (*1st temporary variable *)
END_VAR
```


## Static variables - VAR_STAT

This function is an extension of the IEC 61131-3 standard.
You declare static variables locally between the keywords VAR_STAT and END_VAR. CODESYS initializes static variables the first time each block is called.

You can access static variables only from within the namespace where the variables are declared (like static variables in C). But static variables retain their values when the application leaves the block. For example, you can use static variables as counters for function calls.

You can extend static variables with an attribute keyword.

## Example

```
VAR_STAT
    iVarStat1 : INT;
END_VAR
```

See also

- ${ }^{\text {© }}$ Chapter 6.4.1.9.19 "Data Persistence" on page 1920


## External variables - VAR_EXTERNAL

External variables are global variables that are imported into a block.
You declare these variables between the keywords VAR EXTERNAL and END_VAR. If the global variable does not exist, then an error message is printed.

## NOTICE!

CODESYS does not require you to declare a global variable as external in order to use it in a POU. The keyword exists only for maintaining compliance with IEC 61131-3.

## Syntax

```
<POU keyword> <POU name>
VAR_EXTERNAL
        <variable name> : <data type>;
END_VAR
```

Initialization is not permitted.

## Example

```
FUNCTION_BLOCK FB_DoSomething
VAR_EXTE\overline{RNAL}
    iVarExt1 : INT; (* 1st external variable *)
END_VAR
```

See also

- ${ }^{4}$ Chapter 6.4.1.21.2.12 "Object 'GVL' - Global Variable List" on page 2465


## Instance variables - VAR_INST

CODESYS does not save a VAR_INST method variable in a method stack, but in the stack of the function block instance. This means that the VAR_INST variable functions like other Variables of the function block instance, and it is not reinitialized each time the method is called.
VAR_INST variables are permitted in methods only and you can access these variables only within the method. The variable values of instance variables are monitored in the declaration section of the method.
You can extend instance variable with an attribute keyword.

```
Example
```

```
METHOD meth_last : INT
```

METHOD meth_last : INT
VAR INPUT
VAR INPUT
iV̄ar : INT;
iV̄ar : INT;
END_VAR
END_VAR
VAR ' INST
VAR ' INST
i\overline{Last : INT := 0;}
i\overline{Last : INT := 0;}
END_VAR
END_VAR
met\overline{h}}\mathrm{ last := iLast;
met\overline{h}}\mathrm{ last := iLast;
iLast := iVar;

```
iLast := iVar;
```


## Configuration variables - VAR_CONFIG

Use configuration variables for assigning complete addresses to variables that are declared in function blocks with incomplete addresses and will be mapped on device I/Os.

Declare the variables in a global variables list between VAR_CONFIG and END_VAR. The global variables list is termed "variables configuration", where you type the configuration variables with a complete instance path and the correct address.

Example Declaration of the variable $x \operatorname{Loc}$ In with incomplete address $\% I$ * in a function block:

```
FUNCTION_BLOCK locio
VAR
        xLocIn AT %I* : BOOL := TRUE;
END_VAR
```

The locio function block is used in the PLC_PRG program:

```
PROGRAM PLC_PRG
VAR
        locioVar1 : locio;
END_VAR
```

The correct variables configuration in the global variable list is as follows:

```
VAR_CONFIG
    PLC_PRG.locioVar1.xLocIn AT %IX1.0 : BOOL;
END_VAR
```


## See also

- ${ }^{\mu}$ Chapter 6.4.1.9.12.2 "Variables configuration - VAR_CONFIG" on page 1901


## Constant Variables - 'CONSTANT'

Constant variables are declared in global variable lists or in the declaration part of programming objects. In implementations, constant variables can be accessed as read-only via the instance path.

```
Syntax <scope> CONSTANT
    <identifier> : <data type> := <initial value> ;
END_VAR
<scope> : VAR | VAR_INPUT | VAR_STAT | VAR_GLOBAL
<data type>: <elementary data type> | <user defined data type> |
<function block>
<initial value> : <literal value> | <identifier> | <expression>
```

Always assign an initialization value when declaring a constant variable. Then the constant cannot be written any more.

## Example

```
Declaration VAR CONSTANT
    c_rTAXFACTOR : REAL := 1.19;
    END_VA}
Call rPrice := rValue * c_rTAXFACTOR;
```

You have read-only access to constant variables in an implementation. Constant variables are located to the right of the assignment operator.

See also

- ${ }^{\circ}$ ) Chapter 6.4.1.20.2.5 "Input/Output Variable (VAR_IN_OUT)" on page 2114
- © "Constants and literals" on page 2219


## Persistent Variable - PERSISTENT

Persistent variables are declared in the declaration section VAR_GLOBAL RETAIN PERSISTENT in the persistent global variable list. For variables that are marked with the PERSISTENT keyword outside of the persistence editor, instance paths are added there.

As of CODESYS version 3.3.0.1, a variable declaration with PERSISTENT RETAIN has the same effect as with RETAIN PERSISTENT or PERSISTENT.

Syntax of the declaration in the global persistent variable list
PersistentVar s:

Syntax of the declaration in POUs

VAR_GLOBAL PERSISTENT RETAIN
<identifier> : <data type> (:= <initialization>)?; <instance path to POU variable>
END_VAR
<scope> PERSISTENT RETAIN
<identifier> : <data type> ( := <initialization> )?; //
( ... )? : Optional
END_VAR
<scōpe> : VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | VAR_STAT |
VAR_GLOBAL

An assignment of inputs, outputs, or memory addresses with the AT keyword is not permitted.

Never use the POINTER TO data type in persistent variable lists. If the application is downloaded again, their addresses could change. The corresponding compiler warnings are shown in the message window.

If you frequently change the names or data types of remanent variables, then it is better to declare them as retain variables with the RETAIN keyword only.

NOTICE!
Avoid inserting instance paths because in this case twice as much memory is used and a higher cycle time can occur. Instead, declare variables in the list of persistent variables.

```
Example
Declaration in
the persistent
variable list
PersistentVa
Declaration in
the function
block FB_A:
Declaration in
the program
PLC_PRG:
```

```
rs: // Generated instance path of persistent variable
```

rs: // Generated instance path of persistent variable
PLC_PRG.fb_A.iPersistentCounter_A: INT;
PLC_PRG.fb_A.iPersistentCounter_A: INT;
END_VAR
END_VAR
FUNCTION_BLOCK FB_A
FUNCTION_BLOCK FB_A
VAR_INPUT
VAR_INPUT
END VAR
END VAR
VAR_OUTPUT
VAR_OUTPUT
END VAR
END VAR
VAR - PERSISTENT
VAR - PERSISTENT
iPersistentCounter_A : INT;
iPersistentCounter_A : INT;
END_VAR
END_VAR

```
{attribute 'qualified only'}
```

{attribute 'qualified only'}
VAR_GLOBAL PERSISTENT RETAIN
VAR_GLOBAL PERSISTENT RETAIN
g_iCounter : INT;
g_iCounter : INT;
VAR
VAR
END VA\overline{R}

```
END VA\overline{R}
```


## Possible decla-

ration locations

| Directly in the persistent global <br> variable list | The variable is persistent and lies in the protected memory area. |
| :--- | :--- |
| Locally in a program with an <br> instance path in the persistent <br> variable list <br> Locally in a function block with <br> an instance path in the persis- <br> tent variable list | The variable is persistent and located in the protected memory area and in the <br> memory (double allocation). |
| Only locally in a program <br> Only locally in a function block | This variable is not persistent. A warning is shown in the message window. <br> Hint: Click "Declarations $\rightarrow$ Add All Instance Paths" to import the variables into <br> the persistent variable list. |
| Locally in a function | This declaration does not have any effect. The variable is not persistent. |

## In the persistence editor, click "Declarations $\rightarrow$ Add All Instance Paths" if local variables are marked with PERSISTENT.

Whenever possible, avoid marking variables, which are declared in a function block, with PERSISTENT. This is because the function block instance is stored entirely in remanent memory and not just the marked variable.

See also

- $\sum^{*}$ Chapter 6.4.1.9.19 "Data Persistence" on page 1920
- ${ }^{4}$ Chapter 6.4.1.21.3.18.4 "Command 'Add all instance paths'" on page 2720


## Retain Variable - RETAIN

Retain variables are declared by the keyword RETAIN is added in programming objects in the scope VAR, VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT, VAR_STAT, or VAR_GLOBAL.

## Syntax for the declaration

```
<scope> RETAIN
    <identifier>: <data type> ( := <initialization> )? // ( ... )? :
Optional
END_VAR
<scO्Oe> : VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | VAR_STAT |
VAR_GLOBAL
```

An assignment of inputs, outputs, or memory addresses with the AT keyword is not permitted.

## Example

In a POU:

```
        VAR RETAIN
            iVarRetain: INT;
        END_VAR
```

In a GVL:

```
        VAR_GLOBAL RETAIN
            g_iVarRetain: INT;
        END_VA\overline{R}
```

Possible decla-
ration locations

| Locally in a program | Only the variable is located in the retain memory area. <br> Info: When using redundancy, the entire program with all of its data is located in <br> the retain memory area. |
| :--- | :--- |
| Globally in a global variable list | Only the variable is located in the retain memory area. <br> Info: When using redundancy, the entire global variable list with all of its data is <br> located in the retain memory area. |
| Locally in a function block | The entire instance of the function block with all of its data is located in the retain <br> memory area. Only the declared retain variable is protected. |


| Locally in a function | The variable is not located in the retain memory area. This declaration does not <br> have any effect. |
| :--- | :--- |
| Locally and persistently in a <br> function | The variable is not located in the retain memory area. This declaration does not <br> have any effect. |

Whenever possible, avoid using RETAIN to mark the variables of a function block.

## See also

- 若 Chapter 6.4.1.9.19 "Data Persistence" on page 1920
- \& Chapter 6.4.1.21.3.18.4 "Command 'Add all instance paths'" on page 2720


## SUPER

SUPER is a special variable and is used for object-oriented programming.
SUPER is the pointer of a function block to the basic function block instance from which the function block was generated. The SUPER pointer thus also permits access to the implementation of the methods of the basic function block (basic class). A SUPER pointer is automatically available for each function block.

You can use SUPER only in methods and in the associated function block implementations.
Dereferencing of the pointer: SUPER^
Use of the SUPER pointer: with the help of the keyword SUPER you call a method that is valid in the instance of the basic class or parent class.

| Examples | ST: |
| :--- | :--- |
|  | SUPER^. METH_DoIt () ; $^{\wedge}$ |
|  | FBD/CFC/LD |



THIS is not yet implemented for the instruction list (IL).

```
Examples Use of SUPER and THIS pointers
```

```
FUNCTION_BLOCK FB_Base
```

FUNCTION_BLOCK FB_Base
VAR OUTPUT
VAR OUTPUT
i\overline{Cnt : INT;}
i\overline{Cnt : INT;}
END_VAR
END_VAR
METHOD METH_DOIt : BOOL
METHOD METH_DOIt : BOOL
iCnt := -1;
iCnt := -1;
METHOD METH_DOAlso : BOOL
METHOD METH_DOAlso : BOOL
METH_DoAlso := TRUE;
METH_DoAlso := TRUE;
FUNCTION_BLOCK FB_1 EXTENDS FB_Base
FUNCTION_BLOCK FB_1 EXTENDS FB_Base
VAR_OUTPUTT
VAR_OUTPUTT
iBase : INT;
iBase : INT;
END_VAR
END_VAR
THIS^.METH_DoIt(); //Call of the methods of FB_1
THIS^.METH_DoIt(); //Call of the methods of FB_1
THIS^.METH_DoAlso();
THIS^.METH_DoAlso();
SUPER^.METH_DoIt(); //Call of the methods of FB_Base
SUPER^.METH_DoIt(); //Call of the methods of FB_Base
SUPER^.METH_DOAlso();
SUPER^.METH_DOAlso();
iBase := SUPER^.iCnt;
iBase := SUPER^.iCnt;
METHOD METH_DOIt : BOOL
METHOD METH_DOIt : BOOL
iCnt := 11\overline{11;}
iCnt := 11\overline{11;}
METH_DoIt := TRUE;
METH_DoIt := TRUE;
PROGRAM PLC_PRG
PROGRAM PLC_PRG
VAR
VAR
myBase : FB_Base;
myBase : FB_Base;
myFB_1 : FB_1;
myFB_1 : FB_1;
iTHIS : INT;
iTHIS : INT;
iBase : INT;
iBase : INT;
END_VAR
END_VAR
myBase();
myBase();
iBase := myBase.iCnt;
iBase := myBase.iCnt;
myFB_1();
myFB_1();
iTHI\overline{S}}:= myFB_1.iCnt

```
iTHI\overline{S}}:= myFB_1.iCnt
```


## See also

- ${ }^{4}$ Chapter 6.4.1.20.5 "Data Types" on page 2234
- ${ }^{4}$ Chapter 6.4.1.20.2.16 "THIS" on page 2126


## THIS

THIS is a special variable and is used for object-oriented programming.
THIS is the pointer of a function block to its own function block instance. A THIS pointer is automatically available for each function block.
You can use THIS only in methods and in function blocks. THIS is available for the implementation in the input assistant in the category "Keywords".
Dereferencing of the pointer: THIS^

Use of the THIS pointer

- If a local variable obscures a function block variable in a method, you can set the function block variable with the THIS pointer. See example below (1)
- If the pointer to the function block's own function block instance is referenced for use in a function. (See example below (2))


## Examples ST :

THIS^.METH_DoIt();
FBD/CFC/LD:

## THIS^

METH_DoAlso
METH_DoAlso

THIS is not yet implemented for the instruction list (IL).

Examples (1) The local variable iVarB obscures the function block variable iVarB.

```
FUNCTION_BLOCK fbA
VAR INPU\overline{T}
    iVarA: INT;
END_VAR
iVa\overline{rA := 1;}
FUNCTION_BLOCK fbB EXTENDS fbA
VAR INPU\overline{T}
    iVărB: INT := 0;
END_VAR
iVa\overline{rA := 11;}
iVarB := 2;
METHOD DoIt : BOOL
VAR_INPUT
END_VAR
VAR
    iVarB: INT;
END_VAR
iVa\overline{rB := 22; // The local variable iVarB is set.}
THIS^.iVarB := 222; // The function block variable iVarB is set
even though iVarB is obscured.
PROGRAM PLC_PRG
VAR
    MyfbB: fbB;
END_VAR
MyfbB(iVarA:=0, iVarB:= 0);
MyfbB.DoIt();
```

(2) A function call requires the reference to its own instance.

```
FUNCTION funA
VAR_INPUT
    pFB: fbA;
END_VAR
...;
FUNCTION BLOCK fbA
VAR_INPUT
    iVarA: INT;
END_VAR
...;
FUNCTION BLOCK fbB EXTENDS fbA
VAR_INPUT
    iVarB: INT := 0;
END_VAR
iVa\overline{rA := 11;}
iVarB := 2;
METHOD DoIt : BOOL
VAR_INPUT
END_VAR
VAR
    iVarB: INT;
END_VAR
iVa\overline{r}B := 22; //The local variable iVarB is set.
funA(pFB := THIS^); //funA is called via THIS^.
PROGRAM PLC_PRG
```

```
VAR
MyfbB: fbB;
END VAR
Myf\overline{b}B(iVarA:=0 , iVarB:= 0);
MyfbB.DoIt();
```


## See also

- " ${ }^{\circ}$ Chapter 6.4.1.20.5.13 "Pointers" on page 2243
- E Chapter 6.4.1.20.2.15 "SUPER" on page 2125


### 6.4.1.20.3 Operators

## General

CODESYS V3 supports all IEC-61131-3 operators. These operators are recognized implicitly throughout the project. In addition to these IEC operators, CODESYS also supports some non-IEC 61131-3 operators.

Operators are used in blocks, such as functions.

For information about the processing order (binding strength) of the ST operators, please refer to the section on ST expressions.

## CAUTION!

For operations with floating-point data types, the computational result depends on the applied target system hardware.

## CAUTION!

For operations with overflow or underflow in the data type, the computational result depends on the applied target system hardware.

Overflow/underflow in the data type

The CODESYS compiler generates code for the target device and computes temporary results always with the native size that is defined by the target device. For example, computation is performed at least with 32-bit temporary values on x86 and ARM systems and always with 64 -bit temporary values on x64 systems. This provides considerable advantages in the computation speed and often also produces the desired result. But this also means that an overflow or underflow in the data type is not truncated in some cases.

## Examples Example 1

The result of this addition is not truncated and the result in dwVar is 65536 .

```
VAR
wVar : WORD;
dwVar: DWORD;
END_VAR
wVar := 65535;
dwVar := wVar + 1;
```


## Example 2

The overflow and underflow in the data type is not truncated and the results (bVar1, bVar2) of both comparisons are FALSE on 32-bit and 64-bit hardware.

```
VAR
wVar1 : WORD;
wVar2 : WORD;
bVar1 : BOOL;
bVar2 : BOOL;
END_VAR
wVar1 := 65535;
wVar2 := 0;
bVar1 := (wVar1 + 1) = wVar2;
bVar2 := (wVar2 - 1) = wVar1;
```


## Example 3

By the assignment to wVar3, the value is truncated to the target data type wORD and the result bvar1 is TRUE.

```
VAR
wVar1 : WORD;
wVar2 : WORD;
wVar3 : WORD;
bVar1 : BOOL;
END_VAR
wVar1 := 65535;
wVar2 := 0;
wVar3 := (wVar1 + 1);
bVar1 := wVar3 = wVar2;
```


## Example 4

In order to force the compiler to truncate the temporary results, a conversion can be inserted.
The type conversion makes sure that both comparisons are 16 -bit only and the results (bVar1, bVar2) of both comparisons are each TRUE.

```
VAR
wVar1 : WORD;
wVar2 : WORD;
bVar1 : BOOL;
bVar2 : BOOL;
END_VAR
wVar1 := 65535;
wVar2 := 0;
bVar1 := TO_WORD(wVar1 + 1) = wVar2;
bVar2 := TO_WORD(wVar2 - 1) = wVar1;
```

Arithmetic oper- $\left.{ }^{\wedge}\right\rangle$ Chapter 6.4.1.20.3.2 "Operator 'ADD'" on page 2133
y Chapter 6.4.1.20.3.4 "Operator 'SUB"" on page 2135
ⓨ Chapter 6.4.1.20.3.3 "Operator 'MUL"' on page 2134
« Chapter 6.4.1.20.3.5 "Operator 'DIV'" on page 2136
² Chapter 6.4.1.20.3.6 "Operator 'MOD"' on page 2137
\& Chapter 6.4.1.20.3.7 "Operator 'MOVE"' on page 2137
Chapter 6.4.1.20.3.8 "Operator 'INDEXOF"' on page 2137
凶 Chapter 6.4.1.20.3.9 "Operator 'SIZEOF'" on page 2138
« Chapter 6.4.1.20.3.10 "Operator 'XSIZEOF'" on page 2138

Bitstring opera- ${ }^{3}$ Chapter 6.4.1.20.3.12 "Operator 'AND'" on page 2139
tors $\quad \Perp$ Chapter 6.4.1.20.3.13 "Operator 'OR' " on page 2139
4y Chapter 6.4.1.20.3.14 "Operator 'XOR'" on page 2140
« ${ }^{\wedge}$ Chapter 6.4.1.20.3.11 "Operator 'NOT'" on page 2139
« Chapter 6.4.1.20.3.15 "Operator 'AND_THEN'" on page 2140

* Chapter 6.4.1.20.3.16 "Operator 'OR_ELSE"' on page 2140

Bitshift opera- $y^{4}$ Chapter 6.4.1.20.3.17 "Operator 'SHL"' on page 2141
tors $\quad \Leftrightarrow$ Chapter 6.4.1.20.3.18 "Operator 'SHR"" on page 2142
๕ Chapter 6.4.1.20.3.19 "Operator 'ROL'" on page 2143

* Chapter 6.4.1.20.3.20 "Operator 'ROR'" on page 2144

Selection opera- $\xi^{〔}$ Chapter 6.4.1.20.3.21 "Operator 'SEL"" on page 2145
tors $\quad \Leftrightarrow$ Chapter 6.4.1.20.3.22 "Operator 'MAX'" on page 2146
\& Chapter 6.4.1.20.3.23 "Operator 'MIN'" on page 2146
${ }^{\star}{ }^{4}$ Chapter 6.4.1.20.3.24 "Operator 'LIMIT"' on page 2147
«2 Chapter 6.4.1.20.3.25 "Operator 'MUX'" on page 2147

Comparison A comparison operator is a Boolean that compares two inputs (first and second operand).
operators
y Chapter 6.4.1.20.3.26 "Operator 'GT'" on page 2148
„ Chapter 6.4.1.20.3.27 "Operator 'LT'" on page 2148

* Chapter 6.4.1.20.3.28 "Operator 'LE'" on page 2148

Chapter 6.4.1.20.3.29 "Operator 'GE"' on page 2149
« Chapter 6.4.1.20.3.30 "Operator 'EQ'" on page 2149
Chapter 6.4.1.20.3.31 "Operator 'NE'" on page 2149

Address opera- ${ }^{*} \boldsymbol{y}$ Chapter 6.4.1.20.3.32 "Operator 'ADR'" on page 2150
tors
y Chapter 6.4.1.20.3.33 "Operator 'Content Operator'" on page 2151
Chapter 6.4.1.20.3.34 "Operator 'BITADR'" on page 2151

Call operators ${ }^{\circledR}$ Chapter 6.4.1.20.3.35 "Operator 'CAL"" on page 2152

Type conversion You can explicitly call type conversion operators. The type conversion operators described operators below are available for typed conversions from one elementary type to another elementary type, as well as for overloading. Conversions from a larger type to a smaller type are also implicitly possible (for example, from INT to BYTE or from DINT to WORD).

Typed conversion: <elementary data type> _TO_ <another elementary data type>
Overloaded conversion: TO_ <elementary data type>

## Elementary data

types:

```
<elementary data type> =
    UXINT | XINT | XWORD | BIT | BOOL | BYTE | DATE | DATE AND_TIME
    \ DINT | DT | DWORD | INT | LDATE | LDATE_AND_TIME | LDT | LINNT |
LREAL | LTIME | LTOD | LWORD | REAL | SINT | TIME | TOD | UDINT |
UINT | ULINT | USINT | WORD
```

The keywords T, TIME_OF_DAY and DATE_AND_TIME are alternative forms for the data types TIME, TOD, and DT. T, TIME_OF_DAY and DATE_AND_TIME are not represented as a type conversion command.

## NOTICE!

If the operand value for a type conversion operator is outside of the value range of the target data type, then the result output depends on the processor type and is therefore undefined. This is the case, for example, when a negative operand value is converted from LREAL to the target data type UINT.

Information can be lost when converting from larger data types to smaller data types.

## NOTICE!

## String manipulation when converting to STRING or WSTRING

When converting the type to STRING or WSTRING, the typed value is leftaligned as a character string and truncated if it is too long. Therefore, declare the return variable for the type conversion operators <>_TO_STRING and <>_TO_WSTRING long enough that the character string has enough space without any manipulation.

## See also

- ⓨ Chapter 6.4.1.20.3.39 "Floating-Point Number Conversion" on page 2171
- $\Leftrightarrow$ Chapter 6.4.1.20.3.41 "Time Conversion" on page 2182
- $\Leftrightarrow$ Chapter 6.4.1.20.3.42 "Date and Time Conversion" on page 2187
- $\quad 4$ Chapter 6.4.1.20.3.40 "String Conversion" on page 2174
- $\Leftrightarrow$ Chapter 6.4.1.20.3.43 "Operator 'TRUNC' " on page 2193
- ※ Chapter 6.4.1.20.3.44 "Operator 'TRUNC_INT' " on page 2193

Numeric Opera- "y Chapter 6.4.1.20.3.45 "Operator 'ABS'" on page 2194 tors
ⓨ Chapter 6.4.1.20.3.46 "Operator 'SQRT"' on page 2194
*) Chapter 6.4.1.20.3.47 "Operator 'LN"' on page 2194
« Chapter 6.4.1.20.3.48 "Operator 'LOG'" on page 2195
Chapter 6.4.1.20.3.49 "Operator 'EXP'" on page 2195
Chapter 6.4.1.20.3.50 "Operator 'EXPT"' on page 2195
出 Chapter 6.4.1.20.3.51 "Operator 'SIN"" on page 2196
② Chapter 6.4.1.20.3.54 "Operator 'ASIN'" on page 2197
« Chapter 6.4.1.20.3.52 "Operator 'COS"" on page 2196

② Chapter 6.4.1.20.3.53 "Operator 'TAN"' on page 2197

* $\boldsymbol{y}^{4}$ Chapter 6.4.1.20.3.55 "Operator 'ACOS'" on page 2198

Chapter 6.4.1.20.3.56 "Operator 'ATAN"" on page 2198

| Namespace | Namespace operators are extended from IEC 61131-3 operators. They make it possible for <br> operators |
| :--- | :--- |
| you to provided unique access to variables and modules, even when you use the same name |  |
| multiple times for variables or modules in a project. |  |

ⓨ Chapter 6.4.1.20.3.70 "Operator - Global namespace" on page 2216
« Chapter 6.4.1.20.3.71 "Operator - Namespace for global variables lists" on page 2216
« Chapter 6.4.1.20.3.73 "Operator - Enumeration namespace" on page 2217
Chapter 6.4.1.20.3.72 "Operator - Library namespace" on page 2217
Chapter 6.4.1.20.3.74 "Operator '__POOL"' on page 2217

Multicore opera- Working with different tasks requires the synchronization of these tasks. This is especially true tors when working on multicore platforms. Some special operators are provided in CODESYS to support this synchronization.
These operators are extensions of IEC-61131-3. The operators TEST_AND_SET and __COMPARE_AND_SWAP are used for similar tasks.
«2 Chapter 6.4.1.20.3.69 "Operator 'TEST_AND_SET'" on page 2215
Chapter 6.4.1.20.3.65 "Operator '__COMPARE_AND_SWAP" on page 2212
${ }^{\wedge}$ Chapter 6.4.1.20.3.66 "Operator '__XADD"' on page 2213

Other operators $\sum^{\sharp}$ Chapter 6.4.1.20.3.57 "Operator '__DELETE" on page 2198
« Chapter 6.4.1.20.3.58 "Operator '__ISVALIDREF" on page 2201

* Chapter 6.4.1.20.3.59 "Operator '__NEW'" on page 2201

② Chapter 6.4.1.20.3.60 "Operator '__QUERYINTERFACE"' on page 2204
« $\boldsymbol{y}^{2}$ Chapter 6.4.1.20.3.61 "Operator '__QUERYPOINTER"' on page 2205

* Chapter 6.4.1.20.3.75 "Operator 'INI"" on page 2218

๕ Chapter 6.4.1.20.3.62 "Operators '__TRY', '__CATCH', '__FINALLY', '__ENDTRY"' on page 2206
« Chapter 6.4.1.20.3.67 "Operator '__POSITION"" on page 2214
« Chapter 6.4.1.20.3.68 "Operator '__POUNAME"' on page 2214

## Operator 'ADD'

The IEC operator adds variables.
Permitted data types: __UXINT | __XINT | __XWORD | BYTE | DATE |
DATE_AND_TIME | DINT | DT | DWORD | INT | LDATE | LDATE_AND_TIME | LDT $\overline{\text { LINT }}$ | LREAL | LTIME | LTOD | LWORD | REAL | SINT ${ }^{-}$| TĪME | TIME_OF_DAY | TOD | UDINT | UINT | ULINT | USINT | WORD

Possible combinations for time data types:

- TIME + TIME = TIME
- TIME + LTIME = LTIME
- LTIME + LTIME = LTIME

Possible combinations for date and time data types:

- TOD + TIME = TOD
- $\mathrm{DT}+\mathrm{TIME}=\mathrm{DT}$
- TOD + LTIME = LTOD
- DT + LTIME = LDT
- LTOD + TIME = LTOD
- LDT + LTIME = LDT
- LTOD +LTIME = LTOD
- LDT + LTIME = LDT

Feature in the FBD/LD editor: You can extend the ADD operator to function block inputs. The number of additional function block inputs is limited.

| Examples | ST: |
| :--- | :--- |
|  | var $1:=7+2+4+7 ;$ |
|  | FBD: |



## Operator 'MUL'

This IEC operator is used for multiplying variables.
Permitted data types: BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL, TIME

Feature in the FBD/LD editor: You can extend the MUL operator to additional function block inputs. The number of additional function block inputs is limited.

## Examples ST:

$$
\text { var1 }:=7 * 2 * 4 * 7
$$

FBD:


## Operator 'SUB'

The IEC operator subtracts variables.
Permitted data types: BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL, TIME, LTIME, TIME_OF_DAY (TOD), LTIME_OF_DAY (LTOD), DATE, LDATE, DATE_AND_TIME (DT) LDATE_AND_TIME (DT)

Possible combinations for time data types:

- TIME - TIME = TIME
- LTIME - LTIME = LTIME

Possible combinations for date and time data types:

- DATE - DATE = TIME
- LDATE - LDATE = LTIME
- TOD - TIME = TOD
- LTOD - LTIME = LTOD
- TOD - TOD = TIME
- LTOD - LTOD = LTIME
- DT - TIME = DT
- LDT - LTIME = LDT
- DT - DT = TIME
- LDT - LDT = LTIME

Negative TIME/LTIME values are undefined.

| Examples | ST: |
| :--- | :--- |
|  | var1 $:=7-2 ;$ |
|  | FBD: |



## Operator 'DIV'

This IEC operator is used for dividing variables.
Permitted data types: BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL, TIME

## - NOTICE!

Division by zero may have different results depending on the target system.

Examples ST :
var1 := 8/2;
FBD:

1. Series of DIV blocks, 2. Single DIV block, 3. DIV blocks with EN/ENO parameters


Please note that it is possible to monitor division by zero at runtime by using the implicit monitoring functions CheckDivInt, CheckDivLint, CheckDivReal, and CheckDivLReal.

See also

- ${ }^{4}$, Chapter 6.4.1.21.2.22.3 "POU 'CheckDivInt"' on page 2505
- $y$ Chapter 6.4.1.21.2.22.4 "POU 'CheckDivLInt"" on page 2505
- ${ }^{4}$ Chapter 6.4.1.21.2.22.5 "POU 'CheckDivReal"' on page 2506
- $\sum^{\circ}$ Chapter 6.4.1.21.2.22.6 "POU 'CheckDivLReal"" on page 2507


## Operator 'MOD'

This IEC operator is used for modulo division.
The result of the function is the integer remainder of division.
Permitted data types: BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT

## - NOTICE!

Division by zero may have different results depending on the target system.

Examples $\quad$| Result in Var1: 1 |
| :--- |
|  |
| ST: |
|  |
| var1 $:=9$ MOD 2; |
|  |
| FBD: |

## Operator 'MOVE'

This IEC operator is used for assigning a variable to another variable of a corresponding type.
Because the MOVE block is available in the CFC, FBD, and LD editors, you can also use the EN/ENO functionality for variable assignment.

## CFC with EN/ENO function:

CODESYS assigns the value of var1 to var2 only if "en_i" yields TRUE.


ST:
ivar2 := MOVE(ivar1);
This corresponds to:
ivar2 := ivar1;

## Operator 'INDEXOF'

This operator is an extension of the IEC 61131-3 standard.
Instead of the INDEXOF operator, the ADR operator is provided in CODESYS V3 for obtaining a pointer at the index of a block.

See also

- *y Chapter 6.4.1.20.3.32 "Operator 'ADR'" on page 2150


## Operator 'SIZEOF'

The operator is an extension of the IEC 61131-3 standard.
The operator is used for defining the number of bytes that are required by the variable x . The operator SIZEOF always yields an unsigned value. The type of return variable adapts to the detected size of the variable x .


In compiler version 3.5.16.0 and higher, the operator XSIZEOF should be used instead of this operator.

| Return value of SIZEOF $(x)$ | Data type of the constant which CODESYS <br> uses implicitly for the detected size. |
| :--- | :--- |
| $0<=$ size of $x<256$ | USINT |
| $256<=$ size of $x<65536$ | UINT |
| $65536<=$ size of $x<4294967296$ | UDINT |
| $4294967296<=$ size of $x$ | ULINT |

```
Examples Result in var1:10.
ST:
arr1 : ARRAY[0..4] OF INT;
varl : INT;
var1 := SIZEOF(arr1); (* var1 := USINT#10; *)
```

See also

- « Chapter 6.4.1.20.3.10 "Operator 'XSIZEOF"' on page 2138


## Operator 'XSIZEOF'

The operator is an extension of the IEC 61131-3 standard.
The operator is used for defining the number of bytes that are required by the variable x . The data type of the return value is ULINT on 64-bit platforms and UDINT on all other platforms.

The operator XSIZEOF should be used instead of the operator SIZEOF. Because the data type of the return value is fixed, problems do not occur for XSIZEOF, which do occur in the case of the operator SIZEOF.

## Example Variable udiVarX is

ST:

```
udiVarX : UDINT; (* Data type for 64-bit platforms: ULINT *)
udiVarX := XSIZEOF(<variable>);
```

The variable udiVarX contains the number of bytes that the variable <variable> requires.

## See also

- \# Chapter 6.4.1.20.3.9 "Operator 'SIZEOF"' on page 2138


## Operator 'NOT'

This IEC operator is used for the bitwise NOT of a bit operand.
When the respective input bit yields 0 , the output bit also yields 1 , and vice-versa.
Permitted data types: BOOL, BYTE, WORD, DWORD, LWORD

## Examples

Result in var1: 2\#0110_1100
ST:
var1 := NOT 2\#1001_0011;
FBD:


## Operator 'AND'

This IEC operator is used for the bitwise AND of bit operands.
When the input bits all yield 1 , the output bit also yields 1 ; otherwise 0.
Permitted data types: BOOL, BYTE, WORD, DWORD, LWORD
Examples Result in var1 ist 2\#1000_0010
ST:
var1 := 2\#1001_0011 AND 2\#1000_1010;
FBD:


## Operator 'OR'

This IEC operator is used for the bitwise OR of bit operands.
When at least one of the input bits yields 1 , the output bit also yields 1 ; otherwise 0 .
Permitted data types: BOOL, BYTE, WORD, DWORD, LWORD

```
Examples Result in Var1 ist 2#1001_1011
ST:
    Var1 := 2#1001_0011 OR 2#1000_1010;
FBD:
```



## Operator 'XOR'

This IEC operator is used for the bitwise XOR of bit operands.
When only one of the two input bits yields 1 , the output bit also yields 1 . When both inputs yield 1 or 0 , then the output yields 0 .

Permitted data types: BOOL, BYTE, WORD, DWORD, LWORD

## NOTICE!

Please note the following behavior of the XOR block in extended form (more than two inputs): CODESYS compares the inputs in pairs and then the corresponding results (according to the standard, but not necessarily according to expectations).

```
Examples Result in var1:2#0001_1001
ST:
    var1 := 2#1001_0011 XOR 2#1000_1010;
FBD:
```



## Operator 'AND_THEN'

This operator is an extension of the IEC 61131-3 standard.
The AND_THEN operator is permitted only for programming in structured text with the AND operation of BOOL and BIT operands with short-circuit evaluation. This means that:
When all operands yield TRUE, the result of the operands also yield TRUE; otherwise FALSE.
However, CODESYS also executes the expressions on other operands only if the first operand of the AND_THEN operator is TRUE. This can prevent problems with null pointers, for example in conditions sü as IF (ptr <> 0 AND_THEN ptr^ = 99) THEN....
In contrast, CODESYS always evaluates all operands when using the AND IEC operator.
See also

- ${ }^{\text {® }}$ Chapter 6.4.1.20.3.12 "Operator 'AND'" on page 2139


## Operator 'OR_ELSE'

This operator is an extension of the IEC 61131-3 standard.
The OR_ELSE operator is permitted only for programming in structured text: OR operation of BOOL and BIT operands; with short-circuit evaluation. This means:

When at least one of the operands yields TRUE, the result of the operation also yields TRUE; otherwise FALSE.
In contrast to using the OR IEC operator, for OR_ELSE the expressions on all other operators are not evaluated as soon as one of the operands is evaluated as TRUE.

## Example

```
VAR
    bEver: BOOL;
        bX: BOOL;
    dw: DWORD := 16#000000FF;
END VAR
bEver := FALSE;
bX := dw.8 OR_ELSE dw.1 OR_ELSE dw.1 OR_ELSE (bEver := TRUE);
```

$d w .8$ is FALSE and dw. 1 is TRUE; therefore $b X$ is the result of the operation TRUE. However, the expression at the third input is not executed, and bEver remains FALSE. On the other hand, if the standard OR operation was used, bEver would be set to TRUE.

## See also

- $\xi^{\circledR}$ Chapter 6.4.1.20.3.13 "Operator 'OR'" on page 2139


## Operator 'SHL'

This IEC operator is used for bitwise shift of an operand to the left.
erg := SHL (in, n)
in: Operand that is shifted to the left
n : Number of bits to shift in to the left

## NOTICE!

If $n$ overwrites the data type width, then it depends on the target system how the BYTE, WORD, DWORD, and LWORD operands are padded. The target systems cause padding with zeros or n MOD <tab width>.

## NOTICE!

Please note the number of bits that CODESYS uses for this operation as defined by the data type of the input variable in.

Examples The results for erg_byte and erg_word are different, although the values of the in_byte and in_word input variables are the same and the data types of the input variables are different.

ST:

```
PROGRAM shl_st
VAR
    in byte : BYTE := 16#45; (* 2#01000101 )
    in_word : WORD := 16#0045; (* 2#0000000001000101 )
    erg_byte : BYTE;
    erg_word : WORD;
    n: BYTE := 2;
END_VAR
erg_byte := SHL(in_byte,n); (* Result is 16#14, 2#00010100 *)
erg_word := SHL(in_word,n); (* Result is 16#0114,
2#0\overline{0}00000100010100-*)
```

FBD:


## Operator 'SHR'

This IEC operator is used for bitwise shift of an operand to the right.
erg := SHR (in, n)
in: Operand that is shifted to the right
n : Number of bits for shifting in to the right

## NOTICE!

If $n$ overwrites the data type width, then it depends on the target system how the BYTE, WORD, DWORD, and LWORD operands are padded. The target systems cause padding with zeros or $n$ MOD <tab width>.

```
Examples ST:
    PROGRAM shr_st
    VAR
    in_byte : BYTE:=16#45; (* 2#01000101 )
    in_word : WORD:=16#0045; (* 2#0000000001000101 )
    erg_byte : BYTE;
    erg_word : WORD;
    n: BYTE :=2;
END_VAR
erg_byte := SHR(in_byte,n); (* Result is 16#11, 2#00010001 *)
erg word := SHR(in-word,n); (* Result is 16#0011,
2#0\overline{0}000000000010001-*)
FBD:
```



## Operator 'ROL'

This IEC operator is used for bitwise rotation of an operand to the left.
Permitted data types: BYTE, WORD, DWORD, LWORD
erg := ROL (in, n)
CODESYS moves in $n$-times one bit to the left and adds the bit to the leftmost position from the right.

## NOTICE!

Please note the number of bits that CODESYS uses for this operation as defined by the data type of the input variable in. If this is a constant, then CODESYS uses the smallest possible data type. The data type of the output variables still does not influence this operation.


## Operator 'ROR'

This IEC operator is used for bitwise rotation of an operand to the right.
Permitted data types: BYTE, WORD, DWORD, LWORD
erg := ROR(in, $n$ )
CODESYS moves in $n$-times one bit to the right and adds the bit to the rightmost position from the left.


Please note the number of bits that CODESYS uses for this operation as defined by the data type of the input variable in. If this is a constant, then CODESYS uses the smallest possible data type. The data type of the output variables still does not influence this operation.

Examples The results for erg_byte and erg_word are different depending on the data type of the input variables, although the values of the in_byte and in_word input variables are the same.
ST:

```
PROGRAM ror_st
VAR
    in byte : BYTE := 16#45;
    in_word : WORD := 16#45;
    erg_byte : BYTE;
    erg_word : WORD;
    n: BYTE := 2;
    END_VAR
    erg_byte := ROR(in_byte,n); (* Result: 16#51 *)
    erg_word := ROR(in_word,n); (* Result: 16#4011 *)
```

FBD:


## Operator 'SEL'

The IEC operator is used for bitwise selection.
OUT := SEL(G, INO, IN1) means:
OUT := INO; if $\mathrm{G}=$ FALSE
OUT := IN1; if G = TRUE
Permitted data types:
INO, ..., INn and OUT: Any identical data type. Make sure that variables of the identical type are used at all three positions, especially when using user-defined data types. The compiler checks for type identity and returns any compile errors. The assignment of function block instances to interface variables is specifically not supported.
G: BOOL

## NOTICE!

When $G$ is TRUE, CODESYS does not compute an expression that precedes INO.When $G$ is FALSE, CODESYS does not compute an expression that precedes IN1.

Caution: In the case of graphical programming languages, the expressions at IN0 and IN1 are computed independently of the G input when a "Box", "Jump", "Return", "Line Branch", or "Edge Detection" precedes.

```
Examples ST:
    Var1 := SEL(TRUE,3,4); (* Result: 4 *)
FBD:
```



## Operator 'MAX'

This IEC operator is used for the maximum function. It yields the largest value of two values.
OUT := MAX(INO, IN1)
Permitted data types: all

## Examples <br> ST:

Result: 90
Var1 := MAX $(30,40)$;
Var1 := MAX $(40, \operatorname{MAX}(90,30))$;
FBD:
Result: 90


## Operator 'MIN'

This IEC operator is used for the minimum function. It yields the smallest value of two values.
OUT := MIN(IN0,IN1)
Permitted data types: all

```
Examples Result: 30
    ST:
    Var1:=MIN (90,30);
    Var1 := MIN(MIN (90,30),40);
FBD:
```



## Operator 'LIMIT'

This IEC selection operator is used for limiting.
OUT := LIMIT (Min, IN, Max)
Means: OUT := MIN (MAX (IN, Min), Max)
Max is the upper limit and Min is the lower limit for the result. If the IN value is above the Max upper limit, then LIMIT yields Max. If the value of IN is below the Min lower limit, then the result is Min.

Permitted data types for IN and OUT: all

## Examples Result in Var1 is 80

ST:

```
Var1 := LIMIT(30,90,80);
```


## Operator 'MUX'

This IEC operator is used as a multiplexer.
OUT := MUX(K, INO,...,INn)
Means: OUT = IN_K
Permitted data type for K : BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, LINT, ULINT, oUDINT.

INO, ..., INn, and OUT: Any identical data type. Make sure that variables of the identical type are used at all three positions, especially when using user-defined data types. The compiler checks for type identity and returns any compile errors. The assignment of function block instances to interface variables is specifically not supported.

MUX selects the $K$-th value from a set of values. The first value is $K=0$. If $K$ is greater than the number of other inputs ( $n$ ), then CODESYS passes on the last value (INn).

## NOTICE!

For runtime optimization, CODESYS computes only the expression that precedes IN_K. However, CODESYS computes all branches in simulation mode.

## Examples Result in Var1 is 30.

ST:

```
Var1 := MUX (0, 30,40,50,60,70,80);
```


## Operator 'GT'

This IEC operator is used for the "greater than" function.
Permitted data types of the operands: any basic data type.
If the first operand is greater than the second operand, then the operator yields the result TRUE; otherwise FALSE.


FBD:


## Operator 'LT'

This IEC operator is used for the "less than" function.
Permitted data types of the operands: any basic data type.
If the first operand is less than the second operand, then the operator yields the result TRUE; otherwise FALSE.

| Result: TRUE |  |
| :--- | :--- |
| ST: |  |
|  | Var1 $:=20<30 ;$ |
| LT T |  |

## Operator 'LE'

This IEC operator is used for the "less than or equal to" function.
Permitted data types of the operands: any basic data type.
If the first operand is less than or equal to the second operand, then the operator yields the result TRUE; otherwise FALSE.

## Examples Result in Var1: TRUE

ST:

```
Var1 := 20 <= 30;
```

$20-$ LE

## Operator 'GE'

This IEC operator is used for the "greater than or equal to" function.
Permitted data types of the operands: any basic data type.
If the first operand is greater than or equal to the second operand, then the operator yields the result TRUE; otherwise FALSE.

| Examples $\quad$ Result: TRUE |  |
| :--- | :--- |
|  | ST: |
|  | VAR1 $:=60>=40 ;$ |
|  | FBD: |



## Operator 'EQ'

This IEC operator is used for the "equals" function.
Permitted data types of the operands: any basic data type, depending on target system and compiler version: structure data type.

If the operands are equal, then then the operator yields the result TRUE, otherwise FALSE.


## Operator 'NE'

This IEC operator is used for the "does not equal" function.
Permitted data types of the operands: any basic data type, depending on target system and compiler version: structure data type.

If the operands are not equal, then then the operator yields the result TRUE; otherwise FALSE. If the target system supports the data type, then as from compiler version >= 3.5.7.0 also operands of type STRIUCT (structure) can be compared. Example: IF (stStruct1 := stStruct2) THEN....

Examples $\quad$ Result in Varl is FALSE
ST:
Var1 := 40 <> 40;
FBD:


## Operator 'ADR'

The operator is an extension of the IEC 61131-3 standard.
ADR yields the 32-bit address (or the 64-bit address, if possible) of its argument. You can pass this address to the manufacturer functions or assign them to a pointer in the project.
Syntax

```
VAR
    <address name> : DWORD | LWORD | POINTER TO < basis data type>
END_VAR
<address name> := ADR( <variable name> );
```


## Example

```
FUNCTION_BLOCK FB_Address
VAR
    piAddress1: POINTER TO INT;
    iNumber1: INT := 5;
    lwAddress2
            iNumber2: INT := 10;
END_VAR
piAddress1 := ADR(iNumber1); // piNumber is assigned to address of
iNumber1
lwAddress2 := ADR(iNumber2); // 64 bit runtime system
```


## 0 <br> NOTICE! <br> In contrast to CoDeSys V2.3, you can use the ADR operator with function names, program names, function block names, and method names. Therefore, ADR replaces the INDEXOF operator. <br> When using function pointers, note that you can pass a function pointer to external libraries, but it is not possible to call a function pointer from within CODESYS. To enable a system call (runtime system), you must set the respective object property ("Build" tab) for the function object.

## CAUTION!

When you use an online change, the contents of addresses can shift. As a result, POINTER TO variables could point to an invalid memory area. To avoid problems, you should make sure that the value of pointers is updated in every cycle.

## CAUTION!

Do not return Pointer-TO variables of functions and methods to the caller or assign them to global variables.

## See also

- $\quad \Leftrightarrow$ Chapter 6.4.1.20.5.13 "Pointers" on page 2243


## Operator 'Content Operator'

This operator is an extension of the IEC 61131-3 standard.
You can use this operator to dereference pointers by appending the operator as ${ }^{\wedge}$ to the pointer identifier.

## CAUTION!

When using pointers to addresses, please note that applying an online change can shift address contents.

## Example ST:

```
pt : POINTER TO INT;
var_int1 : INT;
var_int2 : INT;
pt := ADR(var_int1);
var_int2 := p\overline{t};
```


## Operator 'BITADR'

The operator is an extension of the IEC 61131-3 standard.
BITADR yields the bit offset within a segment in a DWORD.

## NOTICE!

```
The offset depends on whether the "Byte addressing" option is selected or cleared in the target system settings.
```

The highest value nibble ( 4 bits) in this DWORD defines the memory range:
Marker M: 16\#40000000
Input I: 16\#80000000
Output Q: 16\#C0000000


## Operator 'CAL'

This IEC operator is used for calling function blocks.
In IL, CAL calls the instance of a function block.
CAL <function block> (<input variable1> := <value>, <input
variableN> := <value>)
Example Call of the Inst instance of a function block with assignment of the input variables Par1 and Par2 with 0 or TRUE.

```
CAL Inst(Par1 := 0, Par2 := TRUE);
```


## Overloading

## NOTICE!

If the operand value for a type conversion operator is outside of the value range of the target data type, then the result output depends on the processor type and is therefore undefined. This is the case, for example, when a negative operand value is converted from LREAL to the target data type UINT.
Information can be lost when converting from larger data types to smaller data types.

## NOTICE!

The rounding logic for borderline cases depends on the target system or the FPU (Floating Point Unit) of the target system. For example, a value of -1.5 can be converted differently on different controllers.

Catch value ranges overflows across the application to program code-independent from the target system.

The IEC61131-3 specification does not provide for overloaded functions.
If you want to program strictly according to IEC61131-3, then you should use the operators of the syntax <type> _TO_ <another type> as described in the following sections.

The rules for typed conversions also apply here for overloading.

The operators convert values into other data types, explicitly specifying only a target data type and no initial data type (data type of the operands) ("overloaded conversion"). Overloading is not part of the IEC 61131-3 specification.

## Call syntax

```
<variable name> := <TO operator> ( <operand> );
<operand> = <variable name> | <literal>
```


## Operators

```
TO
```

$\qquad$

``` UXINT
TO XINT
TO XWORD
TO BIT
TO_BYTE
TO BOOL
TO DATE
TO DINT
TO-DT
TO DWORD
TO_INT
TO-LDATE
TO LDT
TO LINT
TO \({ }^{-}\)LREAL
TO LTIME
TO LTOD
TO LWORD
TO REAL
TO SINT
TO-STRING
TO TIME
TO_TOD
TO-UDINT
TO_UINT
TO ULINT
TO-USINT
TO_WORD
TO_WSTRING
```


## Examples

```
ST implemen-
tation lan-
guage:
```

```
VAR
```

VAR
iNumber_1 : INT;
iNumber_1 : INT;
rNumber 2 : REAL := 123.456;
rNumber 2 : REAL := 123.456;
iNumber_2 : INT;
iNumber_2 : INT;
xIsTrue : BOOL;
xIsTrue : BOOL;
sOutputText : STRING;
sOutputText : STRING;
sText : STRING := 'Hello World!';
sText : STRING := 'Hello World!';
wsText: WSTRING;
wsText: WSTRING;
dateEvent : DATE := D\#2019-9-3;
dateEvent : DATE := D\#2019-9-3;
uiEvent : UINT;
uiEvent : UINT;
uxiData : __UXINT;
uxiData : __UXINT;
END_VAR
END_VAR
iNumber_1 := TO_INT(4.22);
iNumber_1 := TO_INT(4.22);
iNumber_2 := TO_INT(rNumber_2);
iNumber_2 := TO_INT(rNumber_2);
xIsTrue -}:= TO_B\overline{OLL(1);
xIsTrue -}:= TO_B\overline{OLL(1);
sOutputText := TO_STRING(342);
sOutputText := TO_STRING(342);
wsText := TO_WSTRING(sText);
wsText := TO_WSTRING(sText);
uiEvent := T\overline{O}\mathrm{ UINT (dateEvent); (* Result: 44288 *)}
uiEvent := T\overline{O}\mathrm{ UINT (dateEvent); (* Result: 44288 *)}
(* Result: "Hello World!" *)
(* Result: "Hello World!" *)
uxiData := TO___UXINT(iNumber_2); (* Result: 123 *)

```
uxiData := TO___UXINT(iNumber_2); (* Result: 123 *)
```

See also

- \& "Type conversion operators" on page 2132
- $\quad$ \& Chapter 6.4.1.20.3.37 "Boolean Conversion" on page 2154
- $y^{4}$ Chapter 6.4.1.20.3.38 "Integer Conversion" on page 2159
- $\boldsymbol{y y}^{*}$ Chapter 6.4.1.20.3.39 "Floating-Point Number Conversion" on page 2171
- $\quad$ \& Chapter 6.4.1.20.3.40 "String Conversion" on page 2174
- $\Leftrightarrow$ Chapter 6.4.1.20.3.42 "Date and Time Conversion" on page 2187
- $\Leftrightarrow$ Chapter 6.4.1.20.3.41 "Time Conversion" on page 2182


## Boolean Conversion

## O NOTICE! <br> String manipulation when converting to STRING or WSTRING

When converting the type to STRING or WSTRING, the typed value is leftaligned as a character string and truncated if it is too long. Therefore, declare the return variable for the type conversion operators <>_TO_STRING and <>_TO_WSTRING long enough that the character string has enough space without any manipulation.

The operators convert a Boolean value into the specified data types and return a type-converted value.

## Call syntax

```
<variable name> := <BOOL to operator> ( <operand> );
<operand> = <variable name> | <literal>
```


## Operators

```
BOOL_TO UXINT
BOOL TO XINT
BOOL_TO_XWORD
BOOL TO_BIT
BOOL_TO_BYTE
BOOL_TO_DATE
BOOL_TO_DINT
BOOL TO DT
BOOL_TO-DWORD
BOOL_TO_INT
BOOL_TO_LDATE
BOOL-TO_LDT
BOOL_TO_LINT
BOOL TO LREAL
BOOL_TO_LTIME
BOOL_TO_LTOD
BOOL TO LWORD
BOOL-TO-REAL
BOOL_TO_SINT
BOOL_TO_STRING
BOOL_TO-TIME
BOOL_TO_TOD
BOOL TO UDINT
BOOL_TO-UINT
BOOL_TO_ULINT
BOOL_TO_USINT
BOOL_TO-WORD
BOOL_TO_WSTRING
```

When the operand value is TRUE, the following typed values are returned:

- BOOL_TO_DATE: D\#1970-1-1 // The zeroth bit is set, but does not effect the display.
- BOOL_TO_DT:DT\#1970-01-01-0:0:1
- BOOL_TO_LTIME: LTIME\#1NS
- BOOL TO REAL: '1'
- BOOL_TO_STRING: 'TRUE'
- BOOL_TO_TOD: TOD\#0:0:0.001
- BOOL_TO_TIME: T\#1MS
- BOOL_TO_WSTRING: "TRUE"

When the operand value is FALSE, the following typed values are returned:

- BOOL_TO_DATE: D\#1970-1-1
- BOOL_TO_DT: DT\#1970-01-01-00:00:00
- BOOL_TO_LTIME: LTIME\#ONS
- BOOL_TO_REAL: ' $0.0^{\prime}$
- BOOL_TO_STRING: 'FALSE'
- BOOL_TO_TOD: TOD\#0:0:0
- BOOL_TO_TIME: T\#OMS
- BOOL_TO_WSTRING: "FALSE"


## Examples

## ST implemen-

tation language

```
FUNCTION_BLOCK FB_ConvertFromBool
VAR
VAR
    uxiReturn 1: UXINT;
    uxiReturn_10: - UXINT;
    iReturn_2: __XINT;
    iReturn 20: XINT;
    xwReturn 3: -XWORD;
    xwReturn_30: __XWORD;
    bitReturn 4: BOOL;
    bitReturn 40: BOOL;
    bReturn_6: BYTE;
    bReturn_60: BYTE;
    dateRetūrn 7: DATE;
    dateReturn_70: DATE;
    dtReturn_8: DATE_AND_TIME;
    dtReturn_80: DAT\overline{E_AND}\overline{D}_TIME;
    diReturn_9: DINT;
    diReturn_90: DINT;
    dtReturn 10: DATE AND TIME;
    dtReturn_100: DATE_AND_TIME;
    dwReturn_11: DWORD;
    dwReturn 110: DWORD;
    iReturn_12: INT;
    iReturn_120: INT;
    liReturn_13: LINT;
    liReturn_130: LINT;
    lrReturn_14: LREAL;
    lrReturn_140: LREAL;
    lwReturn_15: LWORD;
    lwReturn 150: LWORD;
    rReturn 1'6: REAL;
    rReturn_160: REAL;
    siReturn_17: SINT;
    siReturn 170: SINT;
    sReturn_18: STRING;
    sReturn_180: STRING;
    todReturn 19: TIME OF DAY;
    todReturn_190: TIME O\overline{F DAY;}
    timReturn_20: TIME;
    timReturn 200: TIME;
    todReturn 21: TIME OF DAY;
    todReturn_210: TIME_OF_DAY;
    udiReturn 22: UDINT;
    udiReturn 220: UDINT;
    uiReturn_23: UINT;
    uiReturn_230: UINT;
    uliReturn}\mathrm{ 24: ULINT;
    uliReturn_240: ULINT;
    usiReturn_25: USINT;
    usiReturn 250: USINT;
    wReturn_26: WORD;
    wReturn 260: WORD;
    wsReturn_27: WSTRING;
    wsReturn_270: WSTRING;
END_VAR
// Return value of operand = TRUE or FALSE
uxiReturn_1 := BOOL_TO__UXINT(TRUE);
```

```
uxiReturn_10 := BOOL_TO___UXINT(FALSE);
iReturn_2 := BOOL_TO___XINT(TRUE);
iReturn_20 := BOO\overline{L_TO__XINT(FALSE);}
xwReturn_3 := BOOL_TO___XWORD(TRUE);
xwReturn_}30:= BOO\overline{L}_T\overline{O___XWORD(FALSE);
bitReturn_4 := BOOL_TO_BIT(TRUE);
bitReturn_40 := BOO\overline{L_TO_BIT(FALSE);}
bReturn_6 := BOOL_TO_BYTE (TRUE);
bReturn_60 := BOO\overline{L}_TO_BYTE (FALSE);
dateReturn_7 := BOOL TO DATE(TRUE);
dateReturn_
dtReturn_8 := BOOL_TO_DT(TRUE);
dtReturn_80 := BOO\overline{L_TO}_DT(FALSE);
diReturn_9 := BOOL_TO_DINT(TRUE);
diReturn_90 := BOO\overline{L}_T\overline{O_DINT(FALSE);}
dwReturn_11 := BOOL_TO_DWORD(TRUE);
dwReturn_110 := BOO\overline{L_TO}_DWORD(FALSE);
iReturn_12 := BOOL_TO_INT(TRUE);
iReturn_120 := BOO\overline{L_TO_INT(FALSE);}
liReturn_13 := BOOL_TO_LINT(TRUE);
liReturn_130 := BOO\overline{L_TO__LINT(FALSE);}
lrReturn_14 := BOOL_TO_LREAL(TRUE);
lrReturn_-140 := BOO\overline{L}_TO_LREAL(FALSE);
lwReturn_15 := BOOL_TO_LWORD(TRUE);
lwReturn_-150 := BOO\overline{L}_T\overline{O}_LWORD (FALSE);
rReturn_16 := BOOL_TO_REAL(TRUE);
rReturn_160 := BOO\overline{L_TO_REAL(FALSE);}
siReturn_17 := BOOL_TO_SINT(TRUE);
siReturn_170 := BOO\overline{L_TO_SINT(FALSE);}
sReturn_18 := BOOL_TO_STRING(TRUE);
sReturn_180 := BOO\overline{L}_T\overline{O}_STRING(FALSE);
timReturn_20 := BOOL_TO_TIME(TRUE);
timReturn_200 := BOO\overline{L_TO_TIME (FALSE);}
todReturn_21 := BOOL_TO_TOD(TRUE);
todReturn_210 := BOO\overline{L}_T\overline{O}_TOD(FALSE);
udiReturn_22 := BOOL_TO_UDINT(TRUE);
udiReturn_220 := BOOL__TO_UDINT(FALSE);
uiReturn_23 := BOOL_TO_UINT(TRUE);
uiReturn_230 := BOO\overline{L_TO}_UINT(FALSE);
uliReturn_24 := BOOL_TO_ULINT(TRUE);
uliReturn_240 := BOO\overline{L_TO__ULINT(FALSE);}
usiReturn_25 := BOOL_TO_USINT(TRUE);
usiReturn_250 := BOO\overline{L}_TO_USINT(FALSE);
```



See also

- $乡$ "Type conversion operators" on page 2132
- \& Chapter 6.4.1.20.3.36 "Overloading" on page 2152
- $\quad$ \& Chapter 6.4.1.20.3.38 "Integer Conversion" on page 2159
- «y Chapter 6.4.1.20.3.39 "Floating-Point Number Conversion" on page 2171
- $\quad 4$ Chapter 6.4.1.20.3.40 "String Conversion" on page 2174
- $\Leftrightarrow$ Chapter 6.4.1.20.3.42 "Date and Time Conversion" on page 2187
- $\Leftrightarrow$ Chapter 6.4.1.20.3.41 "Time Conversion" on page 2182


## Integer Conversion

## NOTICE!

If the operand value for a type conversion operator is outside of the value range of the target data type, then the result output depends on the processor type and is therefore undefined. This is the case, for example, when a negative operand value is converted from LREAL to the target data type UINT.

Information can be lost when converting from larger data types to smaller data types.

The operators convert an integer value into the specified data types and return this type-converted value. If the number to be converted exceeds the range limit, then the first bytes of the number are ignored.

## Call syntax

```
<variable name> := <integer conversion type operator> ( <integer
operand> );
<integer conversion type operator> = <integer data type> _TO_ <data
type>
<integer operand> = <variable name> | <literal>
<integer data type> =
    UXINT | __XINT | __XWORD | BIT | BYTE | DINT | DWORD | INT | LINT |
\overline{LW}ORD | SI\overline{NT | UDINT | UINT | ULINT | USINT | WORD}
```


## Operators

```
    UXINT_TO___XINT
    UXINT TO XWORD
__UXINT-TO-BIT
__UXINT_TO_BOOL
    UXINT-TO_BYTE
__UXINT_TO_DATE
__UXINT_TO_DINT
_UUXINT-TO-DT
__UXINT_TO_DWORD
__UXINT_TO_INT
__UXINT_TO_LDATE
__UXINT_TO_LDT
__UXINT_TO_LINT
__UXINT_TO_LREAL
__UXINT_TO_LTIME
__UXINT_TO_LTOD
__UXINT_TO_LWORD
__UXINT_TO_REAL
__UXINT_TO_SINT
__UXINT_TO_STRING
```

```
    UXINT TO TIME
    UXINT TO_TOD
UXINT TO UDINT
UXINT TO UINT
UXINT_TO_ULINT
__UXINT_TO_USINT
__UXINT_TO_WORD
__UXINT_TO_WSTRING
_XINT_TO__UXINT
__XINT_TO___XWORD
_XINT TO_BIT
__XINT_TO-BOOL
__XINT_TO_BYTE
XINT TO DATE
__XINT_TO-DINT
__XINT_TO_DT
XINT TO DWORD
_XINT-TO-INT
__XINT_TO_LDATE
XINT TO LDT
-_XINT_TO-LINT
__XINT_TO_LREAL
XINT TO LTIME
-XINT TO- LTOD
__XINT_TO_LWORD
XINT TO REAL
__XINT-TO-SINT
__XINT_TO_STRING
__XINT_TO_TIME
XINT-TO-
__XINT_TO_UDINT
__XINT_TO_UINT
XINT-TO_ULINT
__XINT_TO_USINT
XINT TO WORD
_-XINT_TO_WSTRING
__XWORD_TO_UXINT
_XWORD_TO_XINT
__XWORD_TO_BIT
__XWORD_TO_BOOL
XWORD TO- BYTE
XWORD_TO_DATE
__XWORD_TO_DINT
XWORD TO DT
XWORD_TO_DWORD
_XWORD_TO_INT
XWORD_TO_LDATE
__XWORD_TO_LDT
__XWORD_TO_LINT
__XWORD_TO_LREAL
__XWORD_TO_LTIME
___XWORD_TO_LTOD
_XWORD_TO_LWORD
__XWORD_TO_REAL
__XWORD_TO_SINT
__XWORD_TO_STRING
__XWORD_TO_TIME
__XWORD_TO_TOD
__XWORD_TO_UDINT
__XWORD_TO_UINT
__XWORD_TO_ULINT
__XWORD_TO_USINT
__XWORD_TO_WORD
```

```
XWORD_TO_WSTRING
BIT TO UXINT
BIT TO- XINT
BIT_TO___XWORD
BIT TO BOOL
BIT 'TO-BYTE
BIT_TO_DATE
BIT TO DINT
BIT_TO_DT
BIT_TO_DWORD
BIT_TO_INT
BIT_TO_LDATE
BIT_TO_LDT
BIT TO LINT
BIT_TO_LREAL
BIT_TO_LTIME
BIT TO LTOD
BIT-TO-LWORD
BIT_TO_REAL
BIT_TO_SINT
BIT_TO_STRING
BIT_TO_TIME
BIT TO_TOD
BIT TO-UDINT
BIT_TO_UINT
BIT_TO_ULINT
BIT-TO-USINT
BIT_TO_WORD
BIT_TO_WSTRING
BYTE_TO____UXINT
BYTE_TO_XINT
BYTE_TO___XWORD
BYTE_TO_BOOL
BYTE_TO_BIT
BYTE_TO_DATE
BYTE_TO_DINT
BYTE_TO_DT
BYTE TO DWORD
BYTE_TO_INT
BYTE_TO_LDATE
BYTE_TO_LDT
BYTE_TO_LINT
BYTE_TO_LREAL
BYTE TO LTIME
BYTE-TO_LTOD
BYTE_TO_LWORD
BYTE TO REAL
BYTE-TO-SINT
BYTE_TO_STRING
BYTE TO TIME
BYTE-TO-TOD
BYTE_TO_UDINT
BYTE_TO_UINT
BYTE_TO_ULINT
BYTE_TO_USINT
BYTE TO_WORD
BYTE_TO_WSTRING
DINT_TO
                UXINT
DINT_TO__XINT
DINT TO XWORD
DINT-TO-BOOL
DINT_TO_BIT
```

```
DINT TO_BYTE
DINT TO_DATE
DINT TO DT
DINT_TO_DWORD
DINT_TO_INT
DINT_TO_LDATE
DINT_TO_LDT
DINT_TO_LINT
DINT TO LREAL
DINT_TO_LTIME
DINT_TO_LTOD
DINT TO LWORD
DINT_TO-REAL
DINT_TO_SINT
DINT TO STRING
DINT_TO-TIME
DINT_TO_TOD
DINT TO UDINT
DINT TO-UINT
DINT_TO_ULINT
DINT TO USINT
DINT-TO-WORD
DINT_TO_WSTRING
DWORD_TO__UXINT
DWORD_TO___XINT
DWORD_TO__XWORD
DWORD_TO-BIT
DWORD_TO_BOOL
DWORD_TO_BYTE
DWORD_TO_DATE
DWORD_TO_DINT
DWORD_TO_DT
DWORD_TO_INT
DWORD_TO_LDATE
DWORD_TO_LDT
DWORD_TO_LINT
DWORD_TO_LREAL
DWORD_TO_LTIME
DWORD_TO_LTOD
DWORD_TO_LWORD
DWORD_TO_REAL
DWORD TO- SINT
DWORD_TO_STRING
DWORD_TO_TIME
DWORD TO_TOD
DWORD_TO-UDINT
DWORD_TO_UINT
DWORD TO_ULINT
DWORD_TO_USINT
DWORD_TO_WORD
DWORD_TO_WSTRING
INT_TO___UXINT
INT TO XINT
INT-TO XWORD
INT_TO_BIT
INT TO BOOL
INT_TO-BYTE
INT_TO_DATE
INT TO DINT
INT_TO-DT
INT_TO_DWORD
INT_TO_LDATE
INT_TO_LDT
```

```
INT TO_LINT
INT_TO_LREAL
INT_TO_LTIME
INT_TO_LTOD
INT_TO_LWORD
INT_TO_REAL
INT_TO_SINT
INT_TO_STRING
INT_TO_TIME
INT_TO-TOD
INT_TO_UDINT
INT_TO_UINT
INT_-TO_ULINT
INT_TO_USINT
INT-TO-WORD
INT_-TO_WSTRING
LINT_TO___UXINT
LINT_TO_XINT
LINT_TO___XWORD
LINT_TO_BIT
LINT_TO-BOOL
LINT_TO_BYTE
LINT TO DATE
LINT TO-DINT
LINT_TO_DT
LINT TO DWORD
LINT-TO-INT
LINT_TO_LDATE
LINT TO LDT
LINT 'TO-
LINT_TO_LTIME
LINT TO_LTOD
LINT-TO-
LINT_TO_REAL
LINT_TO_SINT
LINT 'TO-STRING
LINT_TO_TIME
LINT_TO_TOD
LINT_TO_UDINT
LINT-TO-UINT
LINT_TO_ULINT
LINT_TO_USINT
LINT_TO_WORD
LINT_TO_WSTRING
LWORD_TO___UXINT
LWORD_TO___XINT
LWORD_TO__XWORD
LWORD TO BIT
LWORD_TO_BOOL
LWORD TO BYTE
LWORD_TO_DATE
LWORD_TO_DINT
LWORD TO DT
LWORD_TO_DWORD
LWORD_TO_INT
LWORD TO LDATE
LWORD_TO_LDT
LWORD_TO_LINT
LWORD TO- LREAL
LWORD_TO_LTIME
LWORD_TO_LTOD
LWORD_TO_REAL
LWORD_TO_SINT
```

```
LWORD_TO_STRING
LWORD_TO_TIME
LWORD TO TOD
LWORD_TO-UDINT
LWORD_TO_UINT
LWORD TO ULINT
LWORD_TO-USINT
LWORD_TO_WORD
LWORD_TO_WSTRING
SINT_TO___UXINT
SINT TO XINT
SINT_TO_XWORD
SINT TO BIT
SINT TO BOOL
SINT TO-BYTE
SINT_TO_DATE
SINT TO DINT
SINT-TO-DT
SINT_TO_DWORD
SINT TO INT
SINT-TO-LDATE
SINT_TO_LDT
SINT TO LINT
SINT TO- LREAL
SINT_TO_LTIME
SINT TO LTOD
SINT_TO_LWORD
SINT_TO_REAL
SINT TO STRING
SINT TO-TIME
SINT_TO_TOD
SINT TO UDINT
SINT TO-UINT
SINT_TO_ULINT
SINT TO USINT
SINT TO WORD
SINT_TO_WSTRING
UDINT TO UXINT
UDINT TO_XINT
UDINT_TO___XWORD
UDINT TO BIT
UDINT TO-BOOL
UDINT_TO_BYTE
UDINT TO DATE
UDINT_TO_DINT
UDINT_TO_DT
UDINT TO DWORD
UDINT_TO_INT
UDINT_TO_LDATE
UDINT TO LDT
UDINT-TO-
UDINT_TO_LREAL
UDINT TO LTIME
UDINT TO_LTOD
UDINT_TO_LWORD
UDINT TO_REAL
UDINT_TO_SINT
UDINT_TO_STRING
UDINT TO TIME
UDINT TO- TOD
UDINT_TO_UINT
UDINT TO ULINT
UDINT_TO_-USINT
```

```
UDINT_TO_WORD
UDINT_TO_WSTRING
UINT TO UXINT
UINT TO XINT
UINT TO XWORD
UINT_TO_BIT
UINT_TO_BOOL
UINT TO BYTE
UINT-TO-DATE
UINT_TO_DINT
UINT TO_DT
UINT_TO_DWORD
UINT_TO_INT
UINT TO LDATE
UINT_TO-LDT
UINT_TO_LINT
UINT TO_LREAL
UINT TO-LTIME
UINT_TO_LTOD
UINT TO LWORD
UINT-TO-REAL
UINT_TO_SINT
UINT TO STRING
UINT_TO_TIME
UINT_TO_TOD
UINT TO UDINT
UINT-TO-ULINT
UINT_TO_USINT
UINT TO_WORD
UINT_TO_WSTRING
ULINT_TO__UXINT
ULINT TO- XINT
ULINT_TO__XWORD
ULINT_TO_BIT
ULINT -TO-BOOL
ULINT_TO_BYTE
ULINT_TO_DATE
ULINT TO DINT
ULINT_TO_DT
ULINT TO_DWORD
ULINT_TO_INT
ULINT_TO_LDATE
ULINT_TO_LDT
ULINT TO LINT
ULINT-TO_LREAL
ULINT_TO_LTIME
ULINT TO LTOD
ULINT_TO_LWORD
ULINT_TO_REAL
ULINT TO SINT
ULINT_TO_STRING
ULINT_TO_TIME
ULINT TO TOD
ULINT_TO_UDINT
ULINT_TO_UINT
ULINT TO_USINT
ULINT_TO_WORD
ULINT_TO_WSTRING
USINT_TO__XINT
USINT_TO___XINT
USINT_TO___XWORD
USINT_TO-\BIT
```

```
USINT_TO_BOOL
USINT_TO_BYTE
USINT TO DATE
USINT-TO-DINT
USINT_TO_DT
USINT TO_DWORD
USINT-TO_INT
USINT_TO_LDATE
USINT TO LDT
USINT_TO_LINT
USINT_TO_LREAL
USINT TO_LTIME
USINT TO-
USINT_TO_LWORD
USINT TO REAL
USINT-TO-SINT
USINT_TO_STRING
USINT TO TIME
USINT-TO-TOD
USINT_TO_UDINT
USINT TO UINT
USINT-}\mp@subsup{}{}{-}\mp@subsup{\textrm{TO}}{}{-}\mathrm{ ULINT
USINT_TO_WORD
USINT_TO_WSTRING
WORD_TO___XINT
WORD TO XINT
WORD_TO_X_XOORD
WORD TO BIT
WORD TO BOOL
WORD -TO-BYTE
WORD_TO_DATE
WORD TO DINT
WORD_TO_DT
WORD_TO_DWORD
WORD TO INT
WORD_TO-LDATE
WORD_TO_LDT
WORD_TO_LINT
WORD TO LREAL
WORD_TO_LTIME
WORD TO LTOD
WORD 'TO- LWORD
WORD TO_REAL
WORD TO SINT
WORD_TO_STRING
WORD_TO-TIME
WORD TO TOD
WORD TO UDINT
WORD -TO-UINT
WORD_TO_ULINT
WORD_TO_USINT
WORD_TO_WSTRING
```


## Converting to a string

## NOTICE!

## String manipulation when converting to STRING or WSTRING

When converting the type to STRING or WSTRING, the typed value is leftaligned as a character string and truncated if it is too long. Therefore, declare the return variable for the type conversion operators <>_TO_STRING and <>_TO_WSTRING long enough that the character string has enough space without any manipulation.

The operators that convert a value into a character string of type STRING or WSTRING require an operand that matches the target data type.


## Examples

ST implemen- When a larger data type is converted to a smaller data type, the more high-order (front) tation language bytes are truncated. When a smaller data type is converted to a larger data type, the more high-order bytes filled with zeros.

```
FUNCTION_BLOCK FB_ConvertIntegersFromInt
VAR_INPUT
END_VAR
VAR OUTPUT
END_VAR
VAR
    uxiReturn: UXINT;
    xiReturn: XINT;
    xwReturn: __XWORD;
    bitReturn: BIT;
    xReturn: BOOL;
    bReturn: BYTE;
    dateReturn: DATE;
    diReturn: DINT;
    dtReturn: DATE_AND_TIME;
    dwReturn: DWORD;
    liReturn: LINT;
    lrReturn: LREAL;
    lwReturn: LWORD;
    siReturn: SINT;
    sReturn: STRING;
    timReturn: TIME;
    todReturn: TIME OF DAY;
    udiReturn: UDINT;
    uiReturn: UINT;
    usiReturn: USINT;
    wReturn: WORD;
    wsReturn: WSTRING;
    uliReturn: ULINT;
END_VAR
uxiReturn := INT TO UXINT(127);
xiReturn := INT_TO__XINT(127);
xwReturn := INT_TO__XWORD(127);
bitReturn := INT T\overline{O BIT(127);}
xReturn := INT_TO_BOOL(127);
bReturn := INT_TO_BYTE(127);
dateReturn := \overline{INT TO DATE(127);}
diReturn := INT_T\overline{O}}\textrm{D}\overline{I}NT(127)
dtReturn := INT_TO_DT(127);
dwReturn := INT TO DWORD(127);
liReturn := INT_TO_LINT(127);
lrReturn := INT_TO_LREAL(127);
lwReturn := INT TO LWORD(127);
siReturn := INT TO- SINT(127);
sReturn := INT_TO_STRING(127);
timReturn := INTT TO TIME (127);
todReturn := INT_TO_TOD(127);
udiReturn := INT_TO_UDINT(127);
uiReturn := INT TO UINT(127);
uliReturn := IN\overline{T}T\overline{O}\mathrm{ ULINT(127);}
usiReturn := INT_TO_USINT(127);
wReturn := INT TO WORD(127);
wsReturn := INT_T\overline{O}WSTRING(127);
FUNCTION BLOCK FB ConvertIntegersToInt
VAR INPU\overline{T}
END_VAR
VAR_OUTPUT
```

```
END VAR
VAR
    iReturn_uxi: INT;
    iReturn_xi: INT;
    iReturn_xw: INT;
    iReturn_bit: INT;
    iReturn_bool: INT;
    iReturn_b: INT;
    iReturn_d: INT;
    iReturn_di: INT;
    iReturn_dt: INT;
    iReturn_dw: INT;
    iReturn_li: INT;
    iReturn_lr: INT;
    iReturn_lw: INT;
    iReturn_r: INT;
    iReturn_si: INT;
    iReturn_s: INT;
    iReturn_tim: INT;
    iReturn_tod: INT;
    iReturn_tod_0: INT;
    iReturn_udi: INT;
    iReturn_ui: INT;
    iReturn_uli: INT;
    iReturn_usi: INT;
    iReturn_w: INT;
    iReturn_ws: INT;
END_VAR
iReturn_uxi := __UXINT_TO_INT(18446744073709551615);
iReturn -xi := \overline{XINT T\overline{O}}\mathbf{INNT(9223372036854775807);}
iReturn_xw := __XWORD_T\overline{O_INT(16#FFFF_FFFF_FFFF_FFFF);}
iReturn_bit := BIT_TO_INT(1);
iReturn_bool := BOŌL_TO_INT(TRUE);
iReturn_b := BYTE_TO_INT(2#1111_0000);
iReturn_d := DATE_TO_INT(DATE#2019-9-13);
iReturn_di := DIN\overline{T}_T\overline{O}_INT(2147483647);
iReturn_dt := DT_TO_INTT(DT#1979-1-1-00:00:00);
iReturn_dw := DWORD_TO_INT(16#FFFF_FFFF);
// iRetūrn_i := INT_TO_<>(iData_12);
```



```
iReturn_lr := LREAL_TO_INT(1.7976931348623157E+30);
iReturn_lw := LWORD_TO_INT(16#FFFF_FFFF_FFFF_FFFF);
iReturn_r := REAL_TO_INT(3.402823E+38);
iReturn_si := SINT_TO-_INT(127);
iReturn_S := STRING_TO_INT('127');
iReturn_tim := TIME_TO_INT(T#49D17H2M47S295MS);
iReturn_tod := TOD_TO_INT(TOD#23:59:59.999);
iReturn_tod_0 := TO\overline{D_TO_INT(TOD#1:1:1.001);}
iReturn_udi }\mp@subsup{\mp@code{O}}{= UDINT_TO_INT(4294967295);}{
iReturn_ui := UINT_TO_INNT(65535);
iReturn_uli := ULIN̄T_\overline{TO_INT(18446744073709551615);}
iReturn_usi := USINT_TO_INT(255);
iReturn_w := WORD_TO_INT(16#FFFF);
iReturn_ws := WSTRING_TO_INT("1234567890");
PROGRAM PLC_PRG
VAR
    fbConvertIntegersFromInt : FB_ConvertIntegersFromInt;
    fbConvertIntegersToInt : FB_ConvertIntegersToInt;
END_VAR
fbConvertIntegersFromInt();
fbConvertIntegersToInt();
```



See also

- " "Type conversion operators" on page 2132
- ② Chapter 6.4.1.20.3.37 "Boolean Conversion" on page 2154
- ${ }^{*}$ Chapter 6.4.1.20.3.36 "Overloading" on page 2152
- \& Chapter 6.4.1.20.3.39 "Floating-Point Number Conversion" on page 2171
- \& Chapter 6.4.1.20.3.40 "String Conversion" on page 2174
- $\&$ Chapter 6.4.1.20.3.42 "Date and Time Conversion" on page 2187
-     * Chapter 6.4.1.20.3.41 "Time Conversion" on page 2182


## Floating-Point Number Conversion

## NOTICE!

If the operand value for a type conversion operator is outside of the value range of the target data type, then the result output depends on the processor type and is therefore undefined. This is the case, for example, when a negative operand value is converted from LREAL to the target data type UINT.

Information can be lost when converting from larger data types to smaller data types.

## NOTICE!

If the floating-point number is within the value range of the target data type, then the conversion operates the same way on all systems.

## NOTICE!

If the floating-point number to be converted exceeds the range limit, then the first bytes of the number are ignored.

The operators convert a floating-point number into the specified data types and return a typeconverted value. If applicable, the conversion is rounded.

## Call

Syntax

```
<variable name> := <floating-point conversion operator> ( <floating-
point operand> );
<floating-point operand> = <variable name> | <literal>
<floating-point type> =
REAL |
LREAL
```

Operators

```
REAL_TO UXINT
REAL TO XINT
REAL-TO XWORD
REAL_TO_BIT
REAL-TO-BOOL
REAL TO- BYTE
REAL_TO_DATE
REAL_TO_DINT
REAL_TO_DT
REAL_TO_DWORD
REAL_TO_INT
REAL_TO-
REAL_TO_LREAL
REAL TO LTIME
REAL-TO-LWORD
REAL_TO_SINT
```

```
REAL_TO_STRING
REAL_TO_TIME
REAL TO TOD
REAL-}\mp@subsup{\textrm{TO}}{}{-}\mathrm{ UDINT
REAL_TO_UINT
REAL_TO_ULINT
REAL_TO_USINT
REAL_TO_WORD
REAL_TO_WSTRING
LREAL_TO___UXINT
LREAL TO XINT
LREAL TO XWORD
LREAL TO BIT
LREAL TO BOOL
LREAL_TO_BYTE
LREAL_TO_DATE
LREAL TO DINT
LREAL-TO-DT
LREAL_TO_DWORD
LREAL_TO_INT
LREAL_TO-LINT
LREAL_TO_LTIME
LREAL TO LWORD
LREAL - TO- REAL
LREAL_TO_SINT
LREAL TO STRING
LREAL_TO-TIME
LREAL_TO_TOD
LREAL_TO_UDINT
LREAL-}\mp@subsup{}{}{-}\mp@subsup{}{}{-}\mp@subsup{}{}{-}\mathrm{ UINT
LREAL_TO_ULINT
LREAL_TO_USINT
LREAL_TO_WORD
LREAL_TO_WSTRING
```

Rounding When converting to an integer, the operand is rounded up or down to an integer value. For 1 to 4 after the decimal point, the number is rounded down. For 5 to 9 , the number is rounded up. Then the rounded number is converted to the specified integer type. If the rounded value is outside of the integer value range, then an undefined, target system-dependent value is returned. An exception error is also possible then.

## NOTICE!

The rounding logic for borderline cases depends on the target system or the FPU (Floating Point Unit) of the target system. For example, a value of -1.5 can be converted differently on different controllers.

To program target system-independent code, you have to catch value range overflows across the application.

Converting to a string

## NOTICE!

String manipulation when converting to STRING or WSTRING
When converting the type to STRING or WSTRING, the typed value is leftaligned as a character string and truncated if it is too long. Therefore, declare the return variable for the type conversion operators <>_TO_STRING and <>_TO_WSTRING long enough that the character string has enough space without any manipulation.

For a floating-point number conversion to a string, the number of decimal places of the mantissa is limited to 6 . If the number is $<1$, then the mantissa is $1<=m<10$. If the mantissa has more digits after the comma, then it is rounded to the 6th digit and then converted.
The string variable may also be declared too short for the return value. In this case, the return string is truncated on the right.


## Examples

## ST implementation language

## FB_ConvertFromReal $\times$

Device.App_ConvertFloatingPoints.PLC_PRG.fbConvertFromReal

```
xwReturn 9223372036854775808 := REAL_TO___XWORD (3.402823E+38);
xReturn TRUE := REAL_TO_BOOL (3.402823E+38);
bReturn 0 := REAL_TO_BYTE (3.402823E+38);
dateReturn\D#1970-1-1 := REAL_TO_DATE (3.402823E+38);
diReturn 0 := REAL_TO_DINT (3.402823E+38);
dtReturn DT#1970-1-1-0:0:0 := REAL_TO_DT (3.402823E+38);
dwReturn 0 := REAL TO DWORD (3.402823E+38);
iReturn\-2 := REAL_TO_INT (-1.5);
iReturn_0 -1 := REAL_TO_INT (-1.4);
iReturn_1 1 := REAL_TO_INT (1.4);
iReturn_2 2 := REAL TO INT (1.5);
iReturn_3 12346 := REAL_TO_INT(1.234567890123456789E4);
liReturn -9223372036854775808 := REAL_TO_LINT (3.402823E+38);
lrReturn 3.4E+38 \ := REAL_TO_LREAL (3.402823E+38);
lwReturn 9223372036854775808 := REAL_TO_LWORD (3.402823E+38);
siReturn 0 := REAL_TO_SINT (3.402823E+38);
sReturn 3.402823E3 \ := REAL_TO_STRING (3.402823E+38);
timReturn\ T#0ms := REAL TO TIME (3.402823E+38);
todReturn TOD#0:0:0 := REAL TO TOD (3.402823E+38);
udiReturn 0 := REAL_TO_UDINT (3.402823E+38);
uiReturn\0 := REAL_TO_UINT(3.402823E+38);
uliReturn 9223372036854775808 := REAL_TO_ULINT(3.402823E+38);
usiReturn\0 := REAL_TO_USINT (3.402823E+38);
wReturn 0 := REAL_TO_WORD (3.402823E+38);
wsReturn "3.402823E3 \ := REAL_TO_WSTRING(3.402823E+38);
```

See also

- " "Type conversion operators" on page 2132
- ② Chapter 6.4.1.20.3.37 "Boolean Conversion" on page 2154
- ${ }^{4}$ Chapter 6.4.1.20.3.36 "Overloading" on page 2152
- ${ }^{\text {® }}$ Chapter 6.4.1.20.3.38 "Integer Conversion" on page 2159
- $\Leftrightarrow$ Chapter 6.4.1.20.3.40 "String Conversion" on page 2174
- $\Leftrightarrow$ Chapter 6.4.1.20.3.42 "Date and Time Conversion" on page 2187
- $\Leftrightarrow$ Chapter 6.4.1.20.3.41 "Time Conversion" on page 2182


## String Conversion

## NOTICE!

If the operand value for a type conversion operator is outside of the value range of the target data type, then the result output depends on the processor type and is therefore undefined. This is the case, for example, when a negative operand value is converted from LREAL to the target data type UINT.

Information can be lost when converting from larger data types to smaller data types.

The operators convert a character string (STRING or WSTRING) into the specified target data type and return a type-converted value.

A conversion with a meaningful result is only possible when the operand matches the target data type according to the IEC 61131-3 standard. This is the case if the value of the operand corresponds to a valid constant (literal) of the target data type.
Convertible strings contain:

- Number with type prefix (example: '16\#FFFFFFFF')
- Number with grouping characters (example: '2\#1111_1111')

Note: The international weight and measure grouping character (thin space) is not accepted. Only the underscore is accepted.

- Floating-point number, also in exponential notation (example: '9.876' or '1.2E-34') Note: Floating-point numbers are not convertible. The comma is treated and truncated like a following character.
- Time, time of day, and date specification with prefix and size (example: 'T\#2h', 'DT\#2019-9-9-12:30:30.9')
- Infinite values (example: '1.7E+400')
- Additional character after a number (example: ' 2 m ' or ' 3.14 '). These are truncated. Additional characters before a number are not permitted.
- Spaces before (example: ' 3.14')


## Call syntax

```
<variable name> := <string to operator> ( <operand> );
<operand> = <variable name> | <literal>
```


## Operators

```
STRING_TO__UXINT
STRING TO-- XINT
STRING_TO___XWORD
STRING TO BIT
STRING-TO_BOOL
STRING_TO_BYTE
STRING_TO_DATE
STRING-TO-DINT
STRING_TO_DT
STRING_TO_DWORD
STRING TO INT
STRING_TO_LDATE
STRING TO LDT
STRING_TO_LINT
STRING_TO_LREAL
STRING_TO_LTIME
STRING_TO_LWORD
STRING_TO_LTIME
STRING_TO_LTOD
STRING TO REAL
STRING-TO-SINT
STRING_TO_TIME
STRING TO TOD
STRING-TO-UDINT
STRING_TO_UINT
STRING TO ULINT
STRING-TO-USINT
STRING_TO_WORD
STRING_TO_WSTRING
WSTRING_TO___UXINT
WSTRING-TO XINT
WSTRING_TO___XWORD
```

```
WSTRING TO BIT
WSTRING TO BOOL
WSTRING TO BYTE
WSTRING -TO-DATE
WSTRING_TO_DINT
WSTRING TO DT
WSTRING_TO-DWORD
WSTRING TO_INT
WSTRING TO LDATE
WSTRING_TO-_LDT
WSTRING_TO_LINT
WSTRING TO LREAL
WSTRING_TO_LTIME
WSTRING_TO_LTOD
WSTRING TO LWORD
WSTRING_TO_LTIME
WSTRING_TO_REAL
WSTRING TO SINT
WSTRING_TO-STRING
WSTRING_TO_TIME
WSTRING TO TOD
WSTRING TO-UDINT
WSTRING_TO-UINT
WSTRING TO ULINT
WSTRING TO-
WSTRING_TO_STRING
WSTRING_TO_WORD
```

Converting to a Operator STRING_TO_BOOL: A value of TRUE is returned only if the operand value is 'TRUE ' Boolean value or 'true'. On the other hand, FALSE is returned for 'True'.
Operator WSTRING_TO_BOOL: A value of TRUE is returned only if the operand value is "TRUE" or "true". On the other hand, FALSE is returned for "True".


## Examples

## ST implemen-

 tation language```
FUNCTION_BLOCK FB_ConvertStrings
VAR INPU\overline{T}
END_VAR
VAR OUTPUT
END_VAR
VAR
    xReturn 0: BOOL;
    xReturn_1: BOOL;
    dateReturn: DATE;
    dtReturn: DATE_AND_TIME;
    iReturn: INT;
    lrReturn: LREAL;
    lrReturn_0: LREAL;
    lwReturn` LWORD;
    lwReturn_0: LWORD;
    lwReturn_1: LWORD;
    ltReturn` LTIME;
    ltReturn_0: LTIME;
    ltReturn_1: LTIME;
    ltReturn}\mp@subsup{}{}{-}2: LTIME;
    rReturn: REAL;
    rReturn_0: REAL;
    timReturn: TIME;
    timReturn0: TIME;
    timReturn1: TIME;
    timReturn2: TIME;
    todReturn: TIME_OF_DAY;
    todReturn0: TIME_OF_DAY;
    todReturn1: TIME-OF DAY;
    todReturn2: TIME_OF_DAY;
    uliReurn: ULINT;
    uliReurn_0: ULINT;
    uliReurn_1: ULINT;
    wReturn: WORD;
    wReturn 0: WORD;
    wReturn_1: WORD;
    wstrReturn: WSTRING;
    wstrReturn 0: WSTRING;
END_VAR
xReturn 0 := STRING TO BOOL('FALSE');
xReturn_1 := STRING_TO_BOOL('TRUE');
dateRetūrn := STRING}T\overline{O}\mathrm{ DATE('DATE#2019-9-9');
dtReturn := STRING_TO_DT('DT#2019-9-9-1:1:1.1');
iReturn := STRING TO INT('123abc');
lrReturn := STRIN\overline{G T\overline{O}}\mathrm{ LREAL('4.94E-323');}
lrReturn_0 := STRING_TO_LREAL('1.7E+308');
lwReturn}\mp@subsup{}{}{-}:= STRING T\overline{O} L\overline{W}ORD('16#FFFF FFFF FFFF FFFF');
lwReturn_0 := STRIN\overline{N_TO_LWORD('16#0123456789AB\overline{CDEF');}};=,
lwReturn_1 := STRING_TO_LWORD('16#0123456789ABCDEF');
ltReturn :=
STRING TO LTIME('LTIME#213503d23h34m33s709ms551us615ns');
ltReturn_\overline{0}}:= STRING_TO_LTIME('LTIME#0ns');
ltReturn_1 := STRING_TO_LTIME('LTIME#1ms');
ltReturn_2 := STRING_TO_LTIME('LTIME#2s');
rReturn := STRING_TO_REAL('6.543e21');
rReturn_0 := STRIN̄G_TO_REAL('1.234');
timReturn := STRING TO- TIME('T#5d4h3m2s');
timReturn0 := STRING_TO_TIME('TIME#1s');
timReturn1 := STRING TO TIME('1s');
```

```
timReturn2 := STRING_TO_TIME('TIME#5s');
todReturn := STRING_TO_TOD('TOD#12:0:0.1');
todReturn0 := STRIN\overline{G_TO_TOD('TOD#0:0:0.0');}
todReturn1 := STRING_TO_TOD('20:15');
todReturn2 := STRING_TO_TOD('TOD#20:15');
uliReurn := STRING TO UL̄INT('18446744073709551615');
uliReurn_0 := STRIN̄G_\overline{TO_ULINT('1');}
uliReurn_1 := STRING_TO_ULINT('0');
wReturn := STRING_TO_WORD('16#FFFF_0000');
wReturn_0 := STRIN\overline{NG_TO_WORD('34abc'');}
wReturn_1 := STRING_TO_WORD('16#34abc');
wstrRetūrn := STRIN\overline{G}}T\overline{O}\mathrm{ WSTRING('Hello World!');
wstrReturn_0 := STRIN̄G_\overline{TO_WSTRING('123456789');}
```

```
Device.App_ConvertStrings.PLC_PRG.fbConvertStrings
    xReturn_0FALSE := STRING_TO_BOOL('FALSE');
    xReturn_1\TRUE := STRING_TO_BOOL('TRUE');
    dateReturn D#2019-9-9 := STRING_TO_DATE ('DATE#2019-9-9');
    dtReturn DT#2019-9.9-1:1:1 \= STRING_TO_DT('DI#2019-9-9-1:1:1.1');
    iReturn 123 := STRING_TO INT('123abc');
    - lrReturn4.94E-323 \:= STRING_TO_LREAL ('4.94E-323');
    O IrReturn_0 1.7E+308 \ := STRING_TO_LREAL('1.7E+308');
    - lwReturn 18446744073709551615 := STRING_TO_LWORD('16#FFFF_FFFF_FFFF_FFFF');
    lwReturn_0 81985529216486895 := STRING_TO_LWORD('16#0123456789ABCDEF');
    - lwReturn_1 81985529216486895 := STRING_TO_LWORD('16#0123456789ABCDEF');
    - ltReturn LTIME#213503d23h34m33s709ms551us615ns := STRING TO LTTME ('LTIME#213503d23h34m33s709ms551us6615ns');
    ltReturn_0 LTIME#0ns := STRING_TO_LTIME ('LIIME#0ns');
    ltReturn_1 LTIME#1ms := STRING_TO_LTIMR('LTIME#1ms');
```



```
    rReturn66.54E+21 \ := STRING_TO_REAL('6.543e21');
- rReturn_0 1.23 > := STRING TO REAL('1.234');
- timReturn प T#5d4h3m2s := STRING_TO_TIMR ('T#5d4h3m2s');
timReturn0\square T#1s := STRING_TO_TIME ('TIME#1s');
timReturn1 T#67ms := STRING_TO_TIME ('1s');
    timReturn2 T##s \= STRING_TO_TIME ('TIME#5s');
    todReturn TOD#12:0:0.100 := STRING_TO_TOD('TOD#12:0:0.1');
    todReturn0 TOD#0:0:0 := STRING_TO_TOD ('IOD#0:0:0.0');
LodReturn1 TOD#0:0:0 := STRING_TO_TOD ('20:15');
todReturn2 TOD#20:15:0 := STRING_TO_TOD ('TOD#20:15');
uliReurn 18446744073709551615 := STRING_TO_ULINT('18446744073709551615');
    uliReurn_0 \ 1 := STRING_TO_ULINT('1');
    uliReurn_1 0 := STRING TO ULINT('0');
    wReturn 0 := STRING_TO_WORD('16#FFFF_0000');
    wReturn_0 34 := STRING_TO_WORD ('34abc');
    wReturn_1 19132 := STRING_TO_WORD('16#34abc');
    wstrReturn "HelloWorl \ := STRING_TO_WSTRING('Hello World!');
```



## WSTRING con

 version in ST```
FUNCTION_BLOCK FB_ConvertWstrings
VAR
    xReturn 0: BOOL;
    xReturn_1: BOOL;
    dateReturn: DATE;
    dtReturn: DATE_AND_TIME;
    iReturn: INT;
    lrReturn: LREAL;
    lrReturn_0: LREAL;
    lwReturn: LWORD;
    lwReturn_0: LWORD;
    lwReturn_1: LWORD;
    ltReturn: LTIME;
    ltReturn_0: LTIME;
    ltReturn 1: LTIME;
    ltReturn_2: LTIME;
    rReturn: REAL;
    rReturn 0: REAL;
    timReturn: TIME;
    timReturn0: TIME;
    timReturn1: TIME;
    timReturn2: TIME;
    todReturn: TIME OF DAY;
    todReturn0: TIME O\overline{F DAY;}
    todReturn1: TIME OF DAY;
    todReturn2: TIME_OF_DAY;
    uliReurn: ULINT;
    uliReurn 0: ULINT;
    uliReurn_1: ULINT;
    wReturn: WORD;
    wReturn 0: WORD;
    wReturn_1: WORD;
    wstrReturn: WSTRING;
    wstrReturn_0: WSTRING;
END_VAR
xReturn 0 := WSTRING TO BOOL("FALSE");
xReturn_1 := WSTRING_TO_BOOL("TRUE");
dateReturn := WSTRING_TO_DATE("DATE#2019-9-9");
dtReturn := WSTRING T\overline{O}}\overline{\mathrm{ TT ("DT#2019-9-9-1:1:1.1");}
iReturn := WSTRING \overline{TO_INT("123abc");}
lrReturn := WSTRING TO LREAL("4.94E-323");
lrReturn_0 := WSTRIN̄G_\overline{TO_LREAL("1.7E+308");}
lwReturn - := WSTRING T\overline{O}L\overline{W}ORD("16#FFFF FFFF FFFF FFFF");
```



```
lwReturn }\mp@subsup{}{}{-1}:= WSTRING - TO- LWORD("16#0123456789ABCDEF"); 
ltReturn :=
WSTRING TO LTIME("LTIME#213503d23h34m33s709ms551us615ns");
ltReturn}\mp@subsup{0}{}{-}:= WSTRING TO LTIME("LTIME#Ons");
ltReturn_1 := WSTRING_TO_LTIME("LTIME#1ms");
ltReturn_2 := WSTRING_TO_LTIME("LTIME#2s");
rReturn \overline{ = WSTRING TO- REĀL("6.543e21");}
rReturn_0 := WSTRING_TO_REAL("1.234");
timReturn := WSTRING TO TIME("T#5d4h3m2s");
timReturn0 := WSTRING T\overline{O}TIME("TIME#1s");
timReturn1 := WSTRING_TO_TIME("1s");
timReturn2 := WSTRING TO TIME("TIME#5s");
todReturn := WSTRING \overline{TO TOD("TOD#12:0:0.1");}
todReturn0 := WSTRING_TO_TOD("TOD#0:0:0.0");
todReturn1 := WSTRING_TO_TOD("20:15");
todReturn2 := WSTRING TO TOD("TOD#20:15");
```

```
uliReurn := WSTRING_TO_ULINT("18446744073709551615");
uliReurn_0 := WSTRIN\overline{G_TO_ULINT("1");}
uliReurn_1 := WSTRING_TO-ULINT("0");
wReturn \overline{:= WSTRING TO- WO\overline{RD("16#FFFF 0000");}}\mathbf{=}\mathrm{ ; }
wReturn_0 := WSTRING_TO_WORD("34abc");
wReturn_1 := WSTRING_TO_WORD("16#34abc");
```



FBD implemen-
tation language
FB_FbdConvertString $\mathbf{x}$
Device.App_ConvertStrings.PLC_PRG.fbFbdConvertString


See also

- $乡$ "Type conversion operators" on page 2132
- $\sum^{*}$ Chapter 6.4.1.20.3.37 "Boolean Conversion" on page 2154
- ${ }^{\bullet}$ Chapter 6.4.1.20.3.36 "Overloading" on page 2152
- \& Chapter 6.4.1.20.3.38 "Integer Conversion" on page 2159
- ⓨ Chapter 6.4.1.20.3.39 "Floating-Point Number Conversion" on page 2171
- $\Leftrightarrow$ Chapter 6.4.1.20.3.42 "Date and Time Conversion" on page 2187
- \& Chapter 6.4.1.20.3.41 "Time Conversion" on page 2182


## Time Conversion

## NOTICE!

If the operand value for a type conversion operator is outside of the value range of the target data type, then the result output depends on the processor type and is therefore undefined. This is the case, for example, when a negative operand value is converted from LREAL to the target data type UINT.

Information can be lost when converting from larger data types to smaller data types.

The operators convert time values (TIME or LIME) into the specified data types and return this type-converted value.

## Call syntax

```
<variable name> := <time conversion operator> ( <operand> );
<operand> = <variable name> | <literal>
```


## Operators

```
TIME TO UXINT
TIME TO XINT
TIME TO XWORD
TIME 'TO BIT
TIME_TO_BOOL
TIME_TO_BYTE
TIME TO DATE
TIME TO- DINT
TIME_TO_DT
TIME_TO_DWORD
TIME_TO_INT
TIME_TO_LDATE
TIME TO LDINT
TIME 'TO-
TIME_TO_LREAL
TIME 'TO_LTIME
TIME_TO_LTOD
TIME_TO_LWORD
TIME-TO-REAL
TIME_TO_SINT
TIME_TO_STRING
TIME TO-TOD
TIME -TO_UDINT
TIME_TO_UINT
TIME_TO_ULINT
TIME -TO-USINT
TIME_TO_WORD
TIME_TO_WSTRING
```



## Converting to The operator returns FALSE if and only if the operand value can be interpreted as "0". Boolean values

| xTime : = TIME_TO_BOOL (T\#OMS) ; | xTime = FALSE |
| :---: | :---: |
| xLongTime : = TIME_TO_BOOL (T\#ONS) ; | xLongTime $=$ FALSE |
| xTime : = TIME_TO_BOOL (T\#1MS) ; | xDate = TRUE |
| xLongTime : = TIME_TO_BOOL (T\#1NS) ; | xLongTime $=$ TRUE |

## Converting to a string

## NOTICE!

String manipulation when converting to STRING or WSTRING
When converting the type to STRING or WSTRING, the typed value is leftaligned as a character string and truncated if it is too long. Therefore, declare the return variable for the type conversion operators <>_TO_STRING and <>_TO_WSTRING long enough that the character string has enough space without any manipulation.

| sTime $:=$ TIME_TO_STRING(T\#OMS); | STime $=$ 'T\#OMS' |
| :--- | :--- |
| WSLongTime $:=$ <br> LTIME_TO_WSTRING(T\#OUS); | wsLongTime $=$ "T\#OUS" |

## Examples

## ST implemen-

 tation language```
FUNCTION_BLOCK FB_ConvertTimeAndDate
VAR INPU\overline{T}
END_VAR
VAR OUTPUT
END_VAR
VAR
    ltReturn 1: LTIME;
    lwReturn}\mp@subsup{}{}{-}2: LWORD
    rReturn_3: REAL;
    strReturn 4: STRING;
    timReturn-5: TIME;
    todReturn_6: TIME_OF_DAY;
    uliReurn_7: ULINT;
    wstrReturn_8: WSTRING;
    wstrReturn-80: WSTRING;
    uliReurn_70: ULINT;
    todReturn
    timReturn_50: TIME;
    strReturn 40: STRING;
    rReturn 3\overline{0}: REAL;
    lwReturn_20: LWORD;
    ltReturn_10: LTIME;
    ltReturn 11: LTIME;
    lwReturn_21: LWORD;
    rReturn_31: REAL;
    strReturn 41: STRING;
    timReturn_51: TIME;
    todRedurn_61: TIME_OF_DAY;
    uliReurn \overline{71: ULINT;}
    wstrReturn 81: WSTRING;
    ltReturn 12: LTIME;
    xReturn_\overline{9}: BOOL;
    xReturn_90: BOOL;
    xReturn_91: BOOL;
    xReturn 92: BOOL;
    dateReturn_6: DATE;
    timReturn 60: TIME;
    wReturn 6\overline{1}: WORD;
    todReturn_61: TIME_OF_DAY;
END_VAR
ltReturn_1 := DT TO LTIME(DT#2019-9-9-23:59:59);
ltReturn_10 := DT_TO_LTIME(DT#1970-1-1-0:0:0);
ltReturn 11 := DT TO LTIME(DT#1970-1-2-0:0:1);
ltReturn_12 := DT_TO_LTIME(DT#1970-1-3-12:30:30);
lwReturn 2 := TIME TO LWORD(T#5D4H2M3S2MS);
lwReturn_20 := TIME_TO_LWORD(T#ODOHOMOSOMS);
rReturn 3 := TIME TO REAL(T#5D4H2M3S2MS);
rReturn_30 := TIME_T\overline{O_REAL(T#ODOHOMOSOMS);}
strREturn 4 := TIME TO STRING(T#5D4H2M3S2MS);
strREturn_40 := TIME_T\overline{O}_STRING(T#ODOHOMOSOMS);
timReturn 5 := TOD TO TIME(TOD#23:59:59.999);
timReturn_50 := TO\overline{D_TO_TIME (TOD#0:0:0.000);}
timReturn_51 := TOD_TO_TIME (TOD#0:0:0.001);
```

```
dateReturn_6 := TOD_TO_DATE(TOD#23:59:59.999);
timReturn_60 := TOD_TO_TIME (TOD#0:0:0.000);
wReturn_61 := TOD_TO_WORD(TOD#0:0:0.001);
uliReurn_7 := DATE_TO_ULINT(D#2019-9-9);
uliReurn_70 := DATE\overline{E}T\overline{O}\mathrm{ ULINT(D#1970-1-1);}
uliReurn_71 := DATE_-TO_ULINT(D#1970-1-2);
wstrReturn_8 := DATE_TO_WSTRING(D#2019-9-9);
wstrReturn_80 := DATE_T\overline{O}WSTRING(D#1970-1-1);
wstrReturn_81 := DATE_TO_WSTRING(D#1970-1-2);
xReturn_9 := DATE TO_BOOL(D#2019-9-9);
xReturn_90 := DATE_TO_BOOL(D#1970-1-1);
xReturn_91 := DATE_TO_BOOL(D#1970-1-2);
xReturn_92 := DATE_TO_BOOL(D#1970-1-3);
```


## FB_ConvertTime $\mathbf{x}$

Device.App_ConvertTimes.PLC_PRG.fbConvertTime

```
ltReturn_1 LTMME#1d2h3m4s5ms := TIME_TO_LTIME (T#1D2H3M4S5MS);
ltReturn_10 LTIME#1d2h3m4s5ms := TIME_TO_LTTME (TIME#1D2H3M4S5MS);
ltReturn_11\square LTIME#0ns 
ltReturn_12\square LTIME#49d17h2m47s295ms := TIME_TO_LTIME (T#49D17H2M47S295MS);
lwReturn_2 \ 93784005 := TIME_TO LWORD (T#1D2H3M4S5MS);
lwReturn_20 प 0 := TTME_TO_L_LWORD (T#OMS);
lwReturn_21 4294967295 := TIME_TO_LWORD (T#4 9D17H2M47S295MS);
rReturn_3 9.38E+07 \ := TIME_TO_REAL(T#1D2H3M4S5MS);
rReturn_30 0 := TIME_TO_REAL (I#OMS);
rReturn_31 4.29E+09 \ := TIME_TO_REAL(T#49D17H2M47S295MS);
strREturn_4 T#1d2h3m4s : := TIME_TO_STRING(T#1D2H3M4S5MS);
strREturn_40 T#0ms' := TTME_TO_STRING (T#0MS);
strREturn_41 T#49d17h2m : := TIME_TO_STRING (T#49D17H2M47S295MS);
todRedurn_6 TOD#2:3:4.005 := TTME_TO_TOD (T#1D2H3M4S5MS);
todRedurn_60 TOD#0:0:0 := TTME_TO_TOD (T#0MS);
todRedurn_61 TOD#17:2:47.295 := TIME_TO_TOD (T#49D17H2M47S295MS);
uliReurn_7 93784005 := TIME_TO_ULINT(T#1D2H3M4S5MS);
uliReurn_70 प 0,= TIME_TO_ULINT (T#OMS);
uliReurn_71 4294967295 := TIME_TO_ULINT(I#49D17H2M47S295MS);
wstrReturn_8 T#1d2h3m4s \ := TIME_TO_WSTRING(T#1D2H3M4S5MS);
wstrReturn_80 T"T#ms" := TIME_TO_WSTRING (T#0MS);
wstrReturn_81 T#49d17h2m v := TIME_TO_WSTRING(T#49D17H2M47S295MS);
xReturn_9TRUE := TIME_TO_BOOL (I#1D2H3M4S5MS);
xReturn_90[FALSE := TIME_TO_BOOL (T#OMS);
xReturn_91 TRUE := TIME_TO_BOOL (I#1MS);
xReturn_92TRUE := TIME_TO_BOOL (T#49D17H2M47S295MS);
RETURN


See also
- \(乡\) "Type conversion operators" on page 2132
- \(\sum^{*}\) Chapter 6.4.1.20.3.37 "Boolean Conversion" on page 2154
- \({ }^{\bullet}\) Chapter 6.4.1.20.3.36 "Overloading" on page 2152
- \& Chapter 6.4.1.20.3.38 "Integer Conversion" on page 2159
- \#y Chapter 6.4.1.20.3.39 "Floating-Point Number Conversion" on page 2171
- \(\quad 4\) Chapter 6.4.1.20.3.40 "String Conversion" on page 2174
- \& Chapter 6.4.1.20.3.42 "Date and Time Conversion" on page 2187

\section*{Date and Time Conversion}

\section*{NOTICE!}

If the operand value for a type conversion operator is outside of the value range of the target data type, then the result output depends on the processor type and is therefore undefined. This is the case, for example, when a negative operand value is converted from LREAL to the target data type UINT.

Information can be lost when converting from larger data types to smaller data types.

The operators convert a date and time value into the specified data type and return a type-converted value.

\section*{Call syntax}
```

<variable name> := <date and time conversion operator> ( <operand> );
<operand> = <variable name> | <literal>

```

The data types DATE and DT use the same memory format internally and are stored as DWORD. The resolution for DATE is 1 day. The resolution for \(D T\) is 1 second. Both begin at January 1, 1970.TOD is stored as DWORD with a resolution of 1 millisecond.

\section*{Operators}
```

DATE TO UXINT
DATE TO- XINT
DATE TO XWORD
DATE TO BIT
DATE TO- BOOL
DATE_TO_BYTE
DATE TO- DINT
DATE-TO-DT
DATE_TO_DWORD
DATE TO INT
DATE-TO-
DATE_TO_LREAL
DATE TO LTIME
DATE 'TO-}\mathrm{ LWORD
DATE_TO_REAL
DATE TO SINT
DATE-TO-STRING
DATE_TO_TIME
DATE TO TOD
DATE 'TO-
DATE_TO_UINT
DATE TO ULINT
DATE 'TO-
DATE_TO_WORD

```
```

DATE_TO_WSTRING
DT TO__UXINT
DT TO XINT
DT_TO_XWORD
DT TO BIT
DT - TO-BOOL
DT_TO_BYTE
DT TO DATE
DT TO DINT
DT_TO_DWORD
DT_TO_INT
DT_TO_LINT
DT_TO_LREAL
DT TO LTIME
DT-TO-LWORD
DT_TO_REAL
DT TO SINT
DT-TO-STRING
DT_TO_TIME
DT TO_TOD
DT-TO-UDINT
DT_TO_UINT
DT TO ULINT
DT_TO_USINT
DT_TO_WORD
DT_TO_WSTRING
TOD_TO
UXINT
TOD_TO__XINT
TOD 'TO XWORD
TOD_TO_BOOL
TOD_TO_BIT
TOD_TO_BYTE
TOD_TO_DATE
TOD TO DINT
TOD_TO-DT
TOD_TO_DWORD
TOD_TO_INT
TOD_TO_LINT
TOD_TO_LREAL
TOD_TO_LTIME
TOD_TO_LWORD
TOD_TO_REAL
TOD_TO_SINT
TOD_TO_STRING
TOD_TO-TIME
TOD_TO_UDINT
TOD TO_UINT
TOD_TO-ULINT
TOD_TO_USINT
TOD_TO_WORD
TOD_TO_WSTRING
Long operators

|  | LDATE TO UXINT |
| :---: | :---: |
|  | LDATE TO XINT |
|  | LDATE TO XWORD |
|  | LDATE TO BIT |
|  | LDATE TO_BOOL |
|  | LDATE TO BYTE |
|  | LDATE TO DATE |
|  | LDATE_TO_DINT |
|  | LDATE_TO_DT |
|  | LDATE TO DWORD |

```
```

LDATE_TO_INT
LDATE TO LDT
LDATE TO LINT
LDATE TO LREAL
LDATE_TO_LTIME
LDATE TO LTOD
LDATE_TO_LWORD
LDATE_TO_REAL
LDATE TO SINT
LDATE_TO_STRING
LDATE_TO_TIME
LDATE_TO_TOD
LDATE_TO-UDINT
LDATE_TO_UINT
LDATE TO ULINT
LDATE_TO_USINT
LDATE_TO_WORD
LDATE_TO_WSTRING
LDT_TO___UXINT
LDT TO XINT
LDT-TO- XWORD
LDT TO BIT
LDT TO BOOL
LDT 'TO-BYTE
LDT_TO_DATE
LDT TO DINT
LDT-TO-DWORD
LDT_TO_INT
LDT TO_LDATE
LDT_TO_LINT
LDT-TO_LREAL
LDT TO LTIME
LDT 'TO- LTOD
LDT-TO-LWORD
LDT TO REAL
LDT 'TO-
LDT_TO_STRING
LDT TO_TIME
LDT TO-TOD
LDT_TO_UDINT
LDT_TO_UINT
LDT_TO_ULINT
LDT_TO_USINT
LDT_TO_WORD
LDT_TO_WSTRING
LTOD_TO___UXINT
LTOD TO XINT
LTOD_TO___XWORD
LTOD_TO_BOOL
LTOD TO BIT
LTOD_TO_BYTE
LTOD_TO_DATE
LTOD_TO_DINT
LTOD TO-DT
LTOD_TO_DWORD
LTOD_TO_INT
LTOD_TO_LDATE
LTOD_TO_LDT
LTOD_TO_LINT
LTOD_TO_LREAL
LTOD_TO_LTIME
LTOD_TO_LWORD
LTOD_TO_REAL

```
```

LTOD_TO_SINT
LTOD_TO_STRING
LTOD TO TIME
LTOD -TO-UDINT
LTOD_TO_UINT
LTOD_TO_ULINT
LTOD_TO_USINT
LTOD_TO_WORD
LTOD_TO_WSTRING

```

Converting to a The operator returns FALSE if and only if the operand value can be interpreted as "0". Boolean value
\begin{tabular}{|c|c|}
\hline xDate := DATE_TO_BOOL (D\#1970-1-1); & xDate = FALSE \\
\hline ```
xDateAndTime :=
DT_TO_BOOL(DT#1970-1-1-0:0:0);
``` & xDateAndTime \(=\) FALSE \\
\hline ```
xTimeOfDay :=
TOD TO BOOL(TOD#0:0:0);
``` & xTimeOfDay = FALSE \\
\hline xDate : = DATE_TO_BOOL (D\#2019-9-1); & xDate = TRUE \\
\hline ```
xDateAndTime :=
DT_TO_BOOL(DT#2019-9-1-12:0:0);
``` & \(\mathrm{xDateAndTime}=\) TRUE \\
\hline ```
xTimeOfDay :=
TOD_TO_BOOL(TOD#12:0:0);
``` & xTimeOfDay \(=\) TRUE \\
\hline
\end{tabular}

Converting to
an integer
The data types DATE and DT use the same memory format internally, namely a DWORD. The resolution for DATE is 1 day. The resolution for DT is 1 second. Both begin at January 1, 1970. TOD is stored as DWORD with a resolution of 1 millisecond.
\begin{tabular}{|c|c|}
\hline ```
diReturn_0 :=
DT_TO_DINT(DT#1970-1-1-0:0:0);
``` & diReturn_0 = 0 \\
\hline diReturn_1 := DATE_TO_DINT (D\#1970-1-1); & diReturn_1 = 0 \\
\hline ```
diReturn_2 :=
TOD_TO_DINT(TOD#0:0:0);
``` & diReturn_2 = 0 \\
\hline ```
diReturn_1 :=
DT TO DINT(DT#1970-1-1-0:0:1);
``` & diReturn_3 = 1 \\
\hline diReturn_3 := DATE_TO_DINT (D\#1970-1-2); & diReturn_4 = 86400 \\
\hline \begin{tabular}{l}
diReturn_5 := \\
DT_TO_DINT (DT\#2019-9-1-12:0:0.0);
\end{tabular} & diReturn_5 = 1567339200 \\
\hline diReturn_6 := DATE_TO_DINT (D\#2019-9-1); & diReturn_6 = 1567339200 \\
\hline ```
diReturn 7 :=
TOD_TO_DINT(TOD#12:0:0);
``` & diReturn_7 = 43200000 \\
\hline
\end{tabular}

\section*{Converting to a string}

\section*{NOTICE!}

\section*{String manipulation when converting to STRING or WSTRING}

When converting the type to STRING or WSTRING, the typed value is leftaligned as a character string and truncated if it is too long. Therefore, declare the return variable for the type conversion operators <> TO_STRING and <>_TO_WSTRING long enough that the character string has enough space without any manipulation.

The operands of type DATE, DATE_AND_TIME, TIME_OF_DAY, DT, or TOD, which are passed to an operator for a data and time conversion, are converted to their constant syntax (literal syntax). The generated string contains the keyword D\#, DT\# or TOD\# and then the size with its data and time unit, as indicated in the IEC 61131-3 specification.


\section*{Examples}

FBD implemen- The controller is in online mode in order to monitor the variables. tation language


\section*{See also}
- « "Type conversion operators" on page 2132
- \(\quad\) \& Chapter 6.4.1.20.3.37 "Boolean Conversion" on page 2154
- \(\&\) Chapter 6.4.1.20.3.36 "Overloading" on page 2152
- \(\xi^{*}\) Chapter 6.4.1.20.3.38 "Integer Conversion" on page 2159
- \({ }^{\bullet} \boldsymbol{y}\) Chapter 6.4.1.20.3.39 "Floating-Point Number Conversion" on page 2171
- \(\quad 4\) Chapter 6.4.1.20.3.40 "String Conversion" on page 2174
- \(\Leftrightarrow\) Chapter 6.4.1.20.3.41 "Time Conversion" on page 2182

\section*{Operator 'TRUNC'}

The IEC operator is used for converting the REAL data type into the DINT data type. CODESYS takes only the integer part of the number.

> In CoDeSys V2.3, the TRUNC operator converts REAL into INT. If you import a V2.3 project, then CODESYS automatically replaces TRUNC with TRUNC_INT.

If CODESYS cannot represent the input value by a DINT or INT, then the result of this function is undefined. The behavior of such input values is platform-dependent.

\section*{NOTICE!}

If the operand value for a type conversion operator is outside of the value range of the target data type, then the result output depends on the processor type and is therefore undefined. This is the case, for example, when a negative operand value is converted from LREAL to the target data type UINT.

Information can be lost when converting from larger data types to smaller data types.
```

Examples Result in diVar:1
ST
diVar := TRUNC(1.9); (* Result: 1 *)
diVar := TRUNC(-1.4); (* Result: -1 *)

```

\section*{See also}
- « "Type conversion operators" on page 2132

\section*{Operator 'TRUNC_INT'}

The IEC operator is used for converting the REAL data type into the INT data type. CODESYS takes only the integer part of the number.


TRUNC INT corresponds to the TRUNC operator in CoDeSys V2.3, and it is used aūtomatically at this point when importing V2.3 projects. Note the change function of TRUNC.

If CODESYS cannot represent the input value by a DINT or INT, then the result of this function is undefined. The behavior of such input values is platform-dependent.

\section*{NOTICE!}

If the operand value for a type conversion operator is outside of the value range of the target data type, then the result output depends on the processor type and is therefore undefined. This is the case, for example, when a negative operand value is converted from LREAL to the target data type UINT.

Information can be lost when converting from larger data types to smaller data types.
```

Examples Result in iVAR:1
ST:
iVar := TRUNC_INT(1.9); (* Result: 1 *)
iVar := TRUNC_INT(-1.4); (* Result: -1 *)

```

\section*{See also}
- «" "Type conversion operators" on page 2132

\section*{Operator 'ABS'}

This IEC operator yields the absolute value of a number.
Permitted data types for input and output variables and numeric constants: any numeric basic data type

Examples Result in i:2
ST:
i : \(=\operatorname{ABS}(-2)\);
FBD:


\section*{Operator 'SQRT'}

This IEC of course yields the square root of a number.
Permitted data types for input variables: any numeric basic data type
Permitted data types for output variables: REAL or LREAL
Examples \(\quad\)\begin{tabular}{ll} 
Result in \(q: 4\) \\
& ST: \\
& \(\mathrm{q}:=\operatorname{SQRT}(16) ;\) \\
& FBD:
\end{tabular}


\section*{Operator 'LN'}

This IEC operator yields the natural logarithm of a number.
Permitted data types for input variables: any numeric basic data type
Permitted data types for output variables: REAL and LREAL

\section*{Examples Result: 3.80666}

ST:
```

q := LN(45);

```

FBD:


\section*{Operator 'LOG'}

This IEC operator yields the base-10 logarithm of a number.
The input variable can be any numeric basic data type, but the output variable must be the data type REAL or LREAL.

Examples \(\quad\) Result in q: 2.49762
ST:
q := LOG(314.5);
FBD:


\section*{Operator 'EXP'}

This IEC operator yields the exponential function.
Permitted data types for input variables: any numeric basic data type
Permitted data types for output variables: REAL and LREAL
Examples Result in q: 7.389056099
ST:
q := EXP (2);
FBD:


\section*{Operator 'EXPT'}

This IEC operator raises a number to a higher power and returns the power of the base raised to the exponent: power = base exponent. The input values (parameters) are the base and the exponent. The power function is undefined if the base is zero and the exponent is negative. However, the behavior depends on the platform in this case.

Syntax:
EXPT (<base>,<exponent>)

Permitted data types for the input values: Numeric base data types (SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL, BYTE, WORD, DWORD, and LWORD)

Permitted data types for the return value: Floating-point number types (REAL and LREAL)

```

Example
Power function
with variables PROGRAM PLC_PRG
VAR
lrPow : LREAL;
iBase : INT := 2;
iExponent : INT := 7;
END_VAR
lrPow := EXPT(iBase, iExponent);

```
Return value: lrPow = 128

\section*{Operator 'SIN'}

The IEC operator yields the sine value of a number.
Permitted data types for input variables that measure the angle in radians: any numeric basic data type

Permitted data types for output variable: REAL and LREAL
\[
\begin{aligned}
& \text { The permitted range for the input value is }-2^{63} \text { to }+2^{63} \text {. On } x 86 \text { and } x 64 \text { systems: } \\
& \text { If the input value is outside of the permitted range, the function returns the input } \\
& \text { value }
\end{aligned}
\]

Examples
Result in q: 0.479426
ST:
q := SIN (0.5);
FBD:


\section*{Operator 'COS'}

The IEC operator yields the cosine value of a number.

Permitted data types for input variables that measure the angle in radians: any numeric basic data type
Permitted data types for output variables: REAL and LREAL

> The permitted range for the input value is \(-2^{63}\) to \(+2^{63}\). On x 86 and \(x 64\) systems: If the input value is outside of the permitted range, the function returns the input value

Examples Result in q: 0.877583
ST:
q : \(=\cos (0.5)\);
FBD:


\section*{Operator 'TAN'}

This IEC operator yields the tangent value of a number.
Permitted data types for input variables that measure the angle in radians: any numeric basic data type

Permitted data types for output variables: REAL and LREAL
\begin{tabular}{ll} 
Examples & Result in \(q: 0.546302\) \\
& ST: \\
& \(q:=\operatorname{TAN}(0.5) ;\) \\
& FBD:
\end{tabular}


\section*{Operator 'ASIN'}

This IEC operator yields the arcsine value of a number.
Permitted data types for input variables: any numeric basic data type
Permitted data types for output variables: REAL and LREAL


\section*{Operator 'ACOS'}

This IEC operator yields the arccosine value of a number. The value is computed in radians.
Permitted data types for input variables that measure the angle in radians: any numeric basic data type

Permitted data types for output variables: REAL and LREAL
Examples \(\quad\)\begin{tabular}{l} 
Result in \(\mathrm{q}: 1.0472\) \\
\\
ST: \\
\\
\(\mathrm{q}:\) \\
\\
FBD: \(: \operatorname{ACOS}(0.5) ;\)
\end{tabular}

\section*{Operator 'ATAN'}

This IEC operator yields the arctangent value of a number. The value is computed in radians.
Permitted data types for input variables that measure the angle in radians: any numeric basic data type

Permitted data types for output variables: REAL and LREAL


\section*{Operator '__DELETE'}

This operator is an extension of the IEC 61131-3 standard.

\section*{NOTICE!}

For compatibility, the compiler version must be \(>=\) 3.3.2.0.

The operator releases the memory of instances that the "__NEW" operator generated dynamically. The __DELETE operator does not have a return value and the operand is set to zero after this operation.

Requirement: In the properties dialog of the application, the "Use dynamic memory allocation" check box is selected in the "Application Build Options" tab.
__DELETE (<pointer>)

\section*{NOTICE!}

Two tasks should not call __DELETE simultaneously. Either you use a semaphore (SysSemEnter) or comparable method to prevent any concurrent calling of __DELETE , or you use __DELETE in one tasks only (recommended).

You can use a semaphore (SysSemEnter) to prevent two tasks from allocating memory at the same time. As a consequence, the extensive use of \(\qquad\) DELETE causes higher jitter.

If Pointer references a function block, then CODESYS calls the associated FB_EXIT method before the pointer is set to zero.

\section*{Examples}
```

FUNCTION_BLOCK FBDynamic
VAR_INPUT
in1, in2 : INT;
END_VAR
VAR_OUTPUT
out : INT;
END_VAR
VAR
test1 : INT := 1234;
_inc : INT := 0;
_dut : POINTER TO DUT;
neu : BOOL;
END_VAR
out := in1 + in2;
METHOD FB_Exit : BOOL
VAR INPUT
bInCopyCode : BOOL;
END_VAR
__Delete(_dut);
METHOD FB_Init : BOOL
VAR_INPUT
bInitRetains : BOOL;
bInCopyCode : BOOL;
END_VAR
_dut := __NEW(DUT);
METHOD INC : INT
VAR_INPUT
END_VAR
_inc:= _inc + 1;
INC := _\overline{inc;}
PLC_PRG(PRG)
VAR
pFB : POINTER TO FBDynamic;
bInit: BOOL := TRUE;
bDelete: BOOL;
loc : INT;
END_VAR
IF (bInit) THEN
pFB := __NEW(FBDynamic);
bInit := FALSE;
END_IF

```
```

IF (pFB <> 0) THEN
pFB^(in1 := 1, in2 := loc, out => loc);
pFB^.INC();
END_IF
IF (bDelete) THEN
DELETE (pFB) ;
END_\overline{IF}

```

\section*{Operator '__ISVALIDREF'}

This operator is an extension of the IEC 61131-3 standard.
The operator is used for checking whether a reference refers to a valid value. For a description of use and an example, refer to the description for the REFERENCE data type.
See also
- \({ }^{4}\) Chapter 6.4.1.20.5.14 "Reference" on page 2245

\section*{Operator '__NEW'}

The operator is an extension of the IEC 61131-3 standard.
The _NEW operator reserves dynamic memory to instantiate function blocks, user-defined data types, or arrays of standard types. The operator returns a matching typed pointer.

Requirement: In the properties dialog of the parent application, on the "Application Build Options" tab, the "Use dynamic memory allocation" option is selected.

\section*{Syntax}
```

<pointer name> := __NEW( <type> ( , <size> )? );
__DELETE( <pointer name> );
<type> : <function block> | <data unit type> | <standard data type>

```

The operator generates an instance of the type <type> and returns a pointer to this instance. Then the initialization of the instance is called. If <type> is a scalar standard data type, then the optional operand <size> is also evaluated. Then the operator generates an array of type <standard data type> and size <size>. If the attempt to allocate memory fails, then ___NEW returns the value 0 .

Use the operator within the assignment ":=". Otherwise an error message is displayed.
A function block or a user-defined data type whose instance is created dynamically with NEW uses a fixed memory area. Here it is required that you mark the objects with the pragma \{attribute 'enable_dynamic_creation'\}. It is not required for function blocks that are part of a library.

If you change the data layout of the function block in online mode, then you cannot execute a login with an online change afterwards. This is because the memory area of the function block instance has been invalidated. You change the data layout when you add new variables to the function block, delete existing variables, or change the data types of variables.
```

Example
Array (DWORD):
PROGRAM PLC_PRG
VAR
pdwScalar : POINTER TO DWORD; //Typed pointer
xInit : BOOL := TRUE;
xDelete : BOOL;
END_VAR
IF (xInit) THEN
pdwScalar := __NEW(DWORD, 16); // Allocates memory (16 dwords)
and assigns them to pointer pdwScalar
END_IF
IF (xDelete) THEN
DELETE(pdwScalar); // Frees memory of pointer
END_\overline{IF}
{attribute 'enable_dynamic_creation'}
FUNCTION_BLOCK FBComputeGamma
VAR_INPUT
iAlpha : INT;
iBeta : INT;
END_VAR
VAR_OUTPUT
iGamma : INT;
END VAR
VAR
END_VAR
iGamma := iAlpha + iBeta;
PROGRAM PLC_PRG
VAR
pComputeGamma : POINTER TO FBComputeGamma; // Typed pointer
xInit : BOOL := TRUE;
xDelete : BOOL;
iResult : INT;
END_VAR
IF (xInit) THEN
pComputeGamma := __NEW(FBComputeGamma); // Allocates memory
xInit := FALSE;
END_IF
pComputeGamma^.iAlpha := (pComputeGamma^.iAlpha + 1)MOD 100; //
Sets first input of pComputeGamma
pComputeGamma^.iBeta := 10; // Sets second input of pComputeGamma
pComputeGamma^(); // Calls the FB pComputeGamma is pointing to
iResult := pComputeGamma^.iGamma; // Reads output of pComputeGamma
IF (xDelete) THEN
DELETE(pComputeGamma); // Frees memory
END_\overline{IF}
User-defined
data type
(DUT):
{attribute 'enable_dynamic_creation'}
TYPE ABCDATA :
STRUCT
iA, iB, iC, iD : INT;
END_STRUCT
END_TYPE
PROGRAM PLC_PRG
VAR
pABCData : POINTER TO ABCDATA; // Typed pointer
xInit : BOOL := TRUE;

```
```

    xDelete : BOOL;
    END_VAR
IF (xInit) THEN
pABCData := NEW(ABCDATA); // Allocates memory
xInit := FALSE;
END IF
IF (xDelete) THEN
END_\overline{IF}

```

Array (BYTE):
```

    PROGRAM PLC_PRG
    VAR
pbDataAlpha : POINTER TO BYTE;
pbDataBeta : POINTER TO BYTE;
xInit : BOOL := TRUE;
xDelete : BOOL;
usiCnt : USINT;
bTestC: BYTE;
END_VAR
IF (xInit) THEN
pbDataAlpha := __NEW(BYTE, 16); // Allocates 16 bytes for
pbDataAlpha
pbDataBeta := __NEW(BYTE); // Allocates memory for pbDataBeta
xInit := FALSE;
FOR usiCnt := 0 TO 15 DO
pbDataAlpha[usiCnt] := usiCnt; // Writes to new array
END FOR
pbDātaBeta^:= 16\#FF; // Writes to new data
END_IF
bTestC := pbDataAlpha[12]; // Reads new array by index access
IF (xDelete) THEN // Frees memory
DELETE (pbDataAlpha);
DELETE (pbDataBeta);
END_\overline{IF}

```

\section*{NOTICE!}

We do not recommend the simultaneous execution of two tasks that both call the __NEW operator. You use either a semaphore (SysSemEnter) or a comparable technique to prevent a concurrent call of __NEW. However, this results in a higher jitter when \(\qquad\) NEW is applied extensively.

We recommend that you call \(\qquad\) NEW operators in one task only.

\section*{See also}
- *3 Chapter 6.4.1.21.4.11.10 "Dialog 'Properties - Application Build Options"' on page 2759
- « Chapter 6.4.1.21.2.9 "Object 'DUT"' on page 2461
- \({ }^{*}\) Chapter 6.4.1.21.2.21.3 "Object 'Function Block"' on page 2479
- \& Chapter 6.4.1.20.6.3.13 "Attribute 'enable_dynamic_creation"' on page 2281
- \& Chapter 6.4.1.20.11.159 "Compiler error C0509" on page 2403

\section*{Operator '_QUERYINTERFACE'}

This operator is an extension of the IEC 61131-3 standard.
At runtime, the operator executes a type conversion of an interface reference into another type. The operator returns a BOOL result. TRUE means that CODESYS has performed the conversion successfully.
__QUERYINTERFACE (<ITF_Source>,<ITF_Dest>);
1. Operand: Interface reference or FB interface
2.Operand: Interface reference with required target type

The requirement for the explicit conversion is that both the ITF_Source and ITF_Dest are derived from Interface \(\qquad\) System. IqueryInterface. This interface is implicitly available does not require a library.

\section*{Example}
```

INTERFACE ItfBase EXTENDS

```
\(\qquad\)
``` System.IQueryInterface
METHOD mbase : BOOL
END_METHOD
INTERFACE ItfDerived1 EXTENDS ItfBase
METHOD mderived1 : BOOL
END_METHOD
INTERFACE ItfDerived2 EXTENDS ItfBase
METHOD mderived2 : BOOL
END_METHOD
FUNCTION_BLOCK FB1 IMPLEMENTS ItfDerived1
METHOD m\overline{b}ase : BOOL
    mbase := TRUE;
END METHOD
MET\overline{H}OD mderived1 : BOOL
    mderived1 := TRUE;
END_METHOD
END_FUNCTION_BLOCK
FUNCTION_BLOCK FB2 IMPLEMENTS ItfDerived2
METHOD m\overline{b}ase : BOOL
    mbase := FALSE;
END_METHOD
METH̄OD mderived2 : BOOL
    mderived2 := TRUE;
END_METHOD
END_FUNCTION_BLOCK
PROGRAMM POU
VAR
    inst1 : FB1;
    inst2 : FB2;
    itfbase1 : ItfBase := inst1;
    itfbase2 : ItfBase := inst2;
    itfderived1 : ItfDerived1 := 0;
    itfderived2 : ItfDerived2 := 0;
    xResult1, xResult2, xResult3, xResult4: BOOL;
END_VAR
xResult1 := __QUERYINTERFACE(itfbase1, itfderived1); // xResult =
TRUE, itfderivedi1 <>0
                                    // references
the instance inst1
xResult2 := __QUERYINTERFACE(itfbase1, itfderived2); // xResult =
FALSE, itfderived2 = 0
xResult3 := __QUERYINTERFACE(itfbase2, itfderived1); // xResult =
FALSE, itfderived1 = 0
xResult4 := QUERYINTERFACE(itfbase2, itfderived2); // xResult =
TRUE, itfderived2 <> 0
                                    // references
the instance inst2
```


## Operator '__QUERYPOINTER'

This operator is an extension of the IEC 61131-3 standard.

At runtime, the operator makes it possible to convert the type of an interface reference of a function block to a pointer. The operator returns a BOOL result. TRUE means that CODESYS has performed the conversion successfully.

## NOTICE!

For compatibility, the definition of the pointer to be converted must be an extension of the base interface "__SYSTEM.IQueryInterface".

QUERYPOINTER (<ITF_Source>, <Pointer_Dest>)
The operator receives an interface reference or a FB instance with the required target types as the first operand and a pointer as the second operand. After processing __QUERYPOINTER, Pointer_Dest receives the pointer to the reference or instance of a function block that the ITF_Source interface reference currently refers to. Pointer_Dest is not typed and can be cast to any type. You have to make sure of the type. For example, the interface could offer a method that returns a type code.

Operators '__TRY', '__CATCH', '__FINALLY', '__ENDTRY'
These operators are extended from the IEC 61131-3 standard and they are used for specific exception handling in IEC code.

## Syntax

```
TRY
    <statements_try>
    CATCH (exec)
            <statements_catch>
```

$\qquad$

```
        FINALLY
            <statements_finally>
        ENDTRY
            <statements_next>
```

When a statement in the _Try operator throws an exception, the application does not stop. Instead, the application executes the statements in _ catch, starts the exception handling, and then executes the statements in $\qquad$ FINALLY. The exception handling ends with $\qquad$ ENDTRY, and the application executes the subsequent statements.

An IEC variable for an exception has the data type System. ExceptionCode.

```
Example If the statement in __TRY throws an exception, then program execution is not stopped.
    Instead, the statement in ___CATCH is executed. Therefore, in this example, the application
executes the exc function, then the statement in __FINALLY, and finally the statement in
ENDTRY.
```

```
FUNCTION Tester : UDINT
```

FUNCTION Tester : UDINT
VAR_INPUT
VAR_INPUT
count : UDINT;
count : UDINT;
END_VAR
END_VAR
VAR_OUTPUT
VAR_OUTPUT
strExceptionText : STRING;
strExceptionText : STRING;
END_VAR
END_VAR
VAR
VAR
exc : __SYSTEM.ExceptionCode;
exc : __SYSTEM.ExceptionCode;
END_VAR
END_VAR
TRY
Tester := tryFun(count := count, testcase := g_testcase); // This
statement is tested. If it throws an exception, then the statement
in __CATCH is executed first, and then the statement in __INALLY.
CATCH (exc)
HandleException(exc, strExceptionText => strExceptionText);
FINALLY
GVL.g_count := GVL.g_count + 2;
ENDTRY

```

\section*{See also}
- \({ }^{4}\) Chapter 6.4.1.21.3.7.19 "Command 'Stop Execution on Handled Exceptions'"
on page 2639

\section*{Data Type '__System.Exce ptionCode'}
```

TYPE ExceptionCode :
(
RTSEXCPT_UNKNOWN := 16\#FFFFFFFF,
RTSEXCPT NOEXCEPTION := 16\#00000000,
RTSEXCPT_WATCHDOG := 16\#00000010,
RTSEXCPT_HARDWAREWATCHDOG := 16\#00000011,
RTSEXCPT IO CONFIG ERROR := 16\#00000012,
RTSEXCPT_PRŌGRAMCH\overline{E}CKSUM := 16\#00000013,
RTSEXCPT_FIELDBUS_ERROR := 16\#00000014,
RTSEXCPT 'IOUPDATE ERROR := 16\#00000015,
RTSEXCPT_CYCLE_TIM
RTSEXCPT_ONLCHĀNGE_\overline{PROGRAM_EXCEEDED := 16\#00000017,}
RTSEXCPT 'UNRESOLVED EXTREF\overline{S}}:=16\#00000018
RTSEXCPT DOWNLOAD REJECTED := 16\#00000019,
RTSEXCPT_BOOTPROJECT_REJECTED_DUE_RETAIN_ERROR := 16\#0000001A,
RTSEXCPT_LOADBOOTPROJECT_FAILED := 16\#0000001B,
RTSEXCPT_OUT_OF_MEMORY - := 16\#0000001C,
RTSEXCPT_RETAIN_MEMORY_ERROR := 16\#0000001D,
RTSEXCPT - BOOTPRO
= := 16\#0000001E,
RTSEXCPT_BOOTPROJECT\overline{TARGETMISMATCH := 16\#00000021,}
RTSEXCPT_SCHEDULEERROR
:= 16\#00000022,
RTSEXCPT FILE CHECKSUM ERR
:= 16\#00000023,
RTSEXCPT_RETA\overline{IN_IDENTITYY_MISMATCH := 16\#00000024,}
RTSEXCPT_IEC_TAS\overline{K_CONFIG_ERROR := 16\#00000025,}
RTSEXCPT_APP_TARG\overline{ETT_MISMÄTCH := 16\#000000026,}
RTSEXCPT 'ILL\overline{EGAL INSTTRUCTION := 16\#000000050,}
RTSEXCPT_ACCESS_VIOLATION := 16\#00000051,

```
```

RTSEXCPT_PRIV_INSTRUCTION
RTSEXCPT_IN_P\overline{A}GE_ERROR := 16\#00000053,
RTSEXCPT_STĀCK_O\overline{VERFLOW := 16\#00000054,}
RTSEXCPT_INVAL\overline{ID_DISPOSITION := 16\#00000055,}
RTSEXCPT_INVALID_HANDLE := 16\#00000056,
RTSEXCPT_GUARD_P\overline{A}GE := 16\#00000057,
RTSEXCPT_-DOUBLE_FAULT := 16\#00000058,
RTSEXCPT_INVALI\overline{D_OPCODE }
RTSEXCPT_MISALIG\overline{NMENT := 16\#00000100,}
RTSEXCPT_ARRAYBOUNDS := 16\#00000101,
RTSEXCPT_DIVIDEBYZERO
RTSEXCPT_OVERFLOW := 16\#00000103,
RTSEXCPT_NONCONTINUABLE := 16\#00000104,
RTSEXCPT_PROCESSORLOAD_WATCHDOG := 16\#00000105,
RTSEXCPT_FPU_ERROR - := 16\#00000150,
RTSEXCPT_FPU_DENORMAL_OPERAND := 16\#00000151,
RTSEXCPT_FPU_-DIVIDEBYZERO
RTSEXCPT FPU '_ INEXACT RESULT
RTSEXCPT_FPU_INVALID_OPERATION
RTSEXCPT_FPU_OVERFLO\overline{W}
RTSEXCPT_FPU_STACK_CHECK
RTSEXCPT_FPU_UNDER\overline{FLOW :}=16\#00000157,
RTSEXCPT_VEND̄OR_EXCEPTION_BASE := 16\#00002000
RTSEXCPT_USER_E\overline{XCEPTION_B\overline{A}SE }:=16\#00010000
) UDINT ;
END_TYPE

```
```

    := 16#00000052,
    ```
    := 16#00000052,
    := 16#00000102,
    := 16#00000102,
    := 16#00000152,
    := 16#00000152,
    := 16#00000153,
    := 16#00000153,
    := 16#00000154,
    := 16#00000154,
    := 16#00000155,
    := 16#00000155,
:= 16#00000156,
```

:= 16\#00000156,

```

\section*{Operator '__VARINFO'}

This operator is an extension of the IEC 61131-3 standard.
The operator yields information about a variable. You can save the information as data structure in a variable of data type __SYSTEM.VAR_INFO.
Syntax in the declaration:
```

<name of the info variable> : __SYSTEM.VAR_INFO; // Data structure
for info variable
<name of the info variable> := __ VARINFO( <variable name> ); // Call
of the operator

```
Syntax for the
call:
```

Example

```
```

FUNCTION_BLOCK FB_Velocity

```
FUNCTION_BLOCK FB_Velocity
VAR_INPUT
VAR_INPUT
    rVelocity: REAL := 1.2;
    rVelocity: REAL := 1.2;
END VAR
END VAR
VAR_OUTPUT
VAR_OUTPUT
END_VAR
END_VAR
VAR
VAR
    infoVelocity: __SYSTEM.VAR_INFO; //Info of Velocity
    infoVelocity: __SYSTEM.VAR_INFO; //Info of Velocity
END_VAR
END_VAR
infoVelocity := __VARINFO(rVelocity); // Gets the info of Velocity
infoVelocity := __VARINFO(rVelocity); // Gets the info of Velocity
locally
locally
PROGRAM PLC_PRG
PROGRAM PLC_PRG
VAR
VAR
    iCounter : INT := 0; // Counts the calls
    iCounter : INT := 0; // Counts the calls
    infoCounter : __SYSTEM.VAR_INFO; //Info of Counter
    infoCounter : __SYSTEM.VAR_INFO; //Info of Counter
    arrA : ARRAY [\overline{1..2, 1..2, 1..2] OF INT := [0, 1, 2, 3, 4, 5, 6,}
    arrA : ARRAY [\overline{1..2, 1..2, 1..2] OF INT := [0, 1, 2, 3, 4, 5, 6,}
7]; // Stores the A data
7]; // Stores the A data
    infoA : __SYSTEM.VAR_INFO; //Info of A
    infoA : __SYSTEM.VAR_INFO; //Info of A
    fbVel : FB_Velocity;
    fbVel : FB_Velocity;
END_VAR
END_VAR
iCounter := iCounter + 1;
iCounter := iCounter + 1;
infoCounter := _ VARINFO(iCounter);
infoCounter := _ VARINFO(iCounter);
infoA := __VARINFO(arrA);
infoA := __VARINFO(arrA);
fbVel();
fbVel();
The iCounter and arrA variables are recognized in the application code. The variable information is saved in the infoCounter and infoA variables. Moreover, the FB_Velocity function block is instantiated.
```

| Data type <br> SYSTEM.VAR_ <br> INFO | Name | Data type | Initialization | Description |
| :--- | :--- | :--- | :--- | :--- |
| ByteAddres <br> s | DWORD | 0 | Address of the variable <br> Example: $16 \# 072 \mathrm{E} 35 \mathrm{EC}$ <br> Note: For bit access of a variable <variable <br> name>. < bit index>, the address of the <br> variable that contains the bit is given. |  |
| ByteOffset | DWORD | 0 | Offset of the variable address (in bytes). <br> Example: 13936 bytes. |  |
|  |  | Note: If the variable is global, then the offset <br> is relative to the beginning of the area. If <br> the variable is a local variable in a function <br> or method, then the offset is relative to the <br> current stack frame. If the variable is a local <br> variable in a function block, then the offset is <br> relative to the function block instance. |  |  |


| Name | Data type | Initialization | Description |
| :---: | :---: | :---: | :---: |
| Area | DInT | 0 | Memory area number Area in the runtime system. <br> Example: -1 : Means that the variable is not global in the memory, but relative to an instance or on the stack. <br> Note: The memory areas are devicedependent. |
| BitNr | InT | 0 | Number of bits (in bytes) <br> Example: 16\#00FF bytes <br> Note: If the variable is not an integer data type, then: BitNr $=-1=16 \#$ FFFF. |
| BitSize | INT | 0 | Memory size of the variable (in bits) <br> Example: 16 bits |
| BitAddress | UDInt | 0 | Bit address of the variable <br> Requirement: The variable is located in the input memory area I, output memory area $Q$, or marker memory area M . Otherwise the value is undefined. |
| TypeClass | TYPE_CLASS | TYPE_BOOL | Data type class of the variable <br> Example: TYPE_INT, TYPE_ARRAY <br> Note: For user-defined data types or function block instances, TYPE_USERDEF is output as the data type class. |
| TypeName | STRING (79) | " | Date type name of the variable as STRING (79) <br> Note: For user-defined data types, the function block name or the DUT name is output. <br> Example: 'INT', 'ARRAY' |
| NumElement <br> s | UDINT | 0 | Number of array elements <br> Requirement: The variable has the data type ARRAY. <br> Example: 8 |
| $\begin{aligned} & \text { BaseTypeCl } \\ & \text { ass } \end{aligned}$ | TYPE_CLASS | TYPE_BOOL | Elementary basic data type of the array elements. <br> Requirement: The variable has the data type ARRAY. <br> Example: TYPE INT for arrA : ARRAY [1..2,1..2,1..2] OF INT; |
| $\begin{aligned} & \text { ElemBitSiz } \\ & \mathrm{e} \end{aligned}$ | UDINT | 0 | Memory size of the array element (in bits) <br> Requirement: The variable has the data type ARRAY. <br> Example: 16 bits for arrA : ARRAY [1..2,1..2,1..2] OF INT; |


| Name | Data type | Initialization | Description |
| :---: | :---: | :---: | :---: |
| MemoryArea | $\begin{aligned} & \text { MEMORY_ARE } \\ & \text { A } \end{aligned}$ | MEM_MEMORY | Information about the memory area <br> - MEM_GLOBAL: Global memory area For example in Area 0 <br> - MEM_LOCAL: Local memory area in Area -1 <br> - MEM_MEMORY: Marker memory area $\circ \mathrm{M}$ For example in 16\#10 in Area 1 <br> - MEM_INPUT: Input memory area \%I For example in 16\#04 in Area 2 <br> - MEM_OUTPUT: Output memory area $\% \mathrm{Q}$ For example in 16\#08 in Area 3 <br> - MEM_RETAIN: Retain memory area For example in 16\#20 in Area 0 <br> Example: MEM_GLOBAL <br> Note: The memory area configuration is device-dependent. |
| Symbol | STRING (39) | " | Variable name as STRING (39) <br> Example: 'iCounter', 'arrA' |
| Comment | STRING (79) | " | Comment of the variable declaration <br> Example: 'Counts the calls' or 'Stores the A data' |

## Operator '_CURRENTTASK'

This operator is an extension of the IEC 61131-3 standard.
In runtime mode, the operator provides information about the IEC task that is currently running.


The operator is supported only on target systems in which the target system setting memory-layout $\backslash$ max-stack-size is set to a value $>0$.

The operator allows for access to a structure with two variables:

- TaskIndex: Zero-based index that identifies the task
- pTaskInfo: Detailed information about the currently running task. It can be assigned to a POINTER TO Task_Info2 from the library CmpIecTask.
The operator cannot be used in the declaration of a POU. This would result in an error message. If the current task cannot be determined, then the TaskIndex -1 and the pTaskInfo are zero.


## Example

```
//Declaration
VAR
            idx : INT;
            pInfo : POINTER TO Task_Info2;
END_VAR
//Program code
idx := ___CURRENTTASK.TaskIndex;
pInfo := __CURRENTTASK.pTaskInfo;
```


## Operator '__COMPARE_AND_SWAP

The multicore operator is an extension of the IEC 61131-3 standard.
The operator can be used for implementing a semaphore, for example to guarantee exclusive access to a variable written to by different tasks.

COMPARE_AND_SWAP gets a pointer to a data type __XWORD variable, an old value, and a new value as its input (example: bMutex $:=$ __COMPARE_AND_SWAP (ADR (dwSynch), dwOld, dwNew) ; ). The old and new values can also be data type __XWORD variables. The referenced __XWORD variable is compared with the old value and if both are equal, then the new value is written. The result of the function is TRUE when the new value could be written.

The compiler automatically replaces the data type $\qquad$ XWORD with DWORD on 32-bit systems and LWORD on 64-bit systems.

This operation is atomic, so it cannot be interrupted by another task, even on multicore platforms.

```
Example The following example shows a typical usage. Exclusive access to a type STRING variable,
which is addressed via the pstrOutput pointer, should be implemented.
The access to a string is not atomic. If multiple tasks write to the same string at the same time, then the contents may be inconsistent.
With this function, it is now possible to write the same STRING variable in different tasks.
```

```
FUNCTION ExclusiveStringWrite : BOOL
```

FUNCTION ExclusiveStringWrite : BOOL
VAR_INPUT
VAR_INPUT
strToWrite : STRING;
strToWrite : STRING;
pstrOutput : POINTER TO STRING;
pstrOutput : POINTER TO STRING;
END_VAR
END_VAR
VAR_STAT
VAR_STAT
dwWSynch : XWORD;
dwWSynch : XWORD;
END_VAR
END_VAR
VAR
VAR
bMutex: BOOL;
bMutex: BOOL;
END_VAR
END_VAR
bMutex:= _COMPARE_AND_SWAP(ADR(dwSynch), 0, 1);
bMutex:= _COMPARE_AND_SWAP(ADR(dwSynch), 0, 1);
(* compare with 0 and write 1 as atomic operation *)
(* compare with 0 and write 1 as atomic operation *)
IF bMutex THEN // bMutex is TRUE if write
IF bMutex THEN // bMutex is TRUE if write
could be done
could be done
pstrOutput^ := strToWrite; // Now you can write safely
pstrOutput^ := strToWrite; // Now you can write safely
on the string
on the string
dwSynch := 0; // The __XWORD variable must
dwSynch := 0; // The __XWORD variable must
be reset.
be reset.
ExclusiveStringWrite := TRUE; // Writing was successful
ExclusiveStringWrite := TRUE; // Writing was successful
ELSE
ELSE
ExclusiveStringWrite := FALSE; // Writing was not successful
ExclusiveStringWrite := FALSE; // Writing was not successful
END_IF

```
END_IF
```


## See also

- «, "Multicore operators" on page 2133
- $\Leftrightarrow$ Chapter 6.4.1.20.3.69 "Operator 'TEST_AND_SET"' on page 2215


## Operator '_XXADD'

The multicore operator is an extension of the IEC 61131-3 standard.
The operator can be used for implementing an atomic counter. If an integer variable is incremented by means of ordinary addition, for example iTest := iTest +1 ; , then this operation is not executed atomically. Another access to the variable could take place between reading and writing the variable.

If the counter is incremented in multiple tasks, then the counter result can be less than the number of counting cycles. So if two tasks execute the above code one time and the variable previously had the value 0 , then the variable can then have the value 1 . This is especially problematic if arrays are being processed in multiple tasks and a unique index is required for the array in each processing cycle.
When the __XADD operator is called, it gets a pointer to a type DINT variable as the first summand and a type DINT value as the second summand. __ XADD returns the old value of the first summand and in the same step adds the second summand to the first summand.
For example, the function call can look like this: diold := $\qquad$ XADD (ADR(diVar), deAdd);

```
Example The following example shows a typical usage. An array should be populated from two tasks. In
                the process, all positions in the array should be used and no position should be overwritten.
                With this function, multiple tasks can populate a Boolean array.
```

```
FUNCTION WriteToNextArrayPosition : BOOL
```

FUNCTION WriteToNextArrayPosition : BOOL
VAR_EXTERNAL
VAR_EXTERNAL
g_diIndex : DINT; // Index and array are globally defined and
g_diIndex : DINT; // Index and array are globally defined and
used by multiple tasks
used by multiple tasks
g_boolArray : ARRAY [0..1000] OF BOOL;
g_boolArray : ARRAY [0..1000] OF BOOL;
END VĀR
END VĀR
VAR_INPUT
VAR_INPUT
bToWrite : BOOL;
bToWrite : BOOL;
END_VAR
END_VAR
VAR
VAR
diIndex : DWORD;
diIndex : DWORD;
END_VAR
END_VAR
diIndex := __XADD(ADR(g_diIndex), 1); // Returns a unique
diIndex := __XADD(ADR(g_diIndex), 1); // Returns a unique
index
index
WriteToNextArrayPosition := FALSE;
WriteToNextArrayPosition := FALSE;
IF (diIndex >= 0 AND diIndex <= 1000) THEN
IF (diIndex >= 0 AND diIndex <= 1000) THEN
g_boolArray[diIndex] := bToWrite; //Writes to unique
g_boolArray[diIndex] := bToWrite; //Writes to unique
index
index
WriteToNextArrayPosition := TRUE; // TRUE: Array was
WriteToNextArrayPosition := TRUE; // TRUE: Array was
not full yet
not full yet
END_IF

```
END_IF
```


## See also

- "Multicore operators" on page 2133


## Operator '_POSITION'

The operator is an extension of the IEC 61131-1 standard.
At runtime, the operator yields the position of a variable in the declaration part or in the implementation part of a POU. The operator has to be assigned the variables of type STRING in the declaration part or in the implementation part.
Result of $\qquad$ POSITION

- Declaration part: 'Line <line number> (Decl)'
- Implementation part: 'Line <line number>, Column <column number> (Impl)'


## Example

```
PROGRAM PROG1
VAR
    strPOS : STRING := __POSITION(); //Yields the line number of
this declaration
    strlocalPOS : STRING;
END_VAR
    strlocalPOS := __POSITION(); //Yields the line and column
number of this assignment
```


## Operator '__POUNAME'

The operator is an extension of the IEC 61131-1 standard.

At runtime, the operator yields the name of the POU that contains the operator $\qquad$ POUNAME. The result is of type STRING.

The result of $\qquad$ POUNAME depends where it is used:

- In a program: program name
- In a function name: function name
- In a function block: function block name
- In a method: the method name qualified with the FB name
- In a Get/Set accessor of a property: the property name + Get/Set qualified with the FB name
- In a GVL: GVL name
- In a structure: structure name
- In a data structure UNION: union name


## Example

```
PROGRAM PROG1
VAR
    strPOU : STRING := __POUNAME(); //Yields 'PROG1'
    strlocalPOU : STRING\overline{;}
END_VAR
    strlocalPOU := __POUNAME(); //Yields 'PROG1'
```


## Operator 'TEST_AND_SET'

The multicore operator is an extension of the IEC 61131-3 standard.
The operator can be used for implementing a semaphore, for example to guarantee exclusive access to a variable written to by different tasks.

TEST_AND_SET gets a type DWORD variable as its input. Write access to this variable must be possible. The variable is set to 1 and the previous value is returned as the result.
The operation is atomic, which means that it cannot be interrupted by another task. This also applies to multicore platforms.
For example, the call in the program is dwOldValue $:=$ TEST_AND_SET (dw) ; , in which the variables dwOldValue and dw must be of data type DWORD.


See also

- ¿ Chapter 6.4.1.20.3.65 "Operator '_COMPARE_AND_SWAP" on page 2212
- "Multicore operators" on page 2133


## Operator - Global namespace

This operator is an extension of the IEC 61131-3 standard.
An instance path that begins with a dot (.) always opens a global namespace. If there is a local variable that has the same name <varname> as a global variable, then you refer to the global variable as .<varname>.

## Operator - Namespace for global variables lists

This operator is an extension of the IEC 61131-3 standard.
You can use the name of a global variables list (GVL) as a namespace identifier for the variables that are defined in the list. This makes is possible to use variables with the same name in different global variables lists and still access specific variables uniquely. You use a dot (.) to prepend the name of the global variables list to the variable name.
<global variable list name>.<variable>
Example

```
globlist1.varx := globlist2.varx;
```

The globlist1 and globlist2 global variables lists each contain a varx variable. CODESYS copies the varx global variable from the globlist2 list to varx in the globlist1 list.

If you reference a variable that is declared in several global variables lists without referencing the prepended list name, then an error message is printed.

## Operator - Library namespace

This operator is an extension of the IEC 61131-3 standard.
Syntax: <library namespace>.<library identifier>
Example: LIB_A.FB_A
A library module identifier is appended with the library namespace (as a prefix separated by a dot) for unique and qualified access to the library module. The namespace usually coincides with the library name.

Example A library is included in a project and contains the module FB_A. However, the function block with the same name is already available locally in the project. Identify the library module as LIB_A.FB_A in order to access the library module, not the local function block.

```
var1 := FB_A(in := 12); // Call of the project function FB_A
var2 := LIB_A.FB_A(in := 22); // Call of the library function FB_A
```

You can define another identifier for the namespace. To do this, specify a namespace in the project information (library developers: when creating a library project). As an alternative, you can specify a specific namespace for a library in the library manager in the "Properties" dialog box (application developers: when creating an application).

See also

- ⓨ Chapter 6.4.1.17 "Using Libraries" on page 2034
- $\Leftrightarrow$ Chapter 6.4.1.21.3.15.4 "Command 'Placeholders'" on page 2716
- $y^{\prime}$ Chapter 6.4.1.21.2.16 "Object 'Library Manager'" on page 2469


## Operator - Enumeration namespace

This operator is an extension of the IEC 61131-3 standard.
You can use the TYPE name of an enumeration for unique access to an enumeration constant. In this way, you can use the same constant names in different enumerations.

The enumeration name is prepended to the constant name with a dot (.).
<enumeration name>.<constant name>

```
Example The constant Blue is a component of both the enumeration Colors and the enumeration
Feelings.
color := Colors.Blue; // Access to component blue in enumeration
Colors
feeling := Feelings.Blue; // Access to component blue in
enumeration Feelings
```


## Operator '__POOL'

The operator is an extension of the IEC 61131-3 standard.
The operator is used to reference objects which are managed in the global POU pool (in the "POUs" view). The operator directly accesses objects in the "POUs" view.

```
Example
```

```
PROGRAM PLC_PRG
```

PROGRAM PLC_PRG
VAR
VAR
svar_pou : STRING;
svar_pou : STRING;
END_VAR
END_VAR
svar_pou := ___POOL.POU();

```
svar_pou := ___POOL.POU();
```


## See also

- \#Chapter 6.4.1.20.8 "Shadowing Rules" on page 2333


## Operator 'INI'



```
The INI operator is a CoDeSys V2.3 operator. In CODESYS V3, the FB init method replaces the INI operator. You can still use this operator in projects that are imported from CoDeSys V2.3.
The INI operator is used for initializing retain variables of a function block instance used in a POU.
Assign the operator to a Boolean variable.
```



See also

- ${ }^{\Downarrow}$, Chapter 6.4.1.20.10 "Methods 'FB_Init', 'FB_Reinit', and 'FB_Exit"' on page 2336
- $\boldsymbol{y y}^{\circ}$ Chapter 6.4.1.9.19 "Data Persistence" on page 1920


### 6.4.1.20.4 Operands

## General

Constants and Constants are identifiers for unchangeable values. You can declare constants locally within a literals POU or globally within a global variable list. The declaration segment is extended with the keyword CONSTANT.

Constants are also character strings that represent the value of a base type, such as integers or floating-point numbers (for example, 16\#FFFF FFFF, T\#5s, or -1.234 E-5). To distinguish between them, these constants are also called literals, literal constants, or unnamed constants. There are logical (TRUE, FALSE) or numeric literals (3.1415, T\#5s), but also character string literals ('Hello world!', "black").

## Syntax declara-

tion

```
<scope> CONSTANT
    <identifier> : <data type> := <initial value> ;
END_VAR
<scope> : VAR | VAR INPUT | VAR STAT | VAR GLOBAL
<data type>: <eleme\overline{ntary data type | user defined data type |}
function block >
<initial value> : literal value | identifier | expression
```

Allowed initial values:

- Literal (example: TRUE, FALSE, 16\#FFFF_FFFF)
- Named constant that was declared at another location
- Simple expression composed of literals, also combined with simple operators, such as + - * Inputs or function calls cannot be specified as an initial value.


## Example

```
VAR_GLOBAL CONSTANT
    g_ciMAX_A : INT := 100;
    g_ciSPE\overline{CIAL : INT := g_ciMAX_A - 10;}
END_VAR
```

Constants are defined only for the declaration. The assignment of an initial value is required. Within an implementation, constants are only read and therefore always appear on the right of the assignment operator in a statement.

The constants are replaced with the initial value when the code is compiled. It also has to be possible to calculate the initial value at compile time.
Constants of structured or user-defined types are calculated not until runtime. Structured constants in programs or GVLs are calculated one time at program start. Structured constants in functions or methods are calculated every time the function or method is called. Therefore, the initialization of structured constants can depend on inputs or execute function calls.

See also

- $\Perp^{\Perp}$ Chapter 6.4.1.20.4.2 "BOOL constants" on page 2220
- $y^{*}$ Chapter 6.4.1.20.4.3 "Numeric constants" on page 2220
- ${ }^{4}$ Chapter 6.4.1.20.4.4 "REAL/LREAL constants" on page 2221
- ${ }^{4}$ Chapter 6.4.1.20.4.5 "String Constants" on page 2221
- ${ }^{4}$ Chapter 6.4.1.20.4.8 "Date and Time Constants" on page 2225
- $\quad$ Chapter 6.4.1.20.4.7 "TIME/LTIME Constant" on page 2223
- ${ }^{4}$ Chapter 6.4.1.20.4.9 "Typed literals" on page 2228

Variables You can declare variables as either local in the declaration part of a POU or in a global variable list. The allowed location of a variable depends on its data type.

See also

- \# Chapter 6.4.1.20.4.10 "Access to Variables in Arrays, Structures, and Blocks" on page 2228
- $y^{3}$ Chapter 6.4.1.20.4.11 "Bit Access in Variables" on page 2229

Other - © Chapter 6.4.1.20.4.12 "Addresses" on page 2231

- \& Chapter 6.4.1.20.4.13 "Functions" on page 2233
* Chapter 6.4.1.9.4 "Declaration of Variables" on page 1847
« Chapter 6.4.1.9.7 "Using input assistance" on page 1885
* Chapter 6.4.1.20.2.12 "Constant Variables - 'CONSTANT"' on page 2121
« Chapter 6.4.1.20.1.4.3 "ST expressions" on page 2050


## BOOL constants

BOOL constants are the truth values TRUE (1) and FALSE (0).
See also

- « \& Chapter 6.4.1.20.5.2 "Data type 'BOOL"' on page 2235


## Numeric constants

Numeric values can be binary, octal, decimal, and hexadecimal numbers. If an integer value is not a decimal number, then you must write its base followed by the number sign (\#) before the integer constant. You enter the hexadecimal digit values for the numbers 10 to 15 as usual with the letters A-F.

You can use an underscore within a numeric value.

## Examples:

| 14 | decimal number |
| :--- | :--- |
| $2 \# 1001 \_0011$ | binary number |
| $8 \# 67$ | octal number |
| $16 \#$ A | hexadecimal number |
| DINT\#16\#A1 | typed data type DINT\# and base $16 \#$ combined |

This type of numeric value can be BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, or LREAL.

Implicit conversions from "larger" to "smaller" types are not permitted. You cannot simply use a DINT variable as an INT variable. For this, you have to use a type conversion function.

See also

- $y^{4}$ Chapter 6.4.1.20.3 "Operators" on page 2129
- Hy Chapter 6.4.1.20.4.9 "Typed literals" on page 2228

As number constants basically are treated as integers, in divisions you must enter a constant in the format of a floating-point number in order not to loose the remainder. For example: Division 1/10 results in 0, division 1.0/10 results in 0.1 .

## REAL/LREAL constants

You can specify floating-point numbers as REAL and LREAL constants either in decimal notation or exponential notation with mantissa and exponent The decimal point serves as the decimal separator according to the International System of Units (English).
Syntax of expo-
nential notation

```
<significand> e | E <exponent>
exponent : -44..38 // REAL
exponent : -324..308 // LREAL
```

Example
Table 395: REAL literal

| 7.4 | Decimal number. 7,4 with a comma returns a compiler <br> error |
| :--- | :--- |
| $1 / 3.0$ | Decimal fraction for 0.333333343 <br> Note: In the case of division of integer types, the result <br> remains an integer type. In this case, the value is <br> rounded. For example, $1 / 3$ yields 0 as the result. |
| $1.64 \mathrm{e}+009$ | Exponential notation |
| $-3.402823 \mathrm{e}+38$ | Smallest number |
| $-1 \mathrm{E}-44$ | Largest negative number |
| $1.0 \mathrm{E}-44$ | Smallest positive number |
| $3.402823 \mathrm{e}+38$ | Largest number |

Table 396: LREAL literal

| $-1.7976931348623157 \mathrm{E}+308$ | Smallest number |
| :--- | :--- |
| $-4.94065645841247 \mathrm{E}-324$ | Largest negative number |
| $4.94065645841247 \mathrm{E}-324$ | Smallest positive number |
| $1.7976931348623157 \mathrm{E}+308$ | Largest number |

See also

- $\boldsymbol{y y}^{4}$ Chapter 6.4.1.20.5.4 "Data type 'REAL' / 'LREAL'" on page 2236


## String Constants

A string constant is a character string enclosed in single straight quotation marks. The characters are coded according to the character set specified in ISO/IEC 8859-1. Therefore, a string constant can include spaces and accented characters, as these belong to this character set. This is also referred to as a string literal, or simply a string.

Example: 'Hello world!'
When a dollar sign (\$) is in a string constant, the following two characters are interpreted as a hexadecimal code according to the coding in ISO/IEC 8859-1. The code also corresponds to ASCII code. In addition, please note the special cases.

Table 397: Hexadecimal code

| String with \$ code | Interpretation |
| :--- | :--- |
| '\$<8-bit code>' | 8-bit code: Two-digit hexadecimal number that is interpreted according to <br> ISO/IEC 8859-1. |
| $'^{\prime} \$ 41^{\prime}$ | A |
| '\$A9' | C |
| '\$40' | C |
| '\$0D' | Control character: Line break (corresponds to '\$R') |
| '\$0A' | Control character: New line (corresponds to '\$L' and '\$N') |

Table 398: Special cases

| String with \$ code | Interpretation |
| :---: | :---: |
| '\$L', ' \$l' | Control character: Line feed (corresponds to '\$0A') |
| '\$N', '\$n' | Control character: New line (corresponds to '\$0A') |
| '\$P', '\$p' | Control character: Form feed |
| '\$R', '\$r' | Control character: Line break (corresponds to '\$0D') |
| '\$T', '\$t' | Control character: Tab |
| '\$\$' | Dollar sign: § |
| '\$'' | Single straight quotation mark: ' |

```
Example Constant declaration
```

```
VAR CONSTANT
```

VAR CONSTANT
constA : STRING := 'Hello world';
constA : STRING := 'Hello world';
constB : STRING := 'Hello world \$21'; // Hello world!
constB : STRING := 'Hello world \$21'; // Hello world!
END_VAR

```
END_VAR
```


## Constant: UTF8\# String

With compiler version >=3.5.18.0, UTF-8 encoded string literals are possible as a single-byte string with base "STRING".
Syntax:

```
UTF8#' <string literal> '
```


## Constant declaration

```
VAR CONSTANT
    constA : STRING := UTF8#'aäoöuü';
    constB : STRING := UTF8#'Hello Allgäu $21'; // Hello Allgäu!
END_VAR
```

To ensure correct monitoring, the 'monitoring_encoding' := 'UTF-8' attribute can be added.

## NOTICE!

UTF-8 encoding only for project-wide configuration
A UTF-8 encoding is used if the project-wide compile option UTF8 encoding for STRING is enabled. Library functions and add-ons are then also oriented according to this setting.
If you use single UTF-8 encoded strings, then you have to make sure that they are interpreted correctly wherever they are used. For example, a string variable in the OPC server will be converted to UTF-8 before being transferred to a client if the setting is not selected. Values such as UTF8\#'äöu' would then be misinterpreted. Similar problems can arise when outputting strings in the visualization.

## TIME/LTIME Constant

You can use TIME constants to operate the standard timer modules. The constant has a size of 32 bits and a resolution in milliseconds.
In addition, the time constant LTTME is available as a time basis for high-resolution timers. The LTIME constant has a dimension of 64 bits and a resolution in nanoseconds.

## TIME constant

## Syntax

```
<time keyword> # <length of time>
<time keyword> : TIME | time | T | t
<length of time> : ( <number of days>d )? ( <number of hours>h )?
( <number of minutes>m )? ( <number of seconds>s )? (<number of
milliseconds>ms)? // ( ...)? Optional
```

The order of time units must not be changed. However, it is not required to specify all units. It is permitted to specify the units in uppercase.
Time units

- D | d: Days
- $\mathrm{H} \mid \mathrm{h}$ : Hours
- M | m: Minutes
- s | s:Seconds
- MS | ms: Milliseconds

```
Examples
Correct time
constants of VAR
an ST assign- timLength : TIME := T#14ms;
timLength : TIME := T#14ms;
ment
timLength1 : TIME := T#l00s12ms; // Overflow in the highest
unit is allowed.
    timLength2 : TIME := T#12h34m15s;
    timCompare : TIME;
    xIsOK: BOOL;
    timLongest := T#49D17H2M47S295MS; // 4294967295
END VAR
IF timLength < T#15MS THEN
    IF timCompare < timLength1 THEN
            xIsOK := TRUE;
        END_IF;
END IF
```

Table 399: Incorrect usage:

| timIncorrect $:=$ t\#5m68s; | Overflow at a lower position |
| :--- | :--- |
| timIncorrect1 $:=15 \mathrm{~ms} ;$ | Time marker T\# missing |
| timIncorrect $2:=$ t\#4ms $13 \mathrm{~d} ;$ | Incorrect order of time units |

## LTIME constant

## Syntax

```
<long time keyword> # <length of high resolution time>
<long time keyword> : LTIME | ltime
<length of high resolution time> : <length of time> ( <number of
microseconds>us )? ( <number of nanoseconds>ns )? // ( ...)? Optional
```

You can use the same units for LTIME constants as for TIME constants. You can also specify microseconds and nanoseconds because the specified time is calculated in higher time resolution. LTIME literals are treated internally as data type LWORD and therefore the value resolved in nanoseconds.

Additional time units

- US I us: Microseconds
- NS | ns: Nanoseconds


## Examples of correct usage of an ST <br> assignment:

```
PROGRAM PLC_PRG
VAR
    ltimLength := LTIME#1000d15h23m12s34ms2us44ns;
    ltimLength1 := LTIME#3445343m3424732874823ns;
END_VAR
```


## See also

- ${ }^{*}>$ Chapter 6.4.1.20.5.6 "Data Type 'TIME"' on page 2237
- $\&$ Chapter 6.4.1.20.4.8 "Date and Time Constants" on page 2225


## Date and Time Constants

32-bit date spec- Use the keyword DATE (D) to specify a date. ifications 'DATE'

## Syntax

```
<date keyword>#<year>-<month>-<day>
<date keyword> : DATE | date | D | d
<year> : 1970-2106
<month> : 1-12
<day> : 1-31
```

DATE literals are treated internally as data type DWORD, which corresponds to an upper limit of DATE\#2106-2-7.

## Example

```
PROGRAM PRG Date
VAR
    dateStart : DATE := DATE#2018-8-8;
    dateEnd : DATE := D#2018-8-31;
    dateCompare: DATE := date#1996-05-06;
    xIsDuringTheTime: BOOL;
    dateEarliest : DATE := d#1970-1-1; // = 0
    dateLatest : DATE := DATE#2106-2-7; // = 4294967295
END
    VAR
IF dateStart < dateCompare THEN
    IF dateCompare < dateEnd THEN
                xIsDuringTheTime := TRUE;
            END_IF;
END_IF
```

64-bit date spec- Use the keyword LDATE (LD) to specify a date.

## ifications

'LDATE'

## Syntax

```
<date keyword>#<year>-<month>-<day>
<date keyword> : LDATE | ldate | LD | ld
<year> : 1970-2262
<month> : 1-12
<day> : 1-31
```

LDATE literals are treated internally as data type LWORD, which corresponds to an upper limit of DATE\#2554-7-21.

```
Example
PROGRAM PRG_Ldate
VAR
    ldateStart : LDATE := LDATE#2018-8-8;
    ldateEnd : LDATE := ldate#2018-8-31;
    ldateCompare: LDATE := LD#1996-05-06;
    xIsDuringTheTime: BOOL;
    ldateEarliest : LDATE := ld#1970-1-1; // = 0
    ldateLatest : LDATE := LDATE#2262-4-10; // = 16#7FFF63888C620000
    lwValue: LWORD;
END_VAR
IF ldateStart < ldateCompare THEN
    IF ldateCompare < ldateEnd THEN
                xIsDuringTheTime := TRUE;
    END_IF;
END_IF
lwVālue := LDATE_TO_LWORD(ldateCompare);
```

32-bit date and Use the keyword DATE_AND_TIME (DT) to specify a date and time. time specifications 'DATE_AND_TIM
E'

## Syntax

```
<date and time keyword>#<date and time value>
<date and time keyword> : DATE_AND_TIME | date_and_time | DT | dt
<date and time value> : <year>-<month>-<day>-<\overline{hour>:<minute>:<second>}
<year> : 1970-2106
<month> : 1-12
<day> : 1-31
<hour> : 0-24
<minute> : 0-59
<second> : 0-59
```

DATE_AND_TIME literals are treated internally as data type DWORD. The time is processed in seconds and as a result can take on values from January 1, 1970 00:00 to February 7, 2106 06:28:15.

## Example

```
PROGRAM PLC_PRG
VAR
    dtDate : DATE AND TIME := DATE AND TIME#1996-05-06-15:36:30;
    dtDate1: DATE_AND_TIME := DT#1\overline{9}72-\overline{0}3-29-00:00:00;
    dtDate2: DATE_AND_TIME := DT#2018-08-08-13:33:20.5;
    dtEarliest : DATE AND TIME :=
DATE_AND_TIME#1979-1-1-00:00:00; // 0
    \overline{d}tLā̄est : DATE_AND_TIME := DATE_AND_TIME#2106-2-7-6:28:15; //
4294967295
END_VAR
```

64-bit date and time specifications 'LDATE_AND_TI ME'

## Syntax

```
<date and time keyword>#<long date and time value>
<date and time keyword> : LDATE AND TIME | ldate and time | LDT | ldt
<date and time value> : <year>-<mon\overline{th>-<day>-<hour>:<minute>:<second>}
<year> : 1970-2106
<month> : 1-12
<day> : 1-31
<hour> : 0-24
<minute> : 0-59
<second> : 0-59 LDATE_AND_TIME#2262-4-10-23:59:59.99999999
```

DATE_AND_TIME literals are treated internally as data type LWORD. The time is processed in seconds and as a result can take on values from January 1, 1970 00:00 to July 21, 2554 23:59:59.999999999.

## Example

```
PROGRAM PLC_PRG
VAR
    ldtDate : LDATE AND TIME := LDATE AND TIME#1996-05-06-15:36:30;
    ldtDate1: LDATE_AND_TIME := LDT#1972-03-29-00:00:00;
    ldtDate2: LDATE_AND_TIME := LDT#2018-08-08-13:33:20.5;
    dtEarliest : LDT := LDT#1979-1-1-00:00:00; // 0
    dtLatest : LDT := LDT#2266-4-10-23:59:59; // =
16#7FFF63888C620000
END_VAR
```

32-bit time spec- Use the keyword TIME_OF_DAY (TOD) to specify a time.

## ifications

'TIME_OF DAY'

## Syntax

```
<time keyword>#<time value>
<time keyword> : TIME_OF_DAY | time_of_day | TOD | tod
<time value> : <hour>:<minute>:<second>
<hour> : 0-23
<minute> : 0-59
<second> : 0.000-59.999
```

You can also specify fractions of a second. TIME_OF DAY literals are treated internally as DWORD and the value is resolved in milliseconds.

## Examples

```
PROGRAM POU
VAR
    todClockTime : TIME_OF DAY := TIME_OF DAY#15:36:30.123;
    todEarliest : TIME_OF_DAY := TIME_OF_DAY#0:0:0.000;
    todLatest : TOD := TOD#23:59:59.999;
END_VAR
```

64-bit time spec- Use the keyword LTIME_OF_DAY (LTOD) to specify a time.
ifications
'LTIME_OF_DAY'

## Syntax

```
<time keyword>#<time value>
<time keyword> : LTIME_OF_DAY | ltime_of_day | LTOD | ltod
<time value> : <hour>:<minute>:<second>
<hour> : 0-23
<minute> : 0-59
<second> : 0.000-59.999999999
```

You can also specify fractions of a second. LTIME_OF_DAY literals are treated internally as LWORD and the value is resolved in nanoseconds.

## Examples

```
PROGRAM POU
VAR
    ltodClockTime : LTIME OF DAY := TIME OF DAY#15:36:30.123456789;
    todEarliest : TIME_OF_DAY := TIME_OF_\overline{DAY# 0:0:0;}
    todLatest : TOD := TOD#23:59:59.999999999;
END VAR
```

See also

- $\boldsymbol{y y}^{4}$ Chapter 6.4.1.20.5.8 "Date and Time Data Types" on page 2238


## Typed literals

With the exception of REAL/LREAL constants (LREAL is always used here), CODESYS uses the smallest possible data type when calculating with IEC constants. If you want to use another data type, then you can use typed literals without having to declare the constants explicitly. When doing this, provide the constants with a prefix that indicates the type.

## Syntax:

```
<type>#<literal>
```

<type> defines the desired data type; possible values: BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT, DWORD, REAL, LREAL. You must capitalize the entire type name.
<literal> defines the constants. The entry must match the data type defined in <Type>.

```
Example: var1 := DINT#34;
```

If CODESYS cannot convert the constant into the target type without data loss, then an error message is issued.
You can use typed constants wherever you can use normal constants.

## Access to Variables in Arrays, Structures, and Blocks

Syntax for access to

- Two-dimensional array component: <array name> [ <1st dimension> , <2nd dimension> ]
- Structural variable: <structure name> . <component name>
- Function block and program variable: <function block name> | <program name>. <variable name>

See also

- ${ }^{4}$ Chapter 6.4.1.20.5.15 "Data Type 'ARRAY'" on page 2247
- ${ }^{*}>$ Chapter 6.4.1.20.5.17 "Structure" on page 2261
- $\Leftrightarrow$ Chapter 6.4.1.21.2.21.3 "Object 'Function Block'" on page 2479
- $\boldsymbol{y}^{\boldsymbol{y}}$ Chapter 6.4.1.21.2.21.2 "Object 'Program'" on page 2478


## Bit Access in Variables

## 0 <br> NOTICE! <br> Implement concurrent bit access by two tasks only if the processor can execute bit access directly on the memory. All x86 and x64 systems have commands for bit access in memory. Systems such as ARM and PPC cannot access bits directly in the memory. <br> If two tasks execute bit access simultaneously, even though the processor cannot perform bit access directly in the memory, then proceed as follows. Use a semaphore (SysSemEnter) or a similar technique to prevent competing bit access. However, it is best to execute the bit access within a task.

With index access, individual bits can be addressed in integer variables. Using a structure variable or a function block instance, individual bits can be addressed symbolically.

Index access to You can address individual bits in integer variables. To do this, append the variable with a dot bits integer vari- and the index of the addressed bit. The bit-index can be given by any constant. Indexing is ables 0 -based.
Syntax <integer variable name> . <index>
<integer data type> = BYTE | WORD | DWORD | LWORD | SINT | USINT | INT | UINT | DINT | UDINT | LINT | ULINT

Example In the program, the third bit of the variable $w A$ is set to the value of variable $x B$. The constant c_usiENABLE acts as an index to access the third bit of the variable iX.
Index access

```
        PROGRAM PLC_PRG
        VAR
            WA : WORD := 16#FFFF;
            xB : BOOL := 0;
        END_VAR
        // Index access in an integer variable
        wA.2 := xB;
Result: wA = 2#1111_1111_1111_1011 = 16#FFFB
VAR_GLOBAL CONSTANT
        gc_usiENABLE : USINT := 2;
    END_VA\overline{R}
    PROGRAM PLC_PRG
VAR
            iX : INT := 0;
    END_VAR
    // Constant as index
    iX.gc_usiENABLE := TRUE; // Third bit in iX is set TRUE
Result: iX = 4
```

Constant as
index // GVL declaration

Symbolic bit With the BIT data type, you can combine individual bits into a structure and then access them access in struc- individually. Then the bit is addressed with the component name. ture variables

## Example

Type declaration of the structure:

```
TYPE S CONTROLLER :
STRUCT
    bitOperationEnabled : BIT;
                        bitSwitchOnActive : BIT;
                        bitEnableOperation : BIT;
                        bitError : BIT;
                        bitVoltageEnabled : BIT;
                bitQuickStop : BIT;
                bitSwitchOnLocked : BIT;
            bitWarning : BIT;
END_STRUCT
END_TYPE
PROGRAM PLC_PRG
VAR
                ControlDriveA : S_CONTROLLER;
    END_VAR
    // Symbolic bit access to bitEnableOperation
ControlDriveA.bitEnableOperation := TRUE;
```

Declaration
and write
access to a bit:

Symbolic bit access in function block instances

```
Example
```

```
FUNCTION_BLOCK FB_Controller
```

FUNCTION_BLOCK FB_Controller
VAR_INPUT
VAR_INPUT
bitSwitchOnActive : BIT;
bitSwitchOnActive : BIT;
bitEnableOperation : BIT;
bitEnableOperation : BIT;
bitVoltageEnabled : BIT;
bitVoltageEnabled : BIT;
bitQuickStop : BIT;
bitQuickStop : BIT;
bitSwitchOnLocked : BIT;
bitSwitchOnLocked : BIT;
END_VAR
END_VAR
VAR OUTPUT
VAR OUTPUT
bitOperationEnabled : BIT;
bitOperationEnabled : BIT;
bitError : BIT;
bitError : BIT;
bitWarning : BIT;
bitWarning : BIT;
END_VAR
END_VAR
VAR
VAR
END VAR
END VAR
;
;
PROGRAM PLC PRG
PROGRAM PLC PRG
VAR
VAR
fbController : FB_Controller;
fbController : FB_Controller;
END VAR
END VAR
// S̄ymbolic bit access to bitSwitchOnActive
// S̄ymbolic bit access to bitSwitchOnActive
fbController(bitSwitchOnActive := TRUE);

```
fbController(bitSwitchOnActive := TRUE);
```


## See also

- « Chapter 6.4.1.20.5.3 "Integer data types" on page 2235
- «S "Symbolic bit access in structure variables" on page 2262
- \& Chapter 6.4.1.20.5.11 "Data Type 'BIT"' on page 2243


## Addresses

Syntax:

```
%<memory area prefix> ( <size prefix> )? <memory position>
<memory area prefix> : I | Q | M
<size prefix> : X | B | W | D
<memory position> : <number> ( .<number> )* // Depends on the target
system
```

When defining an address, you use specific character strings to express memory position and size. An address is marked with the percent sign (\%), followed by the memory range prefix, the optional size prefix, and the memory range position. The numbering that you use for addressing the memory position depends on the target system.

| Memory <br> Range <br> Prefix |  |
| :--- | :--- |
| I | Input memory range for "Inputs" <br> For physical inputs via input drivers, "Sensors" |
| Q | Output memory range for "Outputs" <br> For physical outputs via output drivers, "Actuators" |
| M | Flag memory range |


| Size Prefix | Data Type | Data Width |
| :--- | :--- | :--- |
| No size prefix |  | Single bit |
| X |  | Single bit |
| B | BYTE | 8 bits |
| W | WORD | 16 bits |
| D | DWORD | 32 bits |

Examples

| $\%$ QX7.5 <br> $\% Q 7.5$ | Single bit address of the output bit 7.5 |
| :--- | :--- |
| $\%$ IW215 | Word address of the input word 215 |
| $\%$ QB7 | Byte address of the output byte 7 |
| $\%$ MD48 | Address of a double word at memory position 48 in <br> flag memory |
| \%IW2.5.7.1 | Word address of an input word; interpretation <br> dependent on the current controller configuration |
| VAR wVar AT \%IW0 : WORD; END_VAR | Variable declaration with address information of an <br> input word |
| VAR xActuator AT \%QW0 : BOOL; END_VAR | Boolean variable declaration <br> Note: For Boolean variables, one byte is allocated <br> internally if a single bit address is not specified. A <br> change in the value of xActuator affects the range <br> from QX0. 0 to QX0. 7. |
| VAR xSensor AT IX7.5 : BOOL; END_VAR | Boolean variable declaration with explicit specification <br> of a single bit address. On access, only the input bit <br> 7.5 is read. |

Memory posi- Make sure that the address is valid as follows:
tion
To map a valid address in an application, you must know the required position (applicable memory range) in the process image: input memory range (I), output memory range (Q), and flag memory range $(M)$ - see above. Furthermore, you have to specify the required size prefix: bit, BYTE, WORD, DWord (see above: X, B, W, D)
The current device configuration and device settings (hardware structure, device description, I/O settings) play a decisive part. Note specifically the differences in the interpretation of bit addresses for devices with "byte addressing mode" and devices with "word-oriented IEC addressing mode". For example, in a byte addressing device, the number before the point of bit address $\%$ IX5.5 addresses byte 5 . On the other hand, in a word-addressed device, it addresses word 5 . In contrast, addressing with a word or byte address is independent of the device type: with $\%$ IW 5 always word 5 is addressed and with byte address $\%$ IB5 always byte 5 . Regardless of size and addressing mode, you can address different memory cells therefore with the same address information.
The following table shows the comparison of byte addressing and word-oriented IEC addressing for bits, bytes, words, and double words. It also shows the overlapping memory ranges that are present in the case of byte addressing (see also the example below the table).
Regarding syntax, note that the IEC addressing mode is always word-oriented. In this case, the word number is located before the point and the bit number ofter the point.

| DWords / Words |  |  |  | Bytes | X (bits) |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| byte addressing |  | word oriented IEC addressing |  |  | byte addressing |  |  | word oriented IEC addressing |  |  |
| DO | W0 | DO | W0 | B0 | $\times 0.7$ |  | $\times 0.0$ | $\times 0.7$ | ... | $\times 0.0$ |
| D1 | W1 |  |  | B1 | X1.7 |  | $\times 1.0$ | $\times 0.15$ | ... | $\times 0.8$ |
| ... | W2 |  | W1 | B2 | ... |  |  | X1.7 | ... | X1.0 |
|  | W3 |  |  | B3 |  |  |  | X1.15 | . | X1.8 |
|  | W/4 | D1 | W2 | B4 |  |  |  |  |  |  |
|  | ... |  |  | B5 |  |  |  |  |  |  |
|  |  |  | W3 | B6 |  |  |  |  |  |  |
|  |  |  |  | B7 |  |  |  |  |  |  |
|  |  | D2 | W4 | B8 |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| D(n-3) |  | D(n/4) |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  | W(n-1) |  | W(n/2) |  |  |  |  |  |  |  |
|  |  |  |  | Bn | Xn. 7 | ... | Xn. 0 | $\mathrm{X}(\mathrm{n} / 2) .15$ | $\cdots$ | $\mathrm{X}(\mathrm{n} / 2) .8$ |

$\mathrm{n}=$ byte number

Example of memory range overlapping in the case of the byte addressing mode

D0 contains $\mathrm{B} 0-\mathrm{B} 3$, W0 contains B0 and B1, W1 contains B1 and B2, and W2 contains B2 and B3. Consequently, in order to avoid overlap, you must not use W1 (also D1, D2, and D3) for addressing.

See also

- ${ }^{*}$ Chapter 6.4.1.9.12.3 "AT declaration" on page 1902


## Functions

Example: Result :=Fct(7) + 3;

TIME() function This function yields the time (in milliseconds) that has elapsed since system boot.
The data type is TIME.

## Example in ST:

```
systime := TIME();
```

See also

- ${ }^{*} \boldsymbol{y}$ Chapter 6.4.1.21.2.21.4 "Object 'Function'" on page 2482


### 6.4.1.20.5 Data Types

## General

In the programming, a variable is identified by its name and has an address in the memory of the target system. Accordingly, variable names are identifiers under which the allocated memory is addressed. The size of the variable is determined by its data type. This determines how much memory is reserved for the variable and how the values in memory are to be interpreted. The data type also determines which operators are allowed.
In CODESYS, there is also the capability of instantiating function blocks. Function block instances then use memory like variables do. The memory requirement is determined by the function block.
The following groups of data types are available.

Standard data A standard data type (or standard data type) is an elementary data type or a string data type. types

```
<standard data type> : __UXINT | XINT | __XWORD | BIT | BOOL |
BYTE | DATE | DATE_AND_\overline{TIME | DINT | DT | \overline{DWORD | INT | LDATE |}}\mathbf{|}=\mp@code{L}
LDATE AND TIME | LDT | LINT | LREAL | LTIME | LTOD | LWORD | REAL |
SINT TSTRING | TIME | TOD | TIME_OF_DAY | UDINT | UINT | ULINT | USINT
| WORD | WSTRING
```

See also

- ${ }^{4}$ Chapter 6.4.1.20.5.11 "Data Type 'BIT"' on page 2243
-     * Chapter 6.4.1.20.5.2 "Data type 'BOOL'" on page 2235
- $y^{*}$ Chapter 6.4.1.20.5.3 "Integer data types" on page 2235
- ${ }^{\circ}$, Chapter 6.4.1.20.5.12 "Special Data Types '__UXINT', __XINT, and '_XWORD" on page 2243
- ${ }^{\text {® }}$, Chapter 6.4.1.20.5.4 "Data type 'REAL' / 'LREAL'" on page 2236
- $\quad \stackrel{y}{c}$ Chapter 6.4.1.20.5.5 "Data Type 'STRING'" on page 2237
- $\Leftrightarrow$ Chapter 6.4.1.20.5.10 "Data type 'WSTRING'" on page 2242
- $\Leftrightarrow$ Chapter 6.4.1.20.5.6 "Data Type 'TIME"' on page 2237
- $\Leftrightarrow$ Chapter 6.4.1.20.5.7 "Data Type 'LTIME"" on page 2237
- $\sum^{4}$ Chapter 6.4.1.20.5.8 "Date and Time Data Types" on page 2238


## Extensions of <br> the IEC 61131-3 <br> standard

See also

- ⓨ Chapter 6.4.1.20.5.11 "Data Type 'BIT'" on page 2243
- $\quad 4$ Chapter 6.4.1.20.5.13 "Pointers" on page 2243
- $\Leftrightarrow$ Chapter 6.4.1.20.5.20 "Data type 'UNION"" on page 2268
- ※ \& Chapter 6.4.1.20.5.16 "Data Type '__VECTOR"" on page 2253

User-defined You can declare your own data types which are based on the default predefined data types or data types existing data types.

These kinds of data types are called user-defined or user-specific. The data types are either organized as its own DUT object or declared within the declaration part of a programming object. Moreover, they are differentiated according to their purpose and syntax.

| User-Defined <br> Data Type | Declaration | See also |
| :--- | :--- | :--- |
| Alias | DUT object | ๕ Chapter 6.4.1.20.5.19 "Alias" on page 2267 |
| Arrays | Programming <br> object | $\Perp$ Chapter 6.4.1.20.5.15 "Data Type 'ARRAY'" <br> on page 2247 |
| Enumeration | DUT object, pro-- <br> gramming object |  Chapter 6.4.1.20.5.18 "Enumerations" on page 2263 |


| User-Defined Data Type | Declaration | See also |
| :---: | :---: | :---: |
| Reference | Programming object | * Chapter 6.4.1.20.5.14 "Reference" on page 2245 |
| Pointer | Programming object | ${ }^{\text {® }}$ Chapter 6.4.1.20.5.13 "Pointers" on page 2243 |
| Structure | DUT object | ® Chapter 6.4.1.20.5.17 "Structure" on page 2261 |
| Subrange type | Programming object | ②) Chapter 6.4.1.20.5.21 "Subrange types" on page 2268 |
| Union | DUT object | ${ }^{\star}$ Chapter 6.4.1.20.5.20 "Data type 'UNION"' on page 2268 |
| Vector | DUT object | « Chapter 6.4.1.20.5.16 "Data Type '__VECTOR"' on page 2253 |

## NOTICE!

Note the recommendations for naming an identifier.

See also

- (n) Chapter 6.4.1.20.7 "Identifiers" on page 2327


## Data type 'BOOL'

| Data Type | Values | Memory |
| :--- | :--- | :--- |
| BOOL | TRUE (1), FALSE (0) | 8 bit |

See also

- $\quad$ \& Chapter 6.4.1.20.4.2 "BOOL constants" on page 2220

Integer data types
CODESYS provides the following integer data types.

| Data Type | Lower Limit | Upper Limit | Memory |
| :--- | :--- | :--- | :--- |
| BYTE | 0 | 255 | 8 bit |
| WORD | 0 | 65535 | 16 bit |
| DWORD | 0 | 4294967295 | 32 bit |
| LWORD | 0 | $2^{64-1}$ | 64 bit |
| SINT | -128 | 127 | 8 bit |
| USINT | 0 | 255 | 8 bit |
| INT | -32768 | 32767 | 16 bit |
| UINT | 0 | 65535 | 16 bit |
| DINT | -2147483648 | 2147483647 | 32 bit |
| UDINT | 0 | 4294967295 | 32 bit |


| Data Type | Lower Limit | Upper Limit | Memory |
| :--- | :--- | :--- | :--- |
| LINT | $-2^{63}$ | $2^{63}-1$ | 64 bit |
| ULINT | 0 | $2^{64}-1$ | 64 bit |

## - NOTICE! <br> Information can be lost when converting from larger to smaller types.

See also

- « Chapter 6.4.1.20.4.3 "Numeric constants" on page 2220

Data type 'REAL' / 'LREAL'
The data types REAL and LREAL are floating-point types according to IEEE 754. They are necessary when using decimal numbers and floating-point numbers in decimal notation or exponential notation.

## Example

```
PROGRAM PLC_PRG
VAR
    rMax: REAL := 3.402823E+38; // Largest number
    rPosMin : REAL := 1.0E-44; // Smallest positive number
    rNegMax: REAL := -1.0E-44; // Largest negative number
    rMin: REAL := -3.402823E+38; // Smallest number
    lrMax: LREAL := 1.7976931348623157E+308; // Largest number
    lrPosMin : LREAL := 4.94065645841247E-324; // Smallest positve
number
    lNegMax: LREAL := -4.94065645841247E-324; // Largest negative
number
    lrMin: LREAL := -1.7976931348623157E+308; // Smallest number
END_VAR
```


## NOTICE!

Support for the LREAL data type depends on the target device in use. Refer to the respective documentation as to whether or not the 64-bit type LREAL is converted to REAL or remains as LREAL when compiling the application. Conversion may result in the loss of information.

## NOTICE!

If the value of the REAL/LREAL number is outside of the value range of the integer, then an undefined result is yielded from a data type conversion from REAL or LREAL to SINT, USINT, INT, UINT, DINT, UDINT, LINT, or ULINT. The result depends on the target system. An exception error is also possible. To get code that is independent of the target system, the application must catch value range violations.
If the REAL/LREAL number is within the value range of the integer data type, then the conversion operates the same way on all systems.

See also

- \# Chapter 6.4.1.20.4.4 "REAL/LREAL constants" on page 2221


## Data Type 'STRING'

A variable of data type STRING can have contain any character string. The amount of memory that is reserved during a declaration refers to characters and is shown in parentheses or brackets. If a size is not defined, then CODESYS allocates 80 characters by default.
As a rule, CODESYS does not limit the string length. However, the string function processes lengths of 1-255 only. If a variable is initialized with a string that is too long for the data type, then CODESYS truncates the string accordingly from the right.

## NOTICE!

The memory required for a STRING variable is always one byte per character plus one additional byte (for example, 81 bytes for a "STRING(80)" declaration).

```
Example of a
string declara-
tion with }3
characters:
```

```
str : STRING(35):= 'This is a String';
```

```
str : STRING(35):= 'This is a String';
```

See also

- ${ }^{*}$ Chapter 6.4.1.20.4.5 "String Constants" on page 2221
- $\sum_{\text {a }}$ Chapter 6.4.1.20.5.10 "Data type 'WSTRING"' on page 2242


## Data Type 'TIME'

The data type is treated internally as DWORD. TIME is resolved in milliseconds.

| Data type | Lower limit | Upper limit | Storage space | Resolution |
| :--- | :--- | :--- | :--- | :--- |
| TIME | T\#0d0h0m0s0ms | T\#49d17h2m47s29 <br> 5 ms | 32 bit | Milliseconds |

See also

- « Chapter 6.4.1.20.5.7 "Data Type 'LTIME"' on page 2237
- \& Chapter 6.4.1.20.4.7 "TIME/LTIME Constant" on page 2223
- \& Chapter 6.4.1.20.4.8 "Date and Time Constants" on page 2225


## Data Type 'LTIME'

Data Type You can use the data type LTIME as a time base for high-resolution timer. A high-resolution 'LTIME' timer has a resolution in nanoseconds.

| Data Type | Lower Limit | Upper Limit | Memory |
| :--- | :--- | :--- | :--- |
| LTIME | LTIME\#ONS | LTIME\#213503D23H34M3 <br> 3S709MS551US615NS | 64 bits |

## Syntax:

LTIME\#<long time declaration>
The time declaration can include units of time that apply for the TIME constant as well as:

- "US": microseconds
- "NS": nanoseconds

```
Example: LTIME1 := LTIME#1000D15H23M12S34MS2US44NS
```


## See also

- \& Chapter 6.4.1.20.5.6 "Data Type 'TIME"' on page 2237
- © Chapter 6.4.1.20.3.41 "Time Conversion" on page 2182


## Date and Time Data Types

The data types DATE, DATE_AND_TIME (DT), and TIME_OF_DAY (TOD) are handled internally like a DWORD (32-bit value).

The data types LDATE, LDATE_AND_TIME (LDT), and LTIME_OF_DAY (LTOD) are treated internally like an LWORD (64-bit value).

The values of these data types are measured in seconds, milliseconds, and nanoseconds since 01/01/1970.

| Data Type | Lower Limit | Upper Limit | Memory | Resolution |
| :---: | :---: | :---: | :---: | :---: |
| DATE | DATE\#1970-01-01 <br> D\#1970-01-01 | DATE\#2106-02-07 <br> D\#2106-02-07 | 32-bit | Seconds (although only the day is displayed) |
| $\begin{aligned} & \text { DATE_AND_TIME } \\ & \text { DT } \end{aligned}$ | $\begin{aligned} & \text { DATE_AND_TIME\#1970-1 } \\ & -1-0: 0: 0 \\ & \text { DT\#1970-1-1-0:0:0 } \end{aligned}$ | $\begin{aligned} & \text { DATE_AND_TIME\#2106-0 } \\ & 2-07-06: 28: 15 \\ & \text { DT\#2106-02-07-06:28: } \\ & 15 \end{aligned}$ | 32-bit | Seconds |
| $\begin{aligned} & \text { TIME_OF_DAY } \\ & \text { TOD } \end{aligned}$ | $\begin{aligned} & \text { TIME_OF_DAY\#0:0:0 } \\ & \text { TOD\#0:0:0 } \end{aligned}$ | ```TIME_OF_DAY#23:59:59 . }99 TOD#23:59:59.999``` | 32-bit | Milliseconds |
| LDATE | LDATE\#1970-1-1 <br> LD\#1970-1-1 | LDATE\#2554-7-21 <br> LD\#2554-7-21 | 64-bit | Nanoseconds (although only the day is displayed) |
| $\begin{aligned} & \text { LDATE_AND_TIME } \\ & \text { LDT } \end{aligned}$ | LDATE_AND_TIME\#1970-1-1-0:0:0 <br> LDT\#1970-1-1-0:0:0 | $\begin{aligned} & \text { LDATE_AND_TIME\#2554- } \\ & 7-21: 23: 59: 59.999999 \\ & 99 \\ & \text { LDT\#2554-7-21-23:59: } \\ & 59.99999999 \end{aligned}$ | 64-bit | Nanoseconds |
| $\begin{aligned} & \text { LTIME_OF_DAY } \\ & \text { LTOD } \end{aligned}$ | $\begin{aligned} & \text { LTIME_OF_DAY\#0:0:0 } \\ & \text { LTOD\#0:0:0 } \end{aligned}$ | $\begin{aligned} & \text { LTIME_OF DAY\#23:59:5 } \\ & 9.999999999 \\ & \text { LTOD\#23:59:59.999999 } \\ & 999 \end{aligned}$ | 64-bit | Nanoseconds |

## Example

```
VAR
    //Date
    dateBottom : DATE := DATE#1970-1-1;
    dateTop : DATE := DATE#2106-2-7;
    dateAppointment : DATE := D#2020-2-7; // D prohibited
    //Date and time
    dtBottom : DATE AND TIME := DATE AND TIME#1970-1-1-0:0:0;
    dtTop : DT := \overline{DATE_}\mp@subsup{}{~}{A}ND_TIME#2106-02-\overline{0}7-06:28:15;
    dtAppointment : DT := DT#2020-2-7-12:55:1.234;
    //Time of day
    todBottom : TIME_OF_DAY := TIME_OF_DAY#0:0:0;
    todTop : TOD := 'TIME OF DAY#23:缼9:\overline{59.999;}
    todAppointment : TOD \overline{:= TOD#12:3:4.567;}
    // Long date
    ldateBottom : LDATE := LDATE#1970-1-1;
    ldateTop : LDATE := LDATE#2106-2-7;
    ldateAppointment : LDATE := LD#2020-2-7; // LD prohibited
    // Long date and time
    ldtBottom : LDATE_AND_TIME := LDATE_AND_TIME#1970-1-1-0:0:0;
```



```
    ldtAppointment : LDT := LDT#2020-2-7-12:55:1.234567891;
    //Long time of day
    ltodBottom : LTIME_OF_DAY := LTIME_OF_DAY#0:0:0;
```



```
    ltodAppointment : LTOD := L\overline{TOD#12:3:4.567890123;}
END_VAR
```

See also

- ${ }^{\text {® }}$ Chapter 6.4.1.20.4.8 "Date and Time Constants" on page 2225


## Data Type 'ANY' and 'ANY_<type>'

The data types ANY or ANY_<type> are used in interfaces of functions, function blocks, or methods in order to type input parameters whose type is unknown or unspecified: The input variables (VAR_INPUT) have a generic data type.
The compiler replaces the type of input variable internally with the data structure described below, whereby the value is not passed directly. Instead, a pointer is passed to the actual value so only a variable can be passed. Therefore, the data type is only specified when it is called. As a result, calls of such POUs can be made using arguments which each have different data types.

Literals, replaced constants, and results of function calls or expressions cannot be passed to input variables (VAR_IN_OUT).

Internal data structure for 'ANY' and 'ANY_<type>'

When code is compiled, the input variables are typed internally with ANY data type by the following structure. When the POU is called (at runtime), the argument is passed to a reference parameter.

```
TYPE AnyType :
STRUCT
    // the type of the actual parameter
    typeclass : __SYSTEM.TYPE_CLASS ;
    // the pointer to the-actual parameter
    pvalue : POINTER TO BYTE;
    // the size of the data, to which the pointer points
    diSize : DINT;
END STRUCT
END_TYPE
```

You can access the input variable within the POU via this structure by means of this structure, and for example query the passed value.

Example
This compares whether or not two input variables have the same type and the same value.

```
FUNCTION funGenericCompare : BOOL
VAR_INPUT
    any1 : ANY;
    any2 : ANY;
END_VAR
VAR
    pTest : POINTER TO ARRAY [0..100] OF POINTER TO DWORD;
    diCount: DINT;
END_VAR
pTest := ADR(any1);
Generic_Compare := FALSE;
IF any1.typeclass <> any2.typeclass THEN
    RETURN;
END_IF
IF anyl.diSize <> any2.diSize THEN
    RETURN;
END_IF
// B}yte comparison
FOR iCount := O TO anyl.diSize-1 DO
    IF any1.pvalue[iCount] <> any2.pvalue[iCount] THEN
        RETURN;
    END_IF
END_FOR
Genēric_Compare := TRUE;
RETURN;
// END_FUNCTION
```


## Declaration

The syntax descriptions refer to a POU with exactly one parameter (an input variable).

## Syntax

```
FUNCTION | FUNCTION_BLOCK | METHOD <POU name> ( : <return data
type> )?
VAR_INPUT
    <input variable name> : <generic data type>;
END_VAR
<generic data type> = ANY | ANY_BIT | ANY_DATE | ANY_NUM | ANY_REAL |
ANY_INT | ANY_STRING
```


## Example

```
FUNCTION funComputeAny : BOOL
VAR_INPUT
    anyInput1 : ANY; // valid data type see table
END VAR
// END_FUNCTION
FUNCTION_BLOCK FB_ComputeAny
VAR_INPUT
    anyInput1 : ANY;
END VAR
// END_FUNCTION_BLOCK
FUNCTION_BLOCK FB_ComputeMethod
METHOD methComputeAnny : BOOL
VAR_INPUT
    anyInput1 : ANY INT; // valid data types are SINT, INT, DINT,
LINT, USINT, UINT, UDINT, ULINT
END_VAR
//ENND_METHOD
```

With compiler versions > 3.5.1.0, the generic IEC data types in the table are supported.

The table represents the hierarchy of the generic data types and provides information as to which generic data type of the formal parameter (declaration) allows which elementary data types of the argument (call).

| Generic data type in the case of a formal parameter |  |  | Permitted elementary data type in the case of an actual parameter (argument) |
| :---: | :---: | :---: | :---: |
| ANY | ANY_BIT |  | - BYTE <br> - WORD <br> - DWORD <br> - LWORD |
|  | ANY_DATE |  | - DATE <br> - DATE_AND_TIME, DT <br> - TIME_OF_DAY, TOD <br> - LDATE <br> - LDATE_AND_TIME, LDT <br> - LTIME OF DAY, LTOD |
|  | ANY_NUM | ANY_REAL | REAL, LREAL |
|  |  | ANY_INT | USINT, UINT, UDINT, ULINT SINT, INT, DINT, LINT |
|  | ANY_STRING |  | STRING, WSTRING |

Call The syntax descriptions refer to a POU with exactly one parameter, to which an argument is passed. As a result, the data type of the argument specifies the generic data type of the input variable. For example, arguments of the type BYTE, WORD, DWORD, LWORD can be passed to a type ANY_BIT input variable.

```
Syntax of func-
tion call <variable name> := <function name> ( <argument name> );
<argument name> : variable with valid data type
Syntax of func-
tion block call
Syntax of
method call <function block name> . <method name> ( <input variable name> :=
<argument name> );
Example
```

```
PROGRAM PLC_PRG
```

PROGRAM PLC_PRG
VAR
VAR
byValue : BYTE := 16\#AB;
byValue : BYTE := 16\#AB;
iValue : INT := -1234;
iValue : INT := -1234;
xResultByte : BOOL;
xResultByte : BOOL;
xResultInt : BOOL;
xResultInt : BOOL;
fbComputeAnyByte : FB_ComputeAny;
fbComputeAnyByte : FB_ComputeAny;
fbComputeAnyInt : FB_ComputeAny;
fbComputeAnyInt : FB_ComputeAny;
fbComputeM1 : FB ComputeMethod;
fbComputeM1 : FB ComputeMethod;
fbComputeM2 : FB_ComputeMethod;
fbComputeM2 : FB_ComputeMethod;
byN : BYTE := 1;
byN : BYTE := 1;
wBitField1 : WORD := 16\#FFFF;
wBitField1 : WORD := 16\#FFFF;
wBitField2 : WORD := 16\#0001;
wBitField2 : WORD := 16\#0001;
xInit : BOOL;
xInit : BOOL;
xResult : BOOL;
xResult : BOOL;
END_VAR
END_VAR
xResultByte := funComputeAny(byValue);
xResultByte := funComputeAny(byValue);
xResultInt := funComputeAny(iValue);
xResultInt := funComputeAny(iValue);
xResult := funGenericCompare(wBitField1, wBitField2);
xResult := funGenericCompare(wBitField1, wBitField2);
fbComputeAnyByte(anyInput1 := byValue);
fbComputeAnyByte(anyInput1 := byValue);
fbComputeAnyInt(anyInput1 := iValue);
fbComputeAnyInt(anyInput1 := iValue);
fbComputeM1.methComputeAnny(anyInput1 := byValue);
fbComputeM1.methComputeAnny(anyInput1 := byValue);
fbComputeM2.methComputeAnny(anyInput1 := iValue);
fbComputeM2.methComputeAnny(anyInput1 := iValue);
// END_PRG

```
    // END_PRG
```


## Data type 'WSTRING'

The data type WSTRING is interpreted in Unicode format as opposed to the data type STRING (ASCII). As a result of this coding, the number of displayed characters for WSTRING depends on the characters. A length of 10 for WSTRING means that the length of the WSTRING can take a maximum of 10 WORDs. However, for some characters in Unicode, multiple WORDS are required for coding a character so that the number of characters do not have to correspond to the length of the WSTRING (10 in this case). The data type requires 1 WORD of memory per character plus 1 WORD of extra memory. Each STRING requires only 1 byte. The data type WSTRING is terminated with a 0 .

Example: wstr : WSTRING := "This is a WString";

## See also

- ${ }^{\star}$ Chapter 6.4.1.20.5.5 "Data Type 'STRING'" on page 2237
- ${ }^{4}$ Chapter 6.4.1.20.4.5 "String Constants" on page 2221


## Data Type 'BIT'

The data type BIT is valid only in structures for the declaration of structure members or in a function block for the declaration of variables. A BIT variable can have the values TRUE (1) and FALSE (0). In this case, the variable requires exactly one bit of memory.
As a result, you can symbolically address individual bits by a name. BIT variables that are declared in succession are bundled in bytes. In this way, you can optimize memory use as opposed to BOOL types, which reserve 8 bits each. On the other hand, bit access is significantly more time-consuming. Therefore, you should use the BIT data type only when you need to define data in a predefined format.

See also

- ${ }^{4}$ ) Chapter 6.4.1.20.4.11 "Bit Access in Variables" on page 2229
- $y^{*}$ Chapter 6.4.1.20.5.17 "Structure" on page 2261


## Special Data Types '__UXINT', __XINT, and '__XWORD'

Variables with these data types are converted to a platform-compliant data type, depending on the target system.
CODESYS supports systems with address registers of 32-bit and 64-bit widths. For making the IEC code as independent from the target system as possible, you use the pseudo data types UXINT, __XINT, and __XWORD. The compiler checks which target system types are current and then converts these data types into the appropriate standard data types.
Moreover, type conversion operators are provided for variables of these data types.

|  | Type conversion on 64-bit <br> platforms | Type conversion on 32-bit <br> platforms |
| :--- | :--- | :--- |
| $\ldots$ UXINT | ULINT | UDINT |
| $\ldots$ XINT | LINT | DINT |
| $\quad$ XWORD | LWORD | DWORD |

See also

- $\quad$ \& Chapter 6.4.1.20.3.38 "Integer Conversion" on page 2159
- ${ }^{\text {• }}$ Chapter 6.4.1.20.3.36 "Overloading" on page 2152


## Pointers

A pointer stores the memory address of objects, such as variables or function block instances, at runtime.

## Syntax of the pointer declaration:

<pointer name>: POINTER TO <data type | data unit type | function block>;

## Example

```
FUNCTION_BLOCK FB_Point
VAR
    piNumber: POINTER TO INT;
    iNumber1: INT := 5;
    iNumber2: INT;
END_VAR
piNumber := ADR(iNumber1); // piNumber is assigned to address of
iNumber1
iNumber2 := piNumber^; // value 5 of iNumber1 is assigned to
variable iNumber2 by dereferencing of pointer piNumber
```

Dereferencing a pointer means obtaining the value to which the pointer points. A pointer is dereferenced by appending the content operator ${ }^{\wedge}$ to the pointer identifier (for example, piNumber^ in the example above). To assign the address of an object to a pointer, the address operator ADR is applied to the object: ADR (iNumber1).
In online mode, you can click "Edit $\rightarrow$ Browse $\rightarrow$ Go to Reference" to jump from a pointer to the declaration location of the referenced variable.

## NOTICE!

When a pointer points to an I/O input, write access applies. This leads to the compiler warning "'<pointer name >' is not a valid assignment target" when the code is generated. Example: pwInput :=ADR(wInput);

If you require a construct of this kind, you have to first copy the input value (wInput) to a variable with write access.

Index access to pointers

CODESYS permits the index access [] to variables of type POINTER TO, as well as to the data types STRING or WSTRING.
The data, which the pointer points to, can also be accessed by appending the bracket operator [] to the pointer identifier(for example, piData [i]). The base data type of the pointer determines the data type and the size of the indexed component. In this case, the index access to the pointer is done arithmetically by adding the index dependent offset i * SIZEOF ( <base type> ) to the address of the pointer. The pointer is dereferenced implicitly at the same time.

Calculation: piData[i] := (piData + i * SIZEOF(INT))^;
This is not: piData[i] ! = (piData + i)^.

## Index access STRING

When you use the index access with a variable of the type STRING, you get the character at the offset of the index expression. The result is of type BYTE. For example, sData [i] returns the i-th character of the character string sData as SINT (ASCII).

## Index access WSTRING

When you use the index access with a variable of the type WSTRING, you get the character at the offset of the index expression. The result is of type WORD. For example, wsData [i] returns the i-th character of the character string as INT (Unicode).

[^34]Using references provides the advantage of guaranteeing type safety. That is not the case with pointers.

The memory access of pointers can be checked at runtime by the implicit monitoring function CheckPointer.

## See also

- ${ }^{\wedge}$ Chapter 6.4.1.21.3.3.38 "Command 'Go To Reference'" on page 2576
- «\& Chapter 6.4.1.21.4.11.5 "Dialog 'Properties' - 'Build"' on page 2755
- ${ }^{\bullet} \otimes$ Chapter 6.4.1.20.3.33 "Operator 'Content Operator'" on page 2151
- $\Leftrightarrow$ Chapter 6.4.1.20.3.32 "Operator 'ADR'" on page 2150
- $\Leftrightarrow$ Chapter 6.4.1.21.2.22.11 "POU 'CheckPointer'" on page 2513


## Reference

A reference implicitly refers to another object. When accessed, the reference is implicitly dereferenced, and therefore does not need a special content operator ${ }^{\wedge}$ such as a pointer.

```
Syntax <identifier> : REFERENCE TO <data type> ;
<data type>: base type of the reference
```


## Example

```
PROGRAM PLC PRG
VAR
    rspeA : REFERENCE TO DUT_SPECIAL;
    pspeA : POINTER TO DUT_SP\overline{PCIAL;}
    speB : DUT_SPECIAL;
END_VAR
rspeA REF= speB; // Reference rspeA is alias for speB. The code
corresponds to pspeA := ADR(speB);
rspeA := speD; // The code corresponds to pspeA^ := speB;
```

The readability of a program is made difficult when the same memory cell is accessed simultaneously by means of an identifier and its alias (for example, speB and rspeA).

## NOTICE!

With compiler version $>=\mathrm{V} 3.3 .0 .0$, references are initialized (at 0 ).

## NOTICE!

If a reference refers to a device input, then the access (for example, rInput REF= Input; ) is applies as write access. This leads to a compiler warning when the code is generated: ". . . invalid assignment target".
If you require a construct of this kind, you have to first copy the input value (rInput) to a variable with write access.

## Invalid declara-

tions ariTest : ARRAY[0..9] OF REFERENCE TO INT;
priTest : POINTER TO REFERENCE TO INT;
rriTest : REFERENCE TO REFERENCE TO INT;
rbitTest : REFERENCE TO BIT;
A reference type must not be used as the base type of an array, pointer, or reference. Furthermore, a reference must not refer to a bit variable. These kinds of constructs generate compiler errors.

Comparison of reference and pointer

A reference has the following advantages over a pointer:

- Easier to use:

A reference can access the contents of the referenced object directly and without dereferencing.

- Finer and simpler syntax when passing values:

Call of a function block which passes a reference without an address operator instead of a pointer
Example: fbDoIt(riInput:=iValue);
Instead of: fbDoIt_1 (piInput:=ADR(iValue));

- Type safety:

When assigning two references, the compiler checks whether their base types match. This is not checked in the case of pointers.

Testing the validity of a reference
Syntax

You can use the operator $\qquad$ ISVALIDREF to check whether or not a reference points to a valid value (meaning a value not equal to 0 ).
<Boolean variable name> := __ISVALIDREF( <reference name> );
<reference name>: Identifier declared with REFERENCE TO
The Boolean variable is TRUE when the reference points to a valid value. Otherwise it is FALSE.

## Example

```
PROGRAM PLC_PRG
VAR
iAlfa : INT;
riBravo : REFERENCE TO INT;
riCharlie : REFERENCE TO INT;
bIsRef Bravo : BOOL := FALSE;
bIsRef_Charlie : BOOL := FALSE;
END_VA}\overline{R
iAlfa := iAlfa + 1;
riBravo REF= iAlfa;
riCharlie REF= 0;
bIsRef Bravo := ISVALIDREF(riBravo); (* becomes TRUE,
because riBravo references to iAlfa, which is non-zero
*)
bIsRef_Charlie := __ISVALIDREF(riCharlie); (* becomes FALSE,
because\overline{e}riCharlie is set to 0 *)
```

In compiler version 3.5.7.40 and later, the implicit monitoring function "CheckPointer" acts on variables of type REFERENCE TO in the same way as for pointer variables.

See also

- \& Chapter 6.4.1.20.1.4.4.6 "Assignment Operator 'REF="' on page 2054
-     * Chapter 6.4.1.21.2.22.11 "POU 'CheckPointer'" on page 2513


## Data Type 'ARRAY'

An array is a collection of data elements of the same data type. CODESYS supports one- and multi-dimensional arrays of fixed or variable length.

As of Automation Builder 2.6.0 array initializations within type definitions are no longer allowed.

Array of fixed You can define arrays in the declaration part of a POU or in global variable lists. length
Syntax of the declaration of a one-dimensional array

```
<variable name> : ARRAY[ <dimension> ] OF <data type> ( :=
<initialization> )? ;
<dimension> : <lower index bound>..<upper index bound>
<data type> : elementary data types | user defined data types |
function block types
// (...)? : Optional
```

Syntax of the declaration of a multi-dimensional array
<variable name> : ARRAY[ <1st dimension> ( , <next dimension> ) + ]
OF <data type> ( := <initialization> )? ;
<1st dimension> : <1st lower index bound>..<1st upper index bound>
<next dimension> : <next lower index bound>..<next upper index bound>
<data type> : elementary data types | user defined data types |
function block types
// (...)+ : One or more further dimensions
// (...)? : Optional

The index limits are integers; maximum of the data type DINT.

## Syntax for data access

```
<variable name>[ <index of 1st dimension> ( , <index of next
dimension> )* ]
// (...)* : 0, one or more further dimensions
```

Note the capability of using the implicit monitoring function CheckBounds () to monitor the maintenance of the index limits at runtime.

```
Example
One-dimen-
sional array of VAR
10 integer ele-
ments
Lower index limit: 0
    Upper index limit: }
Initialization
    aiCounter : ARRAY[0..9] OF INT := [0, 10, 20, 30, 40, 50, 60, 70,
    80, 90];
Data access
iLocalVariable := aiCounter[2];
```

The value 20 is assigned to the local variable.

```
Example
2-dimensional
array
VAR
        aiCardGame : ARRAY[1..2, 3..4] OF INT;
        END_VAR
    1st dimension: }1\mathrm{ to 2
    2nd dimension: 3 to 4
Initialization
aiCardGame : ARRAY[1..2, 3..4] OF INT := [2(10),2(20)]; // Short
notation for [10, 10, 20, 20]
Data access
```

```
iLocal_1 := aiCardGame[1, 3]; // Assignment of 10
```

iLocal_1 := aiCardGame[1, 3]; // Assignment of 10
iLocal_2 := aiCardGame[2, 4]; // Assignment of 20

```
iLocal_2 := aiCardGame[2, 4]; // Assignment of 20
```

```
Example
3-dimensional
array
    VAR
        aiCardGame : ARRAY[1..2, 3..4, 5..6] OF INT;
        END_VAR
    1st dimension: }1\mathrm{ to 2
    2nd dimension: }3\mathrm{ to 4
    3rd dimension: 5 to 6
    2 * 2 * 2 = 8 array elements
Initialization
aiCardGame : ARRAY[1..2, 3..4, 5..6] OF INT := [10, 20, 30, 40, 50,
60, 70, 80];
```


## Data access

```
iLocal_1 \(:=\) aiCardGame[1, 3, 5]; // Assignment of 10
iLocal_2 \(:=\) aiCardGame[2, 3, 5]; // Assignment of 20
iLocal_3 \(:=\) aiCardGame[1, 4, 5]; // Assignment of 30
iLocal_4 \(:=\) aiCardGame[2, 4, 5]; // Assignment of 40
iLocal_5 \(:=\) aiCardGame[1, 3, 6]; // Assignment of 50
iLocal_6 \(:=\) aiCardGame[2, 3, 6]; // Assignment of 60
iLocal_7 \(:=\) aiCardGame[1, 4, 6]; // Assignment of 70
iLocal_8 \(:=\) aiCardGame[2, 4, 6]; // Assignment of 80
```


## Initialization

```
aiCardGame : ARRAY[1..2, 3..4, 5..6] OF INT := [2(10), 2(20),
```

aiCardGame : ARRAY[1..2, 3..4, 5..6] OF INT := [2(10), 2(20),
2(30), 2(40)]; // Short notation for [10, 10, 20, 20, 30, 30, 40,
2(30), 2(40)]; // Short notation for [10, 10, 20, 20, 30, 30, 40,
40]
Data access
iLocal_1 $:=$ aiCardGame[1, 3, 5]; // Assignment of 10
iLocal_2 $:=$ aiCardGame[2, 3, 5]; // Assignment of 10
iLocal_3 $:=$ aiCardGame[1, 4, 5]; // Assignment of 20
iLocal_4 $:=$ aiCardGame[2, 4, 5]; // Assignment of 20
iLocal_5 $:=$ aiCardGame[1, 3, 6]; // Assignment of 30
iLocal_6 $:=$ aiCardGame[2, 3, 6]; // Assignment of 30
iLocal_7 $:=$ aiCardGame[1, 4, 6]; // Assignment of 40
iLocal_8 $:=$ aiCardGame[2, 4, 6]; // Assignment of 40

```

\section*{Example}

\section*{3-dimensional}
arrays of a
```

TYPE DATA_A

```
user-defined STRUCT
structure iA_1 : INT;
                                iA_2 : INT;
                                dwĀ_3 : DWORD;
END_STRUCT
END_TYPE
PROGRAM PLC_PRG
VAR
END_VAR

The array aData_A consists of a total of 3 * 3 * \(10=90\) array elements of data type DATA_A.
Initialize parti-
ally
```

aData_A : ARRAY[1..3, 1..3, 1..10] OF DATA_A := [(iA_1 := 1,
iA_2 \= 10, dwA 3 := 16\#00FF),(iA_1 := 2, \overline{iA 2 := 20, dwA_3 :=}
16\#FF00),(iA_1 := 3, iA_2 := 30, dwA_3 := 16\#FFFF)];

```

In the example, only the first 3 elements are initialized explicitly. Elements to which no initialization value is assigned explicitly are initialized internally with the default value of the basic data type. This initializes the structure components at 0 starting with the element aData_A [2, 1, 1].
Data access
```

iLocal_1 := aData_A[1,1,1].iA_1; // Assignment of 1
dwLocal_2 := aData_A[3,1,1].dwA_3; // Assignment of 16\#FFFF

```

\section*{Example}

\section*{Array of a}
function block FUNCTION BLOCK FBObject_A
VAR iCounter : INT;
END_VAR
```

PROGRAM PLC_PRG
VAR
aObject_A : ARRAY[1..4] OF FBObject_A;
END_VAR

```

The array aObject_A consists of 4 elements. Each element instantiates a FBObject_A function block.

\section*{Function call}
```

aObject_A[2]();

```
```

Example
Implementa-
tion of
with method
FB_Init

```
FUNCTION_BLOCK FB_Something
```

FUNCTION_BLOCK FB_Something
VAR
VAR
_nId : INT;
_nId : INT;
_lrIn : LREAL;
_lrIn : LREAL;
END_VAR
END_VAR
METHOD FB_Init : BOOL
METHOD FB_Init : BOOL
VAR_INPUT
VAR_INPUT
bInitRetains : BOOL;
bInitRetains : BOOL;
bInCopyCode : BOOL;
bInCopyCode : BOOL;
nId : INT;
nId : INT;
lrIn : LREAL;
lrIn : LREAL;
END_VAR
END_VAR
nId := nId;
nId := nId;
_lrIn := lrIN;
_lrIn := lrIN;
The function block FB_Something has a method FB_Init that requires 2 parameters.

```
```

Instantiation of

```
Instantiation of
the array with
the array with
initialization
```

initialization

```
```

PROGRAM PLC_PRG

```
PROGRAM PLC_PRG
VAR
VAR
            fb_Something_1 : FB_Something(nId := 11, lrIn := 33.44);
            fb_Something_1 : FB_Something(nId := 11, lrIn := 33.44);
            a_Something : ARRAY[0..1, 0..1] OF FB_Something[(nId := 12,
            a_Something : ARRAY[0..1, 0..1] OF FB_Something[(nId := 12,
lrIn := 11.22), (nId := 13, lrIn := 22.33), (nId := 14, lrIn :=
lrIn := 11.22), (nId := 13, lrIn := 22.33), (nId := 14, lrIn :=
33.55),(nId := 15, lrIn := 11.22)];
33.55),(nId := 15, lrIn := 11.22)];
END_VAR
```

END_VAR

```

Array of arrays The declaration of an "array of arrays" is an alternative syntax for multidimensional arrays. A collection of elements is nested instead of dimensioning the elements. The nesting depth is unlimited.

\section*{Syntax for declaration \\ Syntax for data access}
```

<variable name> : ARRAY[<first>] ( OF ARRAY[<next>] )+ OF <data
type> ( := <initialization> )? ;
<first> : <first lower index bound>..<first upper index bound>
<next> : <lower index bound>..<upper index bound> // one or more
arrays
<data type> : elementary data types | user defined data types |
function block types
// (...)+ : One or more further arrays
// (...)? : Optional

```
<variable name>[<index of first array>] ( [<index of next array>] ) + ; // (...)* : 0, one or more further arrays

\section*{Example}
```

PROGRAM PLC_PRG
VAR
aiPoints : ARRAY[1..2,1..3] OF INT := [1,2,3,4,5,6];
ai2Boxes : ARRAY[1..2] OF ARRAY[1..3] OF INT := [ [1, 2, 3],
[ 4, 5, 6]];
ai3Boxes : ARRAY[1..2] OF ARRAY[1..3] OF ARRAY[1..4] OF INT :=
[ [ [1, 2, 3, 4], [5, 6, 7, 8 ], [9, 10, 11, 12] ], [ [13, 14, 15,
16], [ 17, 18, 19, 20], [21, 22, 23, 24] ] ];
ai4Boxes : ARRAY[1..2] OF ARRAY[1..3] OF ARRAY[1..4] OF
ARRAY[1..5] OF INT;
END_VAR
aiPoints[1, 2] := 1200;
ai2Boxes[1][2] := 1200;

```

The variables aiPoints and ai2Boxes collect the same data elements，however the syntax for the declaration differs from that of the data access．
\begin{tabular}{|c|c|c|}
\hline －爭 aiPoints & ARRAY［ \(1 . .2,1.3]\) OF INT & \\
\hline ，\({ }^{\text {人 }}\) aiPoints \([1,1]\) & INT & 1 \\
\hline \％aiPoints［1，2］ & INT & 1200 \\
\hline 者 aiPoints［1，3］ & INT & 3 \\
\hline \％aiPoints \([2,1]\) & INT & 4 \\
\hline ，aiPoints［2，2］ & INT & 5 \\
\hline （1）aiPoints \([2,3]\) & INT & 6 \\
\hline －\(\hat{\beta}^{\text {a }}\) ai2Boxes & ARRAY［1．2］OF ARRAY［ 1.13\(]\) OF INT & \\
\hline －\(\hat{\text {－}}\) ai2Boxes［1］ & ARRAY［1．3］OF INT & \\
\hline ＊ai2Boxes［1］［1］ & INT & 1 \\
\hline \％ai2Boxes［1］［2］ & INT & 1200 \\
\hline 令 ai2Boxes［1］［3］ & INT & 3 \\
\hline －\({ }^{\text {a }}\) ai2Boxes［2］ & ARRAY［1，3］OF INT & \\
\hline \％ai2Boxes［2］［1］ & INT & 4 \\
\hline \％ai2Boxes［2］［2］ & INT & 5 \\
\hline \％ai2Boxes［2］［3］ & INT & 6 \\
\hline
\end{tabular}

Array of variable In function blocks，functions，or methods，you can declare arrays of variable length in the length VAR＿IN＿OUT declaration section．

The LOWER＿BOUND and UPPER＿BOUND operators are provided for determining the index limits of the actual used array at runtime．


Only statically declared arrays（not arrays generated by means of the operator NEW）may be passed to an array with variable length．

Syntax of the declaration of a one-dimensional array of variable length
```

<variable name> : ARRAY[*] OF <data type> ( := <initialization> )? ;
<data type> : elementary data types | user defined data types |
function block types
// (...)? : Optional

```

Syntax of the declaration of a multi-dimensional array of variable length
```

<variable name> : ARRAY[* ( , * )+ ] OF <data type> ( :=

```
<initialization> )? ;
<data type> : elementary data types | user defined data types |
function block types
// (...)+ : One or more further dimensions
// (...)? : Optional

Syntax of the operators for calculating the
```

LOWER BOUND( <variable name> , <dimension number> )

```
UPPER_BOUND ( <variable name> , <dimension number> )
limit index

\begin{abstract}
Example The SUM function adds the integer values of the array elements and returns the calculated sum as a result. The sum is calculated across all array elements available at runtime. As the actual number of array elements will only be known at runtime, the local variable is declared as a one-dimensional array of variable length.
\end{abstract}
```

FUNCTION SUM: INT;
VAR_IN_OUT
aiData : ARRAY[*] OF INT;
END VAR
VAR
diCounter : DINT;
iResult : INT;
END_VAR
iResult := 0;
FOR diCounter := LOWER_BOUND(aiData, 1) TO
UPPER_BOUND(aiData, 1) DO // Calculates the
length of the current array
iResult := iResult + aiData[diCounter];
END_FOR;
SUM := iResult;

```

See also
- \(\quad{ }^{n} \boldsymbol{y}\) Chapter 6.4.1.9.4.4 "Declaring arrays" on page 1853
- \({ }^{4}\) Chapter 6.4.1.21.2.22.2 "POU 'CheckBounds'" on page 2502

\section*{Data Type '__VECTOR'}

Vector operations are supported natively only on 64-bit processors and offer a performance advantage only on these processors. The data sheet of the controller provides information about the processor used on the controller.

Currently, vector operations on the x86/64-bit platforms with SSE2 and ARM64 with NEON are supported natively. On all other platforms, vector operations are translated into individual statements. For example, vector addition is then executed with multiple single addition operations.

The command set extensions of the processors are SIMD extensions. SIMD (Single Instruction, Multiple Data) describes a computer architecture in which multiple data sets of the same type are processed simultaneously in parallel and therefore faster with one command call. In vector operations, for example, 4 pairs of numbers can then be added at the same time.

\section*{Syntax}
```

<variable name> : __VECTOR[ <vector size> ] OF <element type> ( :=
<initialization> ) ? ;
<vector size> : 1 | 2 | 3 | 4 | 5| 6 | 7| 8
<element type> : REAL | LREAL
// (...)? : Optional

```

A vector data type is an array of floating-point numbers with a maximum of 8 elements. The operators __vc<operator name> are available for this data type. You can use these to implement vector operations without additional function calls.

\section*{Syntax for index}
access <variable name>[ <index> ]
<index> : 0 | 1 | \(2|3| 4|5| 6 \mid 7\)
When indexing a vector variable, you can access a single element of the vector. The index starts at 0 and goes until <vector size>-1.

\section*{Example}
```

PROGRAM PLC_PRG
VAR
vcA : __VECTOR[3] OF REAL;
END_VAR
vcA[0] := 1.1;
vcA[1] := 2.2;
vcA[2] := 3.3;

```

\section*{Determining the optimal vector size}

Use the optimal vector size depending on your target system as the vector size in order to program the most efficient code possible.

For target systems whose computer architecture is generally suitable for vector processing, we do not recommend using vectors of arbitrary size. There is an optimal vector size depending on the type of data processing of the processor. Vectors that are declared with this array size are processed as quickly as possible. Vectors that are declared as a larger array do not have an advantage in speed. Vectors that are declared as smaller arrays do not take full advantage of the processor's capabilities.
You can query the optimal size at runtime. You can find the information in the constants Constants.vcOptimalREAL (for vectors with REAL elements) and Constants.vcOptimalLREAL (for vectors with LREAL elements). The constants have the LREAL data type. If a constant returns the value 1 as the optimal value, then this means that accelerated vector processing is not available for the target system.

\section*{Example}
```

PROGRAM PLC_PRG
VAR
iOVS_REAL : INT; // Optimal vector size for REAL elements
iOVS_LREAL : INT; // Optimal vector size for LREAL elements
END_VAR
iOVS_REAL := Constants.vcOptimalREAL;
iOVS_LREAL := Constants.vcOptimalLREAL;

```

An application that is loaded on the CODESYS Control Win V3 x64 target system returns the following values at runtime:

\section*{Device.Application.PLC_PRG}


Operator _VCADD

Syntax

The operator calculates the sum of two vectors.
```

<vector variable> := <1st vector operand> _VCADD <2nd vector
operand>;

```

Example of addition
```

FUNCTION_BLOCK FB_ADD
VAR
vcA : __VECTOR[3] OF REAL := __VCSET_REAL (3, 3, 3);
vcB : __VECTOR[3] OF REAL := __VCSET_REAL (1, 2, 3);
vcResult : __VECTOR[3] OF REAL;
END_VAR
vcResult := vcA __VCADD vcB;

```

Operator VCSUB

\section*{Syntax}

\section*{Example of subtraction}
```

FUNCTION_BLOCK FB_SUB
VAR
vCA : VECTOR[3] OF REAL := VCSET REAL (3, 3, 3);
vcB : __VECTOR[3] OF REAL := _VCSET_REAL(1, 2, 3);
vcResult0 : __VECTOR[3] OF REAL;
vcResult1 : __VECTOR[3] OF REAL;
END_VAR
vcResult0 := vcA VCSUB vcB;
vcResult1 := vcB __VCSUB vcA;

```

Operator VCMUL
Syntax

The operator calculates the product of two vectors or a scalar (floating-point number) and a vector.
```

<vector variable> := <1st vector operand> ___VCMUL <2nd vector
operand> | <scalar operand> __VCMUL <vector operand> | <vector
operand> VCMUL <scalar operand> ;

```

\section*{Example of multiplication}
```

FUNCTION_BLOCK FB_MUL
VAR
rScalar : REAL := 1.1;
vcA : __VECTOR[3] OF REAL;
vcB : VECTOR[3] OF REAL;
vcResult0 : _VECTOR[3] OF REAL;
vcResult1 : __VECTOR[3] OF REAL;
vcResult2 : __VECTOR[3] OF REAL;
END_VAR
vcResult0 := vcA VCMUL vcB;
vcResult1 := rScalar vCMUL vcB;
vcResult2 := vcA __VCMUL 3.3;

```

\section*{Operator} VCDIV

Syntax
```

<vector variable> := <vector dividend> __VCDIV <vector divisor> | <
vector dividend> VCMUL <scalar divisor> ;

```

Example of division
```

FUNCTION_BLOCK FB_DIV
VAR
iScalar : INT := 3;
rScalar : REAL := 1.5;
vCA : __VECTOR[3] OF REAL := __VCSET_REAL (3, 3, 3);
vcB : __VECTOR[3] OF REAL := _VCSET_REAL (1, 2, 3);
vcResult0 : __VECTOR[3] OF REAL;
vcResult1 : __VECTOR[3] OF REAL;
vcResult2 : ___VECTOR[3] OF REAL;
END_VAR
vcResult0 := vcA VCDIV vcB;
// ERROR CODE vcResult1 := rScalar _VCDIV vcB;
// ERROR CODE vcResult1 := iScalar __VCDIV vcB;
// ERROR CODE vcResult1 := 3.3 VC\overline{DIV vcB;}
vcResult2 := vcA __VCDIV 1.5;
vcResult2 := vcA __VCDIV iScalar;
vcResult2 := vcA ___VCDIV rScalar;

```
Operator The operator calculates the dot product (scalar product) of two vectors.

\section*{_VCDOT}

Syntax
```

<scalar variable> := <1st vector operand>

```
\(\qquad\)
``` VCDOT <2nd vector
operand> ;
```

Example of a dot product

```
FUNCTION_BLOCK FB_DOT
VAR
    rResult : REAL;
    vCA : VECTOR[3] OF REAL := VCSET REAL (3, 3, 3);
    vCB : __VECTOR[3] OF REAL := __VCSET_REAL (1, 2, 3);
END_VAR
rResult := vcA __VCDOT vcB; // = 18
```

Operator VCSQRT

Syntax

```
<vector variable> := __VCSQRT( <vector operand> );
```


## Example of a square root

```
FUNCTION_BLOCK FB_SQRT
VAR
    vCA : __VECTOR[3] OF REAL := __VCSET_REAL(4, 9, 16);
    vcResult0 : __VECTOR[3] OF REAL;
END_VAR
vcResult0 := __VCSQRT(vcA);
```

| Operator $\qquad$ VCMAX | The operator calculates the maximum vector of two vectors. The maximum is determined element by element. |
| :---: | :---: |
| Syntax | ```<vector variable> := __VCMAX( <1st vector operand>, <2nd vector operand>);``` |
| Example of a maximum vector |  |
|  | ```FUNCTION_BLOCK FB_MAX VAR vCA : ___VECTOR[3] OF REAL := _VCSET_REAL (3, 3, 3); VCB : __VECTOR[3] OF REAL := _VCSET_REAL (1, 2, 6); vcResult0 : __VECTOR[3] OF RE\overline{AL}; END VAR``` |
|  | vcResulto := V_VCMAX (vcA, vcB) ; |

Operator VCMIN

## Syntax

The operator calculates the minimum vector of two vectors. The minimum is determined element by element.

```
<vector variable> := __VCMIN( <1st vector operand>, <2nd vector
operand>);
```


## Example of a minimum <br> vector

```
FUNCTION_BLOCK FB_MIN
VAR
    vCA : __VECTOR[3] OF REAL := __VCSET_REAL (3, 3, 3);
    vcB : __VECTOR[3] OF REAL := __VCSET_REAL (1, 2, 6);
    vcResult0 : __VECTOR[3] OF REAL;
END_VAR
vcResult0 := __VCMIN(vcA, vcB);
```

Operator
VCSET_REAL
The operator sets all elements of a vector in a statement. The elements have the REAL data type.

## Syntax

```
<vector variable> := __ VCSET_REAL( <first literal>, ( < next
literal> )+ ) ;
( ... ) + // number of elements have to match
```


## Example

```
FUNCTION_BLOCK FB_SET
VAR
    vCA : ___VECTOR[3] OF REAL := _VCSET_REAL (3, 3, 3);
    VCB : ___VECTOR[3] OF REAL :=___VCSET_REAL (1, 2, 3);
END_VAR
VCA := __VCSET_REAL (4, 4, 4);
VCB := __VCSET_REAL(1.1, 2.2, 3.3);
```

| Operator _VCSET_LREA | The operator sets all elements of a vector at once in a statement. The elements data type. |
| :---: | :---: |
| L | They can be used wherever variables are valid, such as in assignments in implem as parameters in function calls. |
| Syntax | ```<vector variable> := __VCSET_LREAL( <first literal>, ( < next literal> )+ ) ; ( ... )+ // number of elements have to match``` |
| Example |  |
|  | ```FUNCTION_BLOCK FB_SET VAR vclA : __VECTOR[3] OF LREAL := _VCSET_LREAL (3, 3, 3); vclB : ___VECTOR[3] OF LREAL := ___VCSET_LREAL(1, 2, 3); END_VAR vclA := __VCSET_LREAL(-1.7976931348623158E+308, 0.0, 1.7976931348623158E+308); vclB := __VCSET_LREAL(-1.7976931348623158E+308, 0.0, 1.797693\overline{1348623158E+308);}``` |

Operator
_VCLOAD_RE
$\overline{\text { AL }}$

## Syntax

## Example of

 vectorizationOperator VCLOAD_LRE

The operator interprets any arbitrary memory area as a vector. This is helpful for connecting vector variables to existing code. The operator requires 2 parameters. The first parameter indicates the number of vector elements. The second parameter is a pointer to the REAL data.

```
<vector variable> := __VCLOAD_REAL( <vector size>, <pointer to data
```

<vector variable> := __VCLOAD_REAL( <vector size>, <pointer to data
of type REAL> ) ;
of type REAL> ) ;
<vector size> : 2 | 3 | 4 | 5| 6 | 7| 8

```
<vector size> : 2 | 3 | 4 | 5| 6 | 7| 8
```

vectorization

```
FUNCTION_BLOCK FB_LOAD
```

FUNCTION_BLOCK FB_LOAD
VAR_INPUT
VAR_INPUT
END VAR
END VAR
VAR_OUTPUT
VAR_OUTPUT
END_VAR
END_VAR
VAR
VAR
rData0 : REAL := 1.234;
rData0 : REAL := 1.234;
rData1: REAL := 5.678;
rData1: REAL := 5.678;
rData2 : REAL := 9.123;
rData2 : REAL := 9.123;
pData: POINTER TO REAL := ADR(rData0);
pData: POINTER TO REAL := ADR(rData0);
vCA : __VECTOR[3] OF REAL := __VCSET_REAL (3, 3, 3);
vCA : __VECTOR[3] OF REAL := __VCSET_REAL (3, 3, 3);
END_VAR
END_VAR
vCA := __VCLOAD_REAL(3, pData);

```
vCA := __VCLOAD_REAL(3, pData);
```

The operator interprets any arbitrary memory area as a vector. This is helpful for connecting

## Syntax

```
<vector variable> := _ vCLOAD REAL( <vector size>, <pointer to data
of type LREAL> );
<number of vector elements> : 1 | 2 | 3 | 4 | 5| 6 | 7| 8
```


## Example of vectorization

```
FUNCTION_BLOCK FB_LOAD
VAR_INPUT
END VAR
VAR_OUTPUT
END VAR
VAR
    lrData0 : LREAL := -1.7976931348623158E+308;
    lrData1: LREAL := 1.6E+308;
    lrData2 : LREAL := 1.7E+308;
    lrData3 : LREAL := -1.6E+308;
    plData: POINTER TO LREAL := ADR(lrDataO);
    vclA : __VECTOR[4] OF LREAL := __VCSET_LREAL(4, 4, 4, 4);
END VAR
vcl\overline{A}:= __VCLOAD_LREAL(4, plData);
```

Operator The operator saves/copies the contents of the vector to the specified memory address. The VCSTORE number and the types of elements are automatically applied from the vector variables.

## Syntax

```
__VCSTORE( <pointer to data>, <vector variable> );
```


## Example of storage

```
FUNCTION_BLOCK FB_STORE
VAR INPUT
END_VAR
VAR_OUTPUT
END VAR
VAR
    rData0 : REAL := 3;
    rDatal: REAL := 3;
    rData2 : REAL := 3;
    pData: POINTER TO REAL := ADR(rDataO);
    lrData0 : LREAL := 4;
    lrDatal: LREAL := 4;
    lrData2 : LREAL := 4;
    lrData3 : LREAL := 4;
    plData: POINTER TO LREAL := ADR(lrDataO);
    vcA : __VECTOR[3] OF REAL := __VCSET_REAL( 1.234, 5.678, 9.123);
    vclA : VECTOR[4] OF LREAL :=
    VCSET_LREAL}(-1.7976931348623158E+308, 1.6E+308, 1.7E+308
-1.6E+308);
END_VAR
__VCSTORE (pData, vcA);
__VCSTORE (plData, vclA);
```

See also

- $y^{*}$ Chapter 6.4.1.9.4.4 "Declaring arrays" on page 1853
- ${ }^{4}$ Chapter 6.4.1.21.2.22.2 "POU 'CheckBounds"' on page 2502


## Structure

A structure is a user-defined data type, which combines multiple variables of any data type into a logical unit. The variables declared within a structure are called members.

You make the type declaration of a structure in a "DUT" object which you create in the "Project $\rightarrow$ Add Object $\rightarrow$ DUT" menu or in the context menu of an application.

## Syntax

```
TYPE <structure name> :
STRUCT
    ( <variable declaration optional with initialization> ) +
END_STRUCT
END_TYPE
```

<structure name> is an identifier which is valid in the entire project so that you can use it like a standard data type. Moreover, you can declare any number of variables (at least one) which are supplemented optionally by an initialization.
Structures can also be nested. This means that you declare a structure member with an existing structure type. Then the only restriction is that you must not assign any address to the variable (structure member). (The AT declaration is not permitted here.)

## Example

```
Type declara- TYPE S_POLYGONLINE :
tion
STRUCT
    aiStart : ARRAY[1..2] OF INT := [-99, -99];
    aiPoint1 : ARRAY[1..2] OF INT;
    aiPoint2 : ARRAY[1..2] OF INT;
    aiPoint3 : ARRAY[1..2] OF INT;
    aiPoint4 : ARRAY[1..2] OF INT;
    aiEnd : ARRAY[1..2] OF INT := [99, 99];
END_STRUCT
END_TYPE
```

Extension of a type declaration

## Syntax

```
TYPE <structure name> EXTENDS <basis structure> :
STRUCT
    ( <variable declaration optional with initialization> ) +
END_STRUCT
END_TYPE
```


## Example

```
Type declara- TYPE S_PENTAGON EXTENDS S_POLYGONLINE :
tion STRUCT
    aiPoint5 : ARRAY[1..2] OF INT;
    END STRUCT
    END_TYPE
```

Declaration and initialization of structure variables

## Example

```
PROGRAM progLine
VAR
    sPolygon : S_POLYGONLINE := (aiStart:=[1,1], aiPoint1:=[5,2],
aiPoint2:=[7,3], aiPoint3:=[8,5], aiPoint4:=[5,7], aiEnd:=[1,1]);
    sPentagon : S_PENTAGON := (aiStart:=[0,0], aiPoint1:=[1,1],
aiPoint2:=[2,2], aiPoint3:=[3,3], aiPoint4:=[4,4], aiPoint5:=[5,5],
aiEnd:=[0,0]);
END_VAR
```

You must not permitted to use initializations with variables. For an example of initializing an array of a structure, see the help page for the data type ARRAY.

Access to a You access structure members with the following syntax: structure member

```
<variable name> . <component name>
```


## Example

```
    PROGRAM prog_Polygon
    VAR
    sPolygon : S_POLYGONLINE := (aiStart:=[1,1], aiPoint1:=[5,2],
    aiPoint2:=[7,3], aiPoint3:=[8,5], aiPoint4:=[5,7], aiEnd:=[1,1]);
        iPoint: INT;
    END_VAR
    // Assigns 5 to aiPoint
    iPoint := sPolygon.aiPoint1[1];
Result:iPoint = 5
```

Symbolic bit access in structure variables

You can declare a structure with variables of data type BIT to combine individual bits into a logical unit. Then you can symbolically address individual bits by a name (instead of by a bit index).
Syntax declaration
TYPE <structure name> :
STRUCT
( <bit name> : BIT; ) +
END_STRUCT
END_TYPE
Syntax of bit <structure name> . <bit name>
access

```
Example
Type declara-
tion TYPE S_CONTROL :
    STRUCT
        bitOperationEnabled : BIT;
        bitSwitchOnActive : BIT;
        bitEnableOperation : BIT;
        bitError : BIT;
        bitVoltageEnabled : BIT;
        bitQuickStop : BIT;
        bitSwitchOnLocked : BIT;
        bitWarning : BIT;
END_STRUCT
END_TYPE
Bit access
```

```
FUNCTION_BLOCK FB_Controller
```

FUNCTION_BLOCK FB_Controller
VAR_INPUT
VAR_INPUT
xStart : BOOL;
xStart : BOOL;
END_VAR
END_VAR
VAR_OUTPUT
VAR_OUTPUT
END VAR
END VAR
VAR
VAR
ControlDriveA : S_CONTROL;
ControlDriveA : S_CONTROL;
END_VAR
END_VAR
IF xStart = TRUE THEN
IF xStart = TRUE THEN
// Symbolic bit access
// Symbolic bit access
ControlDriveA.bitEnableOperation := TRUE;
ControlDriveA.bitEnableOperation := TRUE;
END_IF
END_IF
PROGRAM PLC_PRG
PROGRAM PLC_PRG
VAR
VAR
fbController : FB_Controller;
fbController : FB_Controller;
END_VAR
END_VAR
fbController();
fbController();
fbController.xStart := TRUE;

```
fbController.xStart := TRUE;
```



References and pointers to BIT variables are invalid declarations, as well as array elements with base type BIT.

See also

- ${ }^{4}$ Chapter 6.4.1.20.4.11 "Bit Access in Variables" on page 2229

See also

- ${ }^{4}$ Chapter 6.4.1.20.5.15 "Data Type 'ARRAY"" on page 2247
- ${ }^{\bullet}$ Chapter 6.4.1.20.5.11 "Data Type 'BIT"' on page 2243
- ※ Chapter 6.4.1.21.2.9 "Object 'DUT"' on page 2461


## Enumerations

An enumeration is a user-defined data type composed of a series of comma-separated components (enumeration values) for declaring user-defined variables. Moreover, you can use the enumeration components like constants whose identifier <enumeration name>. <component name> is recognized globally in the project.

You declare an enumeration in a DUT object, which you have already created in the project by clicking "Add Object".

## Declaration

## Syntax

```
( {attribute 'strict'} )? // Pragma optional but recommended
TYPE <enumeration name> :
(
    <first component declaration>,
    ( <component declaration> ,)+
    <last component declaration>
) ( <basic data type> ) ? ( := <default variable initialization> )? ;
END_TYPE
( ... ) ? : Optional
<component declaration> : <component name> ( := <component
initialization> )?
<basic data type> : INT | UINT | SINT | USINT | DINT | UDINT | LINT |
ULINT | BYTE | WORD | DWORD | LWORD
<variable initialization> : <one of the component names>
```

In an enumeration declaration, at least 2 components are usually declared. However, you can declare as many as you want. Every single component can be assigned its own initialization. Enumerations automatically have the basic data type INT, but you can specify another basic data type. Moreover, you can specify a component in the declaration with which an enumeration variable is then initialized.

The pragma \{attribute 'strict'\} causes a strict type test to be performed as described below.

## Example

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE COLOR_BASIC :
(
        yellow,
        green,
        blue,
        black
)
; // Basic data type is INT, default initialization for all
COLOR_BASIC variables is yellow
END_TYPE
```

Enumeration with explicit basic data type

Extensions to the IEC 61131-3 standard
The basic data type for an enumeration declaration is INT by default. However, you can also declare enumerations that are based explicitly on another integer data type.
<basic data type> : INT | UINT | SINT | USINT | DINT | UDINT | LINT |
ULINT | BYTE | WORD | DWORD | LWORD

## Example

```
Enumeration
with basic data TYPE COLOR :
type DWORD (
```

    white := 16#FFFFFF00,
    ```
    white := 16#FFFFFF00,
        yellow := 16#FFFFFFO0,
        yellow := 16#FFFFFFO0,
        green := 16#FF00FF00,
        green := 16#FF00FF00,
        blue := 16#FF0000FF,
        blue := 16#FF0000FF,
        black := 16#88000000
        black := 16#88000000
) DWORD := black; // Basic data type is DWORD, default
) DWORD := black; // Basic data type is DWORD, default
initialization for all COLOR variables is black
initialization for all COLOR variables is black
END_TYPE
```

```
END_TYPE
```

```

\section*{Strict programming rules}

\section*{NOTICE!}

In CODESYS V3.5 SP7 and later, the pragma \{attribute 'strict'\} is added automatically in the first line when declaring an enumeration.

The strict programming rules are activated when adding the pragma \{attribute 'strict'\}.
The following code is considered a compiler error:
- Arithmetic operations with enumeration components

For example, an enumeration variable cannot be used as a counter variable in a FOR loop.
- Assignment of a constant value, which does not correspond to an enumeration value, to an enumeration component
- Assignment of a non-constant variable, which has another data type as the enumeration, to an enumeration component

Arithmetic operations can lead to undeclared values being assigned to enumeration components. A better programming style is to use SWITCH/CASE statements for processing component values.

Declaration and
initialization of
enumeration
variables

\section*{Syntax}
```

<variable name> : <enumeration name> ( := <initialization> )? ;

```

For a declaration of an enumeration variable with user-defined data type <enumeration name>, this can be initialized with an enumeration component.

\section*{Example}
```

PROGRAM PLC_PRG
VAR
colorCar: COLOR;
colorTaxi : COLOR := COLOR.yellow;
END_VAR

```

The variable colorCar is initialized with COLOR.black. That is the default initialization for all enumeration variables of type COLOR and defined this way in the type declaration. The variable colorTaxi has its own initialization.

If no initializations are specified, then the initialization value is 0 .
```

Example
PROGRAM PLC PRG
VAR
cbFlower : COLOR_BASIC;
cbTree: COLOR_BA\overline{SIC := COLOR_BASIC.green;}
END_VAR
The variable cbFlower is initialized with COLOR_BASIC.yellow. That is the default initialization for all enumeration variables of type COLOR_BASIC. Because the enumeration declaration does not specify a component for initialization, the system automatically initializes with the component that has the value 0 . This is usually the first of the enumeration components. However, it can also be another component that is not in the first position but explicitly initialized with 0 .
The variable cbTree has an explicit initialization.

```

If no value is specified for both the type and the variable, then the following rule applies: If an enumeration contains a value for 0 , then this value is the default initialization, and if not, then the first component in the list.

\section*{Example}

Initialization
with the \(\mathbf{0}\) com- TYPE ENUM :
ponent
(
        e1 := 2,
        e2 : = 0,
        e3
    )
;
    END_TYPE
    PROGRAM PLC_PRG
    VAR
        e : ENUM;
    END_VAR
The variable e is initialized with ENUM.e2.
Initialization
with the first TYPE ENUM2 :
component
(
        e1 := 3,
        e2 := 1,
        e3
)
;
END_TYPE
PROGRAM PLC_PRG
VAR
    e2 : ENUM2;
END_VAR

The variable e2 is initialized with ENUM.e1.

Unique access to enumeration components

Extensions to the IEC 61131-3 standard
The enumeration components can also be used as constant variables with the identifier <enumeration name>. <component name>. Enumeration components are recognized globally in the project and access to them is unique. Therefore, a component name can be used in different enumerations.
```

Example
Component
blue PROGRAM PLC_PRG
VAR
cbFlower : COLOR_BASIC;
colorCar : COLOR;
END_VAR
(* unambiguous identifiers although the component names are
identical *)
cbFlower := COLOR_BASIC.blue;
colorCar := COLOR.blue;
(* invalid code *)
cbFlower := blue;
colorCar := blue;

```

\section*{See also}
- \(\quad \stackrel{y}{ }{ }^{\circ}\) Chapter 6.4.1.20.3.73 "Operator - Enumeration namespace" on page 2217

Alias
```

Syntax

```
```

TYPE <DUT name> : <basic type> | <data type> | <function block

```
TYPE <DUT name> : <basic type> | <data type> | <function block
name> ;
name> ;
END_TYPE
```

END_TYPE

```

\section*{Example}
```

FUNCTION_BLOCK FB_Machine
VAR_INPUT
END-VAR
VAR_OUTPUT
END_VAR
VAR }\mp@subsup{}{}{-
iCounter : INT;
END_VAR
iCounter := iCounter + 1;
// Alias for FB_Machine
TYPE A_ROBOT : \overline{FB_Machine;}
END_TYPE
PROGRAM prog_Robot
VAR
fbRobot : A_ROBOT;
END VAR
fbRobot();

```

\section*{Data type 'UNION'}

A UNION is a data structure that usually contains different data types.
In a union, all components have the same offset and therefore the same amount of memory. In the following declaration example of a union, an assignment to name. a will also affect name.b.

\section*{Example}
```

TYPE name:
UNION
a : LREAL;
b : LINT;
END UNION
END_TYPE

```

\section*{Subrange types}

A subrange type is a data type whose value range is a subset of a base type.

\section*{Syntax for the declaration:}
```

<name> : <int type> (<lower limit>..<upper limit>);

```
\begin{tabular}{|l|l|}
\hline <name> & valid IEC identifier \\
\hline \begin{tabular}{l} 
<int \\
type>
\end{tabular} & \begin{tabular}{l} 
data type of the subrange \\
(SINT, USINT, INT, UINT, DINT, UDINT, BYTE, WORD, DWORD, LINT, ULINT, \\
LWORD).
\end{tabular} \\
\hline \begin{tabular}{l} 
<lower \\
limit>
\end{tabular} & \begin{tabular}{l} 
Lower limit of the range: constants that have to be compatible with the basic data \\
type. The lower limit is also included in this range.
\end{tabular} \\
\hline \begin{tabular}{l} 
<upper \\
limit>
\end{tabular} & \begin{tabular}{l} 
Upper limit of the range: constants that have to be compatible with the base data \\
type. The upper limit is also included in this range.
\end{tabular} \\
\hline
\end{tabular}

\section*{Examples:}
```

VAR
i : INT (-4095..4095);
ui : UINT (0..10000);
END_VAR

```

If you assign a value to a subrange type in the declaration or implementation section that is not within this range (example: \(i:=5000\) ), then CODESYS issues an error message.

Please note: In runtime mode, it is possible to monitor the range limits of a subrange type by using the implicit monitoring functions CheckRangeSigned and CheckRangeUnsigned.

See also
- \$ Chapter 6.4.1.21.2.22.8 "POU 'CheckLRangeSigned"" on page 2510
- \(\Leftrightarrow\) Chapter 6.4.1.21.2.22.10 "POU 'CheckLRangeUnsigned"" on page 2512
6.4.1.20.6.1 General ..... 2269
6.4.1.20.6.2 Message Pragmas ..... 2269
6.4.1.20.6.3 Attribute Pragmas ..... 2271
6.4.1.20.6.4 Conditional Pragmas ..... 2318
6.4.1.20.6.5 Region Pragma ..... 2327

\section*{General}

Pragma instructions affect the properties of one or more variables with regard to the compilation or pre-compilation process. Various categories of pragmas are available to you for this.

Pragmas are special statements that influence the behavior of an AC500 V3 project during compilation (build) and precompilation. The basic functionalities and use cases for AC500 V3 PLCs are described in the application note AC500 V3 Using Pragmas.

\section*{Message Pragmas}

Message pragmas serve to force the display of messages in the Message window during the compilation process.
Insertion position: separate or already existing line in the text editor of a POU.

Table 400: 4 types of message pragmas
\begin{tabular}{|c|c|}
\hline Pragma & Message type \\
\hline ```
|{text 
``` & "Text": display of <textstring>. \\
\hline \[
\begin{aligned}
& \text { \{info } \\
& <' \text { textstring' } \\
& \gg\}
\end{aligned}
\] & 6 "Information": display of <textstring>. \\
\hline ```
{warning
<'textstring'
>}
``` & \begin{tabular}{l}
"Warning": display of <textstring>. \\
Unlike the attribute pragma 'obsolete', you define the warning locally for the current position.
\end{tabular} \\
\hline ```
l}\begin{array}{l}{\mathrm{ {error }}\\{<'textstring'}\\{>}}
``` & © "Error ": display of <textstring>. \\
\hline
\end{tabular}

In the CODESYS Message window you can jump with the help of the commands "Next Message" and "Previous Message" from a message of the category "Information", "Warning" and "Error" to the source position of the message. This means you jump to the position where the pragma is added in the source code.

\section*{Example}
```

VAR
var : INT; {info 'TODO: should get another name'}
bvar : BOOL;
arrTest : ARRAY [0..10] OF INT;
i:INT;
END_VAR
arrTest[i] := arrTest[i]+1;
ivar:=ivar+1;
{warning 'This is a warning'}
{text 'Part xy has been compiled completely'}

```

Display in the Message window:
\begin{tabular}{|c|c|c|c|c|}
\hline \multicolumn{4}{|l|}{Messages} & \multirow[t]{2}{*}{\[
\begin{array}{r}
4 \times \\
\hline-7
\end{array}
\]} \\
\hline Build & & & & \\
\hline Description & Project & Object & Position & \\
\hline ------ Build started: Application: Res.App2 ------- & & & & \\
\hline typify code ... & & & & \\
\hline (1) Compile time before typification: 0 ms & & & & \\
\hline (1) Compile time after typification: 15 ms & & & & \\
\hline (3) TODO: should get another name! & TS pragma & NewPOU & Line 3 (Decl) & \\
\hline (9) This is a warning & TS pragma & NewPOU & Line 7 ( Impl) & \\
\hline Part \(x y\) has been compiled completely & TS pragma & NewPOU & Line 10 (Impl) & \\
\hline Compile complete -- 1 errors, 2 warnings & & & & \\
\hline
\end{tabular}

See also
- \(\quad 4\) Chapter 6.4.1.20.6.4 "Conditional Pragmas" on page 2318

\section*{Attribute Pragmas}
6.4.1.20.6.3.1 General ..... 2271
6.4.1.20.6.3.2 User-defined attributes ..... 2272
6.4.1.20.6.3.3 Attribute 'call_after_global_init_slot'. ..... 2273
6.4.1.20.6.3.4 Attribute 'call_after_init'. ..... 2273
6.4.1.20.6.3.5 Attribute 'call_after_online_change_slot'. ..... 2274
6.4.1.20.6.3.6 Attribute 'call_before_global_exit_slot' ..... 2275
6.4.1.20.6.3.7 Attribute 'call_on_type_change' ..... 2275
6.4.1.20.6.3.8 Attribute 'conditionalshow'. ..... 2276
6.4.1.20.6.3.9 Attribute 'conditionalshow_all_locals'. ..... 2277
6.4.1.20.6.3.10 Attribute 'const_replaced', Attribute 'const_non_replaced'. ..... 2278
6.4.1.20.6.3.11 Attribute 'dataflow' ..... 2279
6.4.1.20.6.3.12 Attribute 'displaymode' ..... 2280
6.4.1.20.6.3.13 Attribute 'enable_dynamic_creation' ..... 2281
6.4.1.20.6.3.14 Attribute 'estimated-stack-usage'. ..... 2281
6.4.1.20.6.3.15 Attribute 'ExpandFully' ..... 2284
6.4.1.20.6.3.16 Attribute 'global_init_slot'. ..... 2285
6.4.1.20.6.3.17 Attribute 'hide' ..... 2286
6.4.1.20.6.3.18 Attribute 'hide_all_locals' ..... 2289
6.4.1.20.6.3.19 Attribute 'initialize_on_call'. ..... 2290
6.4.1.20.6.3.20 Attribute 'init_namespace'. ..... 2291
6.4.1.20.6.3.21 Attribute 'init_on_onlchange' ..... 2291
6.4.1.20.6.3.22 Attribute 'instance-path'. ..... 2292
6.4.1.20.6.3.23 Attribute 'io_function_block', 'io_function_block_mapping'. ..... 2293
6.4.1.20.6.3.24 Attribute 'is_connected' ..... 2293
6.4.1.20.6.3.25 Attribute 'linkalways' ..... 2294
6.4.1.20.6.3.26 Attribute 'monitoring'. ..... 2295
6.4.1.20.6.3.27 Attribute 'monitoring_encoding'.... ..... 2297
6.4.1.20.6.3.28 Attribute 'no_assign', Attribute 'no_assign_warning'. ..... 2298
6.4.1.20.6.3.29 Attribute 'no_check' ..... 2298
6.4.1.20.6.3.30 Attribute 'no_copy' ..... 2299
6.4.1.20.6.3.31 Attribute 'no-exit'. ..... 2299
6.4.1.20.6.3.32 Attribute 'noinit'. ..... 2299
6.4.1.20.6.3.33 Attribute 'no_instance_in_retain' ..... 2300
6.4.1.20.6.3.34 Attribute 'no_virtual_actions' ..... 2300
6.4.1.20.6.3.35 Attribute 'pingroup' ..... 2302
6.4.1.20.6.3.36 Attribute 'pin_presentation_order_inputs/outputs' ..... 2303
6.4.1.20.6.3.37 Attribute 'obsolete' ..... 2304
6.4.1.20.6.3.38 Attribute 'pack_mode' ..... 2305
6.4.1.20.6.3.39 Attribute 'ProcessValue'.. ..... 2312
6.4.1.20.6.3.40 Attribute 'qualified_only'. ..... 2312
6.4.1.20.6.3.41 Attribute 'reflection'. ..... 2313
6.4.1.20.6.3.42 Attribute 'subsequent' ..... 2313
6.4.1.20.6.3.43 Attribute 'symbol'. ..... 2314
6.4.1.20.6.3.44 Attribute 'to_string' ..... 2314
6.4.1.20.6.3.45 Attribute 'warning disable', attribute 'warning restore' ..... 2315
6.4.1.20.6.3.46 Effects of Pragmas on Symbols ..... 2315
General

Attribute pragmas affect the compilation and the pre-compilation.

CODESYS supports a series of pre-defined attribute pragmas. In addition you can use userdefined pragmas, which you can query with the help of conditional pragmas before the compilation of the project.
Attributes are defined within the declaration part. Exception: For the objects Action and Transition, which have no own declaration part, you can define the attributes at the beginning of the implementation part

When you define own attributes, please make them unambiguous. Uniqueness can be reached for example by adding a prefix to the attribute name. OEMs can use the vendor prefix for this purpose.

\section*{User-defined attributes}

User-defined attributes are any application-defined or user-defined attributes that you can apply to POUs, actions, data type definitions and variables. You can query a user-defined attribute with the help of conditional pragmas before the compilation of the application.

You can query user-defined attributes with conditional pragmas with the operator hasattribute.

More detailed information and examples can be found in the chapter 'Conditional pragmas'.

\section*{Syntax:}
```

{attribute 'attribute'}

```

\section*{Example for}

POUs and actions
```

Attribute
'vision' for
function "fun1"

```
```

    {attribute 'vision'}
    ```
    {attribute 'vision'}
    FUNCTION fun1 : INT
    FUNCTION fun1 : INT
    VAR_INPUT
    VAR_INPUT
        i : INT;
        i : INT;
    END_VAR
```

    END_VAR
    ```

\section*{Example for var-}
iables
```

Attribute
'DoCount' for
variable ivar:

```
```

PROGRAM PLC_PRG

```
PROGRAM PLC_PRG
VAR
VAR
    {attribute 'DoCount'};
    {attribute 'DoCount'};
    ivar:INT;
    ivar:INT;
    bvar:BOOL;
    bvar:BOOL;
    END_VAR
```

    END_VAR
    ```

\section*{Example for data types}
```

Attribute
'aType' for
data type
DUT_1:

```
```

{attribute 'aType'}

```
{attribute 'aType'}
TYPE DUT_1 :
TYPE DUT_1 :
STRUCT
STRUCT
    a:INT;
    a:INT;
    b:BOOL;
    b:BOOL;
END STRUCT
END STRUCT
END_TYPE
```

END_TYPE

```

See also
- * Chapter 6.4.1.20.6.4 "Conditional Pragmas" on page 2318

\section*{Attribute 'call_after_global_init_slot'}

\section*{NOTICE!}

VAR_INPUT declarations in functions or methods that use the attribute lead to compile errors. Reason: Input variables are unknown in this case at the time of the call, which occurs implicitly during the online change.

The effect of this pragma is that all functions and programs containing this attribute are called after the global initialization. You define the order of calling by means of the attribute value.

\section*{Syntax:}
\{attribute 'call_after_global_init_slot' := '<slot>'\}
<slot>: Integer value that defines the ranking in the order of the calls; the lower the value, the earlier the call takes place. If several function blocks have the same ranking for the attribute, then the order of their calls remains indefinite.
Insert location: First line above the declaration part of functions and programs
If a method possesses the attribute, CODESYS determines all instances of the corresponding function block and calls all instances in the specified slot. In this case you have no influence on the order of the instances among themselves.
See also
- \({ }^{4}\) ) Chapter 6.4.1.20.10 "Methods 'FB_Init', 'FB_Reinit', and 'FB_Exit"' on page 2336

\section*{Attribute 'call_after_init'}

\section*{NOTICE!}

VAR_INPUT declarations in functions or methods that use the attribute lead to compile errors. Reason: Input variables are unknown in this case at the time of the call, which occurs implicitly during the online change.

The effect of this pragma is that a method is called implicitly after the initialization of a function block instance. For reasons of performance you must add the attribute both to the function block and to the method in its own first line above the declaration part.

\section*{Syntax:}
```

{attribute 'call_after_init'}

```

Call: First line above the declaration part of the method and the function block.
CODESYS calls the method after the method FB_init and after the variable values of an initialization expression in the instance declaration have become valid.
This functionality is supported from compiler version 3.4.1.0.

\section*{Example Definition:}
```

{attribute 'call_after_init'}
FUNCTION_BLOCK F\overline{B}
... <function block definition>
{attribute 'call_after_init'}
METHOD FB AfterInit
... <methōd definition>

```

The definition implements, for example, the following declaration in the subsequent code processing:
```

inst : FB := (in1 := 99);

```

Code processing:
```

inst.FB_Init();
inst.in\overline{1}}:=99
inst.FB_AfterInit();

```

This allows a reaction to the user-defined initialization in FB_AfterInit.

\section*{See also}
- ⓨ Chapter 6.4.1.20.10 "Methods 'FB_Init', 'FB_Reinit', and 'FB_Exit"' on page 2336

\section*{Attribute 'call_after_online_change_slot'}

\section*{NOTICE!}

VAR_INPUT declarations in functions or methods that use the attribute lead to compile errors. Reason: Input variables are unknown in this case at the time of the call, which occurs implicitly during the online change.

The effect of this pragma is that all functions and programs containing this attribute are called after an online change. You define the order of calling by means of the attribute <slot>.

\section*{Syntax:}
```

{attribute 'call_after_online_change_slot' := '<slot>'}

```
<slot>: Integer value that defines the ranking in the order of the calls; the lower the value, the earlier the call takes place. If several function blocks have the same ranking for the attribute, then the order of their calls remains indefinite.
Call: First line above the declaration part of functions and programs.
If a method possesses the attribute, then CODESYS determines all instances of the function block concerned. CODESYS calls all instances in the specified slot. In this case you have no influence on the order of the instances among themselves.

\section*{NOTICE!}

Since the application cannot run during the online change, each code executed in this situation can lead to a jitter. Therefore, keep the extent of the executive code as small as possible.

See also
- \({ }^{*} \Rightarrow\) Chapter 6.4.1.21.3.7.6 "Command 'Online Change"" on page 2629

\section*{Attribute 'call_before_global_exit_slot'}

\section*{NOTICE!}

VAR_INPUT declarations in functions or methods that use the attribute lead to compile errors. Reason: Input variables are unknown in this case at the time of the call, which occurs implicitly during the online change.

The effect of this pragma is that all functions and programs containing this attribute in a dedicated first line of their declaration are called before the GlobalExit. The GlobalExit takes place before a new download or a reset. Function blocks provided with an FB_Exit method are affected. The order of calling is defined by means of the attribute value.

\section*{Syntax:}
```

{attribute 'call_before_global_exit_slot' := '<slot>'}

```

Insert location: First line above the declaration part of functions and programs.
<slot>: Integer value that defines the ranking in the order of the calls; the lower the value, the earlier the call takes place. If several function blocks have the same ranking for the attribute, then the order of their calls remains indefinite.

If a method possesses the attribute, then the method is called for all instances of the function block concerned. CODESYS calls all instances in the specified slot. In this case you have no influence on the order of the instances among themselves.
See also
- \(\quad\) Chapter 6.4.1.20.10 "Methods 'FB_Init', 'FB_Reinit', and 'FB_Exit"' on page 2336

\section*{Attribute 'call_on_type_change'}

With this pragma, you can mark a method of a function block \(A\) that should be called when the data type changes for one or more function blocks \(B, C\), etc. that are referenced by \(A\). The referencing can be defined by a pointer variable or a REFERENCE variable.

\section*{Syntax:}
```

{attribute 'call_on_type_change':= '<name of the first referenced
function block>|<name of the second referenced function block>|<name
of the ... referenced function block>'}

```

Insert location: Line above the first line in the method declaration.
```

Example
Function FUNCTION_BLOCK FB_A
blocks with ref- ...
erences VAR
var_pt: POINTER TO FB_B;
var ref: REFERENCE TO-
END_VAR
...
Method for
reaction to a {attribute 'call_on_type_change' := 'FB_B, FB_C'}
type change in
the references
FB_B and
FB_C

```
```

METHOD METH_react_on_type_change : INT

```
METHOD METH_react_on_type_change : INT
VAR INPUT
```

VAR INPUT

```

\section*{Attribute 'conditionalshow'}

The pragma has the effect that the identifiers of an integrated compiled library <library name> . compiled-library, which are decorated with the pragma, are hidden before programming an application. The POUs can be called but the variables are invisible in the CODESYS user interface.

Affected features
- Library management
- Debugging
- Input Assistant
- Function "List components"
- Monitoring
- Symbol configuration

This is useful when you develop libraries. As the library developer, you decorate function blocks or variables with the pragma. As a result, you determine which identifiers are hidden in an application after integration. If you want to show the hidden identifiers later, for example for debugging or further development of the library, you can reactivate its visibility.

\section*{Syntax}
```

{attribute 'conditionalshow' ( := ' <some text> ' )? }

```
<some text>: Optional string literal to control the visibility of the identifiers decorated with this kind of pragma by means of a command-line command and this literal. When the pragma is specified without a literal, the variables in the CODESYS development environment are always hidden, regardless of how CODESYS was started. For more help about this, see the document "Library Development Summary".
Insert location: Top line in the declaration part of a function block, above a variable


Visibility in case When the source code file <library name>. library from an integrated library also exists of existing at the same memory location (repository), the identifiers are visible despite the pragmas. That is regardless of whether or not an attribute value has been specified in the declaration.

\section*{Command-line call to activate visibility}

Syntax
You can also enable the visibility of the hidden variable without a source code file by starting CODESYS with the command-line option conditionalshowsymbols. To enable the visibility, specify the attribute values of the pragma which are separated by commas.
codesys.exe --conditionalshowsymbols=" <some text> ( ,<next text> )* "

\section*{Example}
```

codesys.exe --conditionalshowsymbols="Library_Developer"
codesys.exe --conditionalshowsymbols="Group_A,Group_B"

```

See also
- \({ }^{4}\) Chapter 6.4.1.16 "Using the Command-Line Interface" on page 2028
- 祭 Chapter 6.4.1.20.6.3.17 "Attribute 'hide"' on page 2286
- \({ }^{4}\), Chapter 6.4.1.20.6.3.9 "Attribute 'conditionalshow_all_locals'" on page 2277
- "Library Development Summary", "Visibility Control" Chapter

\section*{Attribute 'conditionalshow_all_locals'}

The pragma has the effect that all local variables of a library POU decorated with the pragma are hidden from application programmers. The POUs of an integrated compiled library <library name> .compiled-library can be called, but the variables are invisible in the CODESYS user interface.

Affects features:
- Library management
- Debugging
- Input Assistant
- Function "List components"
- Monitoring
- Symbol configuration

This is useful when you develop libraries. As the library developer, you decorate function blocks with the pragma. As a result, you determine that their identifiers are hidden in an application after integration. If you want to show these identifiers later, for example for debugging or further development of the library, you can reactivate its visibility.

\section*{Syntax}
\{attribute 'conditionalshow_all_locals' ( := ' <some text> ' )? \}
<some text>: Optional string literal to control the visibility of the identifiers decorated with this kind of pragma by means of a command-line command and this literal. When the pragma is specified without a literal, the variables in the CODESYS development environment are always hidden, regardless of how CODESYS was started. For more help about this, see the document "Library Development Summary".
Insert location: Top line in the declaration part of the function block.

\section*{Example}
variables
```

Hiding all local {attribute 'conditionalshow_all_locals' := 'Library_Developer'}

```
Hiding all local {attribute 'conditionalshow_all_locals' := 'Library_Developer'}
FUNCTION_BLOCK FB_DataManager
VAR
            iLocal : INT;
            iCounter : INT;
        END_VAR
```

For more examples, see the document "Library Development Summary".

Visibility in case When the source code file <library name>. library from an integrated library also exists of existing source code file at the same memory location (repository), the library POU variables are visible despite the pragmas. That is regardless of whether or not an attribute value has been specified in the declaration.

| Command-line | You can also enable the visibility of the hidden variable without a source code file by starting |
| :--- | :--- |
| call to activate | CODESYS with the command-line option conditionalshowsymbols. To enable the visibility |
| visibility | specify the attribute values of the pragma which are separated by commas. |
| Syntax | codesys.exe --conditionalshowsymbols=" <some text> (, <next text> )* |

Example

```
codesys.exe --conditionalshowsymbols="Library_Developer"
codesys.exe --conditionalshowsymbols="Group_A,Group_B"
```

See also

- $\&$ Chapter 6.4.1.20.6.3.18 "Attribute 'hide_all_locals"" on page 2289
- $\Leftrightarrow$ Chapter 6.4.1.20.6.3.8 "Attribute 'conditionalshow'" on page 2276
- "Library Development Summary", chapter "Visibility Control"


## Attribute 'const_replaced', Attribute 'const_non_replaced'

The attribute 'const_replaced' has the effect that the constant is replaced in the code, independently of the setting of the "Replace constants" compiler option. The attribute has an effect for variables of scalar types only, but not for compound types like arrays and structures.

You insert the pragma \{attribute 'const_non_replaced'\} accordingly in order to explicitly deactivate the "Replace constants" compiler option. This has the effect, for example in the symbol configuration, that the constant is available and can be exported despite the compiler option.

The "Replace constants" option in the "Compile Options" category of the "Project Settings" dialog is preset for the entire project, because replacing constants generally leads to faster code and less memory usage.

## Syntax:

```
{attribute 'const_replaced'}
{attribute 'const_non_replaced'}
```

Insert location: Line above the declaration line of the global variables.


See also

- ${ }^{\Perp}$ Chapter 6.4.1.21.4.12.4 "Dialog Box 'Project Settings' - 'Compileoptions'" on page 2769
- $\xi^{*}$ Chapter 6.4.1.10.3 "Symbol Configuration" on page 1941


## Attribute 'dataflow'

With this pragma you control the data flow in the processing of function blocks in the FBD/LD/IL editor. The attribute defines the input or output of a function block to which the continuing connection to the next or previous function block is connected.
You may provide only 1 input and 1 output with the attribute in the declaration of a function block.

## Syntax:

```
{attribute 'dataflow'}
```

Insertion position: line above the line with the declaration of the corresponding variables.
In the case of function blocks without the attribute 'dataflow', CODESYS determines the data flow as follows: first of all the connection is placed between an output and an input of same data type. The highest input or output variable of the function blocks is always taken. If there are no variables of a corresponding data type, CODESYS connects the highest output with the highest input of the neighboring function blocks.

```
Example The connection between FB and the preceding function block is established via the input
        variable i1. The connection between FB and the following function block is established via the
        output variable outRes1.
```

```
FUNCTION BLOCK FB
```

FUNCTION BLOCK FB
VAR_INPUT
VAR_INPUT
r1 : REAL;
r1 : REAL;
{attribute 'dataflow'}
{attribute 'dataflow'}
i1 : INT;
i1 : INT;
i2 : INT;
i2 : INT;
r2 : REAL;
r2 : REAL;
END_VAR
END_VAR
VAR OUTPUT
VAR OUTPUT
{attribute 'dataflow'}
{attribute 'dataflow'}
outRes1 : REAL;
outRes1 : REAL;
out1 : INT;
out1 : INT;
g1 : INT;
g1 : INT;
g2 : REAL;
g2 : REAL;
END_VAR

```
END_VAR
```


## See also

- © Chapter 6.4.1.9.5.2.2 "Programming function block diagrams (FBD)" on page 1862


## Attribute 'displaymode'

With this pragma you define the display mode of an individual variable. This definition overwrites the global setting for the display of the monitoring variable, which takes place via the commands in the menu "Debug $\rightarrow$ Display Mode".

## Syntax:

```
{attribute 'displaymode':=<displaymode>}
```

The following definitions are possible

- Binary format

```
- {attribute 'displaymode':='bin'}
- {attribute 'displaymode':='binary'}
```

- Decimal format
- attribute 'displaymode':='dec'\}
- \{attribute 'displaymode':='decimal'\}
- Hexadecimal format
- \{attribute 'displaymode':='hex'\}
- attribute 'displaymode':='hexadecimal'\}

Insertion position: line above the line with the declaration of the corresponding variables.

## Example

```
VAR
    {attribute 'displaymode':='hex'}
    dwVar1: DWORD;
    END_VAR
```


## See also

- ${ }^{\text {B }}$, Chapter 6.4.1.21.3.8.24 "Command 'Display Mode' - 'Binary', 'Decimal', 'Hexadecimal'" on page 2654


## Attribute 'enable_dynamic_creation'

The pragma enable_dynamic_creation is needed for using the $\qquad$ NEW operator for function blocks.

## Syntax:

```
{attribute 'enable_dynamic_creation'}
```

Insert location: First line in the declaration of the function block.
See also

- ⓨ Chapter 6.4.1.20.3.59 "Operator '__NEW'" on page 2201


## Attribute 'estimated-stack-usage'

The pragma provides an estimated value for the stack size requirement.
Methods with recursive calls cannot pass a stack check because stack usage cannot be determined. As a result, a warning is issued. To prevent this warning, you can give the method an estimated value (in bytes) for the stack size requirement. Then the method passes the stack check successfully.

```
Syntax
{attribute 'estimated-stack-usage' := '<estimated stack size in
bytes>'}
```


## Example

```
{attribute 'estimated-stack-usage' := '127'} // 127 bytes
METHOD PUBLIC DoIt : BOOL
VAR_INPUT
END_VAR
```

Insert location: First line above the declaration part of the method.
The section "Method call" includes an example that uses this pragma.

Recursive Within its implementation, a method can call itself, either directly by means of the THIS pointer, method call or by means of a local variable for the assigned function block.

Use recursions mainly for processing recursive data types such as linked lists. In general, we recommend to be careful when using recursion, as unexpectedly deep recursions can cause stack overflow and machine downtime.

## Example

Calculation of

the factorial | The following program PLC_PRG calculates the factorial of a number in |
| :--- |
| function block in a different way, each in its own method. |
| - Method m_Iterative: Iterative |
| - Method m_Pragmaed: Recursive with warning suppression |
| - Method m_Recursive: Recursive |
| - Method m_Temp: Temporary with warning suppression |

```
// Contains the data of the factorial calculation of uiNumber
TYPE FACTORIAL RESULT :
STRUCT
            uiNumber : UINT;
            udiIterative : UDINT;
            udiRecursive : UDINT;
            udiPragmaed : UDINT;
            udiTemp : UDINT;
END_STRUCT
END_TYPE
PROGRAM PLC_PRG
VAR
    fb_Factorial_A : FB Factorial;
    fac̄torial_A : FACTORIAL_RESULT := (uiNumber := 9,
udiIterative := 0, udiRecursive := 0, udiPragmaed := 0 );
END VAR
fb_\overline{Factorial_A.p_Number := factorial_A.uiNumber;}
factorial_A.udiIterative := fb_Factorial_A.m_Iterative();
factorial_A.udiRecursive := fb_Factorial_A.m_Recursive(uiN :=
factorial_A.uiNumber);
factorial_A.udiPragmaed := fb_Factorial_A.m_Pragmaed(uiN :=
factorial_A.uiNumber);
factorial_A.udiTemp := fb_Factorial_A.m_Temp(uiN :=
factorial_A.uiNumber);
```

```
//Factorial calculation in different ways
FUNCTION_BLOCK FB_Factorial
VAR_INPUT
END_VAR
VAR_OUTPUT
END-VAR
VAR
    uiN : UINT;
    udiIterative : UDINT;
    udiPragmaed : UDINT;
    udiRecursive : UDINT;
END_VAR
// Iterative calculation
METHOD PUBLIC m_Iterative : UDINT
VAR
    uiCnt : UINT;
END VAR
m_ITErative := 1;
I\overline{F}}\mathrm{ UiN > 1 THEN
    FOR uiCnt := 1 TO uiN DO
            m_Iterative := m_Iterative * uiCnt;
    END_FO\overline{R;}
    RETŪRN;
ELSE
    RETURN;
END_IF;
//Recursive calculation with suppressed warning
{attribute 'estimated-stack-usage' := '99'}
METHOD PUBLIC m_Pragmaed : UDINT
VAR_INPUT
    uiN : UINT;
END VAR
VAR
END_VAR
m_Pragmaed := 1;
I\overline{F} uiN > 1 THEN
    m_Pragmaed := uiN * THIS^.m_Pragmaed(uiN := (uiN - 1));
    R\overline{ETURN;}
ELSE
    RETURN;
END_IF;
//Recursive calculation
METHOD PUBLIC m_Recursive : UDINT
VAR_INPUT
    uiN : UINT;
END_VAR
VAR
END_VAR
m_Rēcursive := 1;
IF uiN > 1 THEN
    m_Recursive := uiN * THIS^.m_Recursive(uiN := (uiN - 1) );
    RETURN;
ELSE
    RETURN;
END_IF;
// Called by temporary FB instance
{attribute 'estimated-stack-usage' := '99'}
METHOD PUBLIC m_Temp : UDINT
VAR_INPUT
    uiN : UINT;
END_VAR
```

```
VAR
    fb_Temp : FB_Factorial;
END VAR
m_Temp := 1;
IF uiN > 1 THEN
    m_Temp := uiN * fb_Temp.m_Temp(uiN := (uiN - 1));
    RETURN;
ELSE
    RETURN;
END_IF;
PROPERTY p_Number : UINT
uiN := p_Number; //Setter method
```

Only the m_Recursive issues a warning when the program is executed.


## See also

- $\Rightarrow$ Chapter 6.4.1.9.22.5 "Calling methods" on page 1933
- \& Chapter 6.4.1.9.22 "Object-Oriented Programming" on page 1929
- ${ }^{*}>$ Chapter 6.4.1.21.2.21.6 "Object 'Method'" on page 2485
- $\quad \stackrel{y}{c}$ Chapter 6.4.1.21.2.21.9 "Object 'Property'" on page 2493


## Attribute 'ExpandFully'

The effect of this pragma is that the components of an array used as an input variable for referenced visualizations are made visible in the Properties dialog box of the visualization.

Syntax:

```
{attribute 'ExpandFully'}
```

Insertion position: the line above the line with the declaration of the array.


## Attribute 'global_init_slot'

The pragma defines the initialization order of programming blocks and global variable lists.
Variables in a list (GVL or POU) are initialized from top to bottom.
If there are several global variable lists, then the initialization order is not defined.
The initialization does not apply for the initialization of literal values, for example 1, 'hello', 3.6 , or constants of base data types. However, you must define the initialization order yourself if there are dependencies between the lists. You can assign a defined initialization slot to a GVL or POU with the 'global_init_slot' attribute.

Constants are initialized before the variables and in the same order as the variables. During initialization, the POUs are sorted according to the value for <slot>. Then the code for initializing the constants is generated and afterwards the code for initializing the variables.

## Syntax:

```
{attribute 'global_init_slot' := '<slot>'}
```

<slot>: Integer value that defines the position in the call order. The default value for a POU (program, function block) is 50000. The default value for a GVL is 49990. A lower value means an earlier initialization. Caution: If several blocks or GVLs receive the same value for the 'global_init_slot' attribute, then the initialization order remains undefined.

Insert location: The pragma always affects the entire GVL or POU and therefore it must be located above the VAR_GLOBAL or POU declaration.

If several programming blocks have got assigned the same value for the attribute 'global_init_slot', the order of their initialization remains undefined.

```
Example A, which is initialized in GVL_2 with a value of 1000 .
```

```
GVL_1
```

GVL_1
GVL_2
GVL_2
PLC_PRG

```
PLC_PRG
```

```
    VAR_GLOBAL //49990
```

    VAR_GLOBAL //49990
    ```
    VAR_GLOBAL //49990
    A \ INT := GVL_2.B*100;
    A \ INT := GVL_2.B*100;
    A \ INT := GVL_2.B*100;
    END_VAR
    END_VAR
    END_VAR
    VAR_GLOBAL //49990
    VAR_GLOBAL //49990
    VAR_GLOBAL //49990
    B : INT := 1000;
    B : INT := 1000;
    B : INT := 1000;
        C : INT := 10;
        C : INT := 10;
        C : INT := 10;
    END_VAR
    END_VAR
    END_VAR
    PROGRAM PLC_PRG //50000
    PROGRAM PLC_PRG //50000
    PROGRAM PLC_PRG //50000
    VAR
    VAR
    VAR
        ivar: INT := GVL_1.A;
        ivar: INT := GVL_1.A;
        ivar: INT := GVL_1.A;
        ivar2: INT;
        ivar2: INT;
        ivar2: INT;
    END_VAR
    END_VAR
    END_VAR
    ivar:=ivar+1;
    ivar:=ivar+1;
    ivar:=ivar+1;
    ivar2:=GVL_2.C;
```

    ivar2:=GVL_2.C;
    ```
    ivar2:=GVL_2.C;
```

GVL_2

Using GVL_2. C in the implementation part of PLC_PRG is also not critical even without using a pragma because both GVLs are initialized before the program in either case.

## Attribute 'hide'

The program includes two global variable lists GVL_1 and GVL_2, as well as a PLC_PRG program that uses variables from both lists. GVL_1 uses the variable $B$ for initializing a variable

In this case, the compiler prints an error because GVL_2.B is used for initializing GVL_1.A before GVL_2 has been initialized. You can prevent this by using the global_init_slot attribute to position GVL_2 before GVL_1 in the initialization sequence.
In this example, GVL_1 must have at least one slot value of 49989 in order to achieve the earliest initialization within the program. Every lower value has the same effect:

GVL_2

```
{attribute 'global_init_slot' := '100'}
VAR_GLOBAL
    B}\mp@subsup{}{}{-}: INT := 1000
END_VAR
```

pragma because both GVLs are initialized before the program in either case.

Using the pragma \{attribute 'hide'\} to hide variables and POUs does not have the desired effect in most cases. Instead, you should use the pragma \{attribute 'conditionalshow'\}.

The pragma prevents the variables and POUs defined with it from being shown in the CODESYS user interface. As a result, you can intentionally hide these identifiers without restricting the access. This can be useful when you develop libraries.
Affected features:

- Library management
- Debugging
- Input Assistant
- Function "List components"
- Monitoring
- Symbol configuration

The variables or POUs defined with the pragma are neither visible in the Library Manager nor are they suggested in the Input Assistant or in the "List components" function. The pragma prevents those marked variables from being displayed in the symbol configuration. As a result, you cannot export these kinds of variables as symbols. The variables are also invisible in online mode, and therefore their values cannot be monitored. Moreover, you cannot use any debugging functionalities and you do not have any support when checking for bugs.

## Syntax:

\{attribute 'hide'\}
Insert location: For variables, above the line with the declaration of the variables. For POUs, in the first line.

If you, the application developer, know the exact instance path of the hidden POUs and variables, then you can access them in the code.

```
Example of
hidden variable local variable xInvisibleIn
```

```
FUNCTION_BLOCK FB_MyA
```

FUNCTION_BLOCK FB_MyA
VAR_INPUT
VAR_INPUT
iInA : INT;
iInA : INT;
{attribute 'hide'}
{attribute 'hide'}
xInvisibleIn : BOOL;
xInvisibleIn : BOOL;
xInit: BOOL;
xInit: BOOL;
END VAR
END VAR
VAR_OUTPUT
VAR_OUTPUT
iOutA : INT;
iOutA : INT;
END VAR
END VAR
VAR
VAR
iCounter : INT;
iCounter : INT;
END_VAR

```
END_VAR
```

Two instances of the function block FB_MyA are defined in the main program.

```
PROGRAM PLC_PRG
VAR
            fbMyA1, fbMyA2 : FB_MyA;
            xVar2 : BOOL;
            iVar1 : INT;
            iVar2 : INT;
END_VAR
fbMȳA1(iInA := 1, xInit := TRUE, xInvisibleIn := TRUE, iOutA =>
iVar1);
fbMyA2(iInA := 1, xInit := TRUE, iOutA => iVar2);
```

When the input value for $£$ bMyA1 is implemented, the "List components" function, which opens when you type $£ 6 M y A 1$. (in the implementation part of PLC_PRG), displays the variables iInA, xInit, and iOutA, but not the hidden variable xInvisibleIn.

```
Example of FB_A is a function block of the library HiddenFunctionality with the default namespace
hidden library
POU HIDDEN. To hide the identifier and the POU code from application developers, begin the declaration of the POU with the attribute pragma \{attribute 'hide' \}. To hide the subordinate POUs (actions, methods, properties, and transitions) in the same way, also begin their declarations with \{attribute 'hide'\}.
```

```
{attribute 'hide'}
```

{attribute 'hide'}
FUNCTION_BLOCK FB_A
FUNCTION_BLOCK FB_A
VAR_INPUT
VAR_INPUT
END_VAR
END_VAR
VAR-OUTPUT
VAR-OUTPUT
END_VAR
END_VAR
VAR
VAR
iA : INT;
iA : INT;
iCount : INT;
iCount : INT;
iInvisible : INT;
iInvisible : INT;
END
END
VAR
VAR
{attribute 'hide'}
{attribute 'hide'}
METHOD METH_Count : INT
METHOD METH_Count : INT
VAR_INPUT
VAR_INPUT
END_VAR
END_VAR
iCount := iCount + 1;
iCount := iCount + 1;
{attribute 'hide'}
{attribute 'hide'}
METHOD METH_Invisible : BOOL
METHOD METH_Invisible : BOOL
VAR_INPUT
VAR_INPUT
END_VAR
END_VAR
iInvisible := iInvisible + 1;
iInvisible := iInvisible + 1;
{attribute 'hide'}
{attribute 'hide'}
PROPERTY PUBLIC prop_iA : INT
PROPERTY PUBLIC prop_iA : INT
For you as the application developer, all POUs are invisible. You can use them only if you know the instance path.

```
```

PROGRAM PLC_PRG

```
PROGRAM PLC_PRG
VAR
VAR
    fbHidden : HIDDEN.FB_A; // Hidden function block from library
    fbHidden : HIDDEN.FB_A; // Hidden function block from library
HiddenFunctionality
HiddenFunctionality
    iCounter : INT;
    iCounter : INT;
END_VAR
END_VAR
fbHīdden.METH_Invisible();
fbHīdden.METH_Invisible();
iCounter := f\overline{b}Hidden.iInvisible;
iCounter := f\overline{b}Hidden.iInvisible;
In online mode, no monitoring is performed.
```


## PLC_PRG X

## Device.Application.PLC_PRG



With the pragma hide_all_locals you can hide all local variables of a POU.

See also

- \&y Chapter 6.4.1.20.6.3.18 "Attribute 'hide_all_locals'" on page 2289
- ${ }^{4}$ Chapter 6.4.1.20.6.3.8 "Attribute 'conditionalshow"' on page 2276
- \# Chapter 6.4.1.20.6.3.9 "Attribute 'conditionalshow_all_locals"' on page 2277


## Attribute 'hide_all_locals'

The pragma prevents all local variables of a signature from being visible in the display of the 'List components' function, in the Input Assistant or in the declaration part in online mode. Moreover, these variables are hidden in the symbol configuration and therefore cannot be exported as symbols. The pragma is especially useful in library POUs to hide POU variables from users.
Affected features

- Library management
- Debugging
- Input Assistant
- Function "List components"
- Monitoring
- Symbol configuration


## Syntax:

```
{attribute 'hide_all_locals'}
```

Insert location: First line above the declaration part of the POU

```
Example The function block FB_MyB uses the attribute:
```

```
{attribute 'hide_all_locals'}
```

{attribute 'hide_all_locals'}
FUNCTION_BLOCK F\overline{B}_MyB
FUNCTION_BLOCK F\overline{B}_MyB
VAR_INPUT
VAR_INPUT
iInB : INT;
iInB : INT;
{attribute 'hide'}
{attribute 'hide'}
xInvisibleIn : BOOL;
xInvisibleIn : BOOL;
xInit: BOOL;
xInit: BOOL;
END VAR
END VAR
VAR_OUTPUT
VAR_OUTPUT
iOutB : INT;
iOutB : INT;
END_VAR
END_VAR
VAR
VAR
iCounter : INT;
iCounter : INT;
xVar : BOOL;
xVar : BOOL;
END_VAR

```
END_VAR
```

Two instances of the function block FB_MyB are defined in the main program.

```
PROGRAM PLC_PRG
VAR
    fbMyB1, fbMyB2: FB_MyB;
    iVar3: INT;
    iVar4: INT;
END_VAR
fbMyB1(iInB := 2, xInvisibleIn := TRUE, iOutB => iVar3);
fbMyB2(iInB := 2, iOutB => iVar4);
IF fbMyB2.iCounter > 100 THEN
    fbMyB2.xInit := TRUE;
END_IF
```

Now when you download the program to the controller, start it, and switch to online mode, the variables iInB, xInit, iOutB, and xReset are displayed in the declaration editor. However, the hidden local variables iCounter and xVar are not displayed.

## See also

- ${ }^{*}>$ Chapter 6.4.1.20.6.3.17 "Attribute 'hide'" on page 2286


## Attribute 'initialize_on_call'

The pragma causes input variables of a function block to be initialized on each call of the function block. If an input variable is affected which expects a pointer and this pointer has been removed during an online change, then the variable is initialized to zero.

## Syntax:

```
{attribute 'initialize_on_call'}
```

Insert location: Always in the first line of the declaration part for the entire function block, and also in a line above the declaration of the individual input variable.

## Example

```
{attribute 'initialize_on_call'}
FUNCTION BLOCK fb
VAR_INPUT
    {attribute 'initialize_on_call'}
    pInt : POINTER TO INT \overline{:}=\overline{0};
    {attribute 'initialize_on_call'}
    iVal : INT := 0;
END_VAR
```


## Attribute 'init_namespace'

The effect of this pragma is that a variable of the type STRING or WSTRING, which is declared in a library function block with this pragma, is initialized when used in the project with the current namespace of the library.

## Syntax

```
{attribute 'init_namespace'}
```

Insertion position: the line above the line with the declaration of the variables in a library function block.

Example The function block "POU" is provided with the necessary attributes:

```
FUNCTION_BLOCK POU
VAR OUTPŪT
    {a\overline{trribute 'init_namespace'}}
    myStr: STRING;
END_VAR
```

An instance fb of the function block POU is defined within the main program PLC_PRG:

```
PROGRAM PLC_PRG
VAR
    fb:POU;
    newString: STRING;
END VAR
    newString := fb.myStr;
```

The variable myStr is initialized with the current namespace, for example MyLib. This value is assigned to newString in the main program.

## See also

- $\Leftrightarrow$ Chapter 6.4.1.21.2.16 "Object 'Library Manager'" on page 2469


## Attribute 'init_on_onlchange'

The effect of this pragma is that the variable to which the pragma is applied is initialized with each online change.

## NOTICE!

For compiler version 3.5.0.0 and later, a fast online change is performed for minor changes. In this case, only the modified blocks are compiled and downloaded. In particular, no initialization code is generated. This means that also no code is generated when variables with the init on onlchange attribute are initialized. As a rule, this has no effect because the attribute is used primarily for initializing variables with addresses. However, it cannot happen that a variable changes its address during an online change.
To secure the effect of the init_on_onlchange attribute in the entire application code, you must deactivate the fast online change in general for the application by using the compiler definition no_fast_online_change. To do this, insert the definition in the application "Properties" ("Build" $\overline{t a b}$ ).

## Syntax:

```
{attribute 'init_on_onlchange' }
```

Insert location: The line above the line with the declaration of the variables.
See also

- ${ }^{*}$ Chapter 6.4.1.21.4.11.5 "Dialog 'Properties' - 'Build'" on page 2755


## Attribute 'instance-path'

This pragma can be applied to a local STRING variable and causes this local STRING variable to be initialized in sequence with the device tree path of the POU to which it belongs. This can be useful for error messages. The application of the pragma requires the application of the attribute 'reflection' to the associated POU, as well as the application of the additional attribute 'noinit' to the STRING variable.

## Syntax:

```
{attribute 'instance-path'}
```

Insertion position: the line above the line with the declaration of the STRING variable.

```
Example The following function block contains the attributes 'reflection', 'instance-path' and
'noinit'.
    {attribute 'reflection'}
    FUNCTION_BLOCK POU
    VAR
        {attribute 'instance-path'}
        {attribute 'noinit'}
        str: STRING;
    END_VAR
An instance " \(m y P O U\) " of the function block " \(P O U\) " is defined within the main program "PLC_PRG":
```

```
PROGRAM PLC_PRG
```

PROGRAM PLC_PRG
VAR
VAR
myPOU:POU;
myPOU:POU;
myString: STRING;
myString: STRING;
END VAR
END VAR
myPŌU();
myPŌU();
myString:=myPOU.str;

```
myString:=myPOU.str;
```

Following the initialization of the instance myPOU, the path of the instance myPOU is assigned to the string variable str, in the example PLCWinNT. Application.PLC_PRG.myPOU. This path is assigned in the main program to the variable myString.

## NOTICE!

You can define the length of a string to be whatever you like (even $>255$ ), but you must consider that the string will be truncated at the end if it is assigned to a variable whose data type is too small for it.

See also

- «\& Chapter 6.4.1.20.6.3.41 "Attribute 'reflection"' on page 2313
- $\Leftrightarrow$ Chapter 6.4.1.20.6.3.32 "Attribute 'noinit"' on page 2299


## Attribute 'io_function_block', 'io_function_block_mapping'

With the 'io_function_block' attribute, you mark a function block in order to prepare it for the assignment to a channel in the I/O mapping of the device configuration. Then it is shown in the "Select function block" dialog.
With the 'io_function_block_mapping' attribute, you mark a parameter that should be used when mapping the $\overline{\mathrm{FB}}$ to a device channel in this kind of function block. You can provide the attribute to multiple parameters of the function block. For I/O mapping, the first one is used automatically whose type matches the channel (input, output, data type).

## Syntax:

```
{attribute 'io_function_block'}
{attribute 'io_function_block_mapping'}
```

Insert location: The line above the first line in the declaration of the function block, or the line above the parameter declaration.

## Example

```
{attribute 'io_function_block'}
FUNCTION_BLOCK-Scale_Output_Int
VAR_INPUT
    iInput : INT;
        iNumerator : INT;
        iDenominator : INT :=1;
        iOffset : INT := 0;
END VAR
VAR_OUTPUT
    {attribute 'io_function_block_mapping'}
        iOutput : INT;
END VAR
VAR
```


## See also

- ² Chapter 6.4.1.21.4.4 "Dialog 'Select Function Block'" on page 2746
- « "Linking a device with a function block instance" on page 1840


## Attribute 'is_connected'

You use the pragma 'is_connected' to mark a Boolean function block variable which, when a function module instance is called, provides information about whether the associated input of the POU has an assignment.
The use of the pragma requires the use of the attribute 'reflection' on the affected function block.

## Syntax:

```
{attribute 'is_connected' := '<input variable>'}
```


## Example In the function block FB , a local variable is declared for each input variable (in1 and in2)

 and the attribute 'is connected' is prepended to it each time with the name of the input variable. The func itself gets the pragma attribute 'reflection'.When an instance of the function block is called, the local variable is TRUE in the case that the input assigned to it has received an assignment.

```
{attribute 'reflection'}
FUNCTION BLOCK FB
VAR_INPUT
    in1: INT;
    in2: INT;
END_VAR
VAR
    {attribute 'is connected' := 'in1'}
    in1_connection_info: BOOL;
    {attribute 'is_connected' := 'in2'}
    in2_connection_info: BOOL;
END_VAR
```

Assumption: When the function block instance is called, in1 receives an external assignment and in 2 does not receive an assignment. This results in the following code:

```
in1_connection_info := TRUE;
in2_connection_info := FALSE;
```


## See also

- ${ }^{*}>$ Chapter 6.4.1.20.6.3.41 "Attribute 'reflection'" on page 2313
- $\quad>$ Chapter 6.4.1.21.2.21.3 "Object 'Function Block"' on page 2479


## Attribute 'linkalways'

The pragma \{attribute 'linkalways'\} instructs the compiler to always include a POU or a library POU in the compile information. During the build, the POU is compiled and is part of the application code. During the download, the POU is downloaded to the PLC.

## Syntax:

```
{attribute 'linkalways'}
```

Insertion location: The first line in the declaration part of the POU or library POU
The POU may be valid throughout the project (saved in the "POUs" view) or throughout the application (saved in the "Devices" view).

You can also select the "Link always" option in the "Build" tab of a POU's object properties.

```
Example The "MoreSymbols" GVL contains the pragma {attribute 'linkalways'}. The variables
    declared there are also part of the application code, regardless of any access.
GVL
MoreSymbols {attribute 'linkalways'}
VAR_GLOBAL
    g_iAlpha: INT;
    g_iBravo: INT;
    g_iCharlie: INT;
END_VAR
```

The symbol configuration also accesses the compile information. As a result, the variables of the MoreSymbols GVL are always provided for selection in the "Symbol Configuration" editor.


See also

- ② Chapter 6.4.1.21.4.11.5 "Dialog 'Properties' - 'Build'" on page 2755
- $\quad$ Chapter 6.4.1.10.3 "Symbol Configuration" on page 1941


## Attribute 'monitoring'

The effect of this pragma is that you can monitor values of properties or function calls in the online view of the IEC editor or in a watch list. There are two possible attribute values for this: 'variable' and 'call'

## Syntax

```
{attribute 'monitoring' := 'variable'}
```

\{attribute 'monitoring' := 'call'\}

Monitoring of programming objects and their properties

In the online view of a function block or program, you can monitor the subordinate properties in addition to the local variables. This allows you to monitor the values of the Get and Set methods.

Insert either the pragma \{attribute 'monitoring' := 'variable'\} or \{attribute 'monitoring' := 'call'\} in the declaration of the property block. The current values of the property are then displayed automatically in the IEC editor or in a watch list.

## Example In online mode, the PLC_PRG object shows the value of the Minutes property at the call location inline in the ST editor. This is because the pragma \{attribute 'monitoring' := 'variable'\} is located in the declaration of the Minutes property.



Check carefully for each application which attribute pragma is suitable for displaying the desired value. This depends on whether further operations with the variables are implemented within the property.

```
1. Pragma {attribute 'monitoring':='variable'}:
```

An implicit variable is created for the property, which is then always given the current property value when the application calls the Set or Get method. The value stored last in this variable is displayed in the monitoring.
2. Pragma \{attribute 'monitoring':='call'\}:

You can use this attribute only for properties that return simple data types or pointers, but not for structured types. The value to be monitored is read or written by calling the property directly. This means that the monitoring service of the runtime executes the Get or set method of the property.

## NOTICE!

When you insert the pragma \{attribute 'monitoring':='call'\} for monitoring, you have to pay attention to possible side effects. These kinds of side effects can occur if additional operations are implemented in the property.

## NOTICE!

The pragma \{attribute 'monitoring'\} is also evaluated for the symbol configuration. Only read access is possible for the value 'variable'.

With the context menu command "Add Watch", a variable on which the cursor is currently positioned is applied directly into the monitoring list in online mode.

The forcing or writing of functions is not supported. However, you can implicitly implement forcing by adding an additional input parameter for the respective function, which serves as an internal force flag.

Function monitoring is not possible in the compact runtime.

See also

- $\Leftrightarrow$ Chapter 6.4.1.21.2.21.9 "Object 'Property"' on page 2493


## Attribute 'monitoring_encoding'

The attribute pragma is allowed for string variables and aliases. The attribute has the effect that the values of the variables are marked with it to be decoded in UTF-8 format during monitoring.

## Syntax

```
{attribute 'monitoring_encoding' := 'UTF8'}
```



## Attribute 'no_assign', Attribute 'no_assign_warning'

The pragma 'no_assign' results in compiler errors being displayed if an instance of the function block is assigned to another instance of the same function block. Such assignments are often to be avoided if the function block contains pointers and pointers lead to problems, because they are copied as well during the value assignment.

The pragma 'no_assign_warning' results in the same as for the pragma 'no_assign' with compiler warnings instead of compiler errors.

## Syntax:

```
{attribute 'no_assign'}
```

Insert location: First line in the declaration part of a function block.

```
Example Assignment of function block instances containing pointers.
In this example the value assignment of the function block instances will lead to problems
during the execution of fb_exit:
```

```
VAR_GLOBAL
```

VAR_GLOBAL
ins\overline{t1 : TestFB;}
ins\overline{t1 : TestFB;}
awsBufferLogFile : ARRAY [0..9] OF WSTRING(66);(* Area: 0,
awsBufferLogFile : ARRAY [0..9] OF WSTRING(66);(* Area: 0,
Offset: 0x1304 (4868)*)
Offset: 0x1304 (4868)*)
LogFile : SEDL.LogRecord := (sFileName := 'LogFile.log',
LogFile : SEDL.LogRecord := (sFileName := 'LogFile.log',
pBuffer := ADR(awsBufferLogFile), udiMaxEntriesFile := UDINT\#10000,
pBuffer := ADR(awsBufferLogFile), udiMaxEntriesFile := UDINT\#10000,
udiMaxBuffered := UDINT\#10, uiLineSize := UINT\#64, wsSep := " ",
udiMaxBuffered := UDINT\#10, uiLineSize := UINT\#64, wsSep := " ",
xCircular := TRUE, siDateFormat := SINT\#O, siTimeFormat := SINT\#O);
xCircular := TRUE, siDateFormat := SINT\#O, siTimeFormat := SINT\#O);
END_VAR
END_VAR
PROGRAM PLC_PRG
VAR
inst2 : TestFB := inst1;
LogFileNew
END_VAR

```

In this case LogRecord manages a list of pointers, for which various actions are executed in the case of fb_exit. Problems result due to the assignment, because fb_exit will be executed twice. You should prevent this by adding the attribute ' no_assign' in the declaration of the function block "TestFB":
```

{attribute 'no assign'}
FUNCTION_BLOCK TestFB
VAR_INPUT

```

The following compiler errors are then displayed:
```

C0328: Assignment not allowed for type TestFB
C0328: Assignment not allowed for type LogRecord

```

If the pragma no_assign_warning is used instead of the pragma no_assign for the function block "TestF \(\bar{B}\) ", then the c0328 message is issued as compiler warning, not as a compiler error.

\section*{Attribute 'no_check'}

This pragma prevents the check function being called for the POU (POUs for implicit checks). Since the check functions can affect the processing speed of the program, it can be useful to apply the attribute to function blocks that have already been checked or are frequently called.

You add the pragma to the declaration of a POU.

\section*{Syntax:}
```

{attribute 'no_check'}

```

Insertion position: first line in the declaration part of the POU.

\section*{NOTICE!}

The attribute also automatically affects the child objects of a POU!
Example: If the attribute is entered in a program, check functions will also not be carried out for actions that are assigned to this program.

\section*{Attribute 'no_copy'}

In general an online change requires a re-allocation of instances, for example of a POU. In the process, the value of the variable contained in the instance is copied.
The pragma prevents the value of the variable contained in the instance from being copied in the course of an online change; instead, the variable is re-initialized in the course of an online change. This can be useful for a local pointer variable that points to a variable that has just been shifted by the online change and thus has a changed address.
You insert the attribute in the declaration part above the line of the declaration of the variables concerned.

\section*{Syntax:}
```

{attribute 'no_copy'}

```

\section*{Attribute 'no-exit'}

This attribute suppresses the call of the FB_exit method of a function block for a certain one of its instances. To do this you insert the attribute in the line before the declaration of the function block instance.

\section*{Syntax:}
```

{attribute 'no-exit'}

```

Example The method "FB_exit" is added to the function block "POU_ex". Two instances of the function block "POU_ex" are created in the main program "PLC_P \(\overline{R G}\) ".
```

PROGRAM PLC_PRG
VAR
POU1 : POU_ex;
{attribute ''no-exit'}
POU2 : POU_ex;
END_VAR
POU1 is called, POU2 is not called.

```

\section*{See also}
- \({ }^{\text {n }}\) Chapter 6.4.1.20.10 "Methods 'FB_Init', 'FB_Reinit', and 'FB_Exit"' on page 2336

\section*{Attribute 'noinit'}

This pragma is applied to variables that should not be implicitly initialized.

\section*{Syntax:}
```

{attribute 'no_init'}
{attribute 'no-init'}
{attribute 'noinit'}

```

Insertion position: line above the declaration line of the variables concerned in the declaration part.

\section*{Example}
```

PROGRAM PLC_PRG
VAR
A : INT;
{attribute 'no init'}
B : INT;
END_VAR

```

When the associated application is reset, the integer variable A is implicitly re-initialized with 0 , whereas the variable \(B\) retains its current value.

\section*{Attribute 'no_instance_in_retain'}

You can use this pragma to prevent the instance of a function block from being stored in the retain memory.

\section*{Syntax:}
```

{attribute 'no_instance_in_retain'}

```

Insert location:
Lines above the FUNCTION_BLOCK declaration in the declaration part of the function block.
Now when you declare an instance declaration of the function block as a RETAIN variable, an error message is issued.

See also
- ② Chapter 6.4.1.9.19 "Data Persistence" on page 1920

\section*{Attribute 'no_virtual_actions'}

The pragma is used for function blocks that are derived from a function block implemented in SFC and use the fundamental SFC sequence of this base class. The actions called from it exhibit the same virtual behavior as methods. This means that the implementations of the actions in the base class can be replaced by the derived class with its own specific implementations.

If you apply the pragma to the base class, then its actions are protected against overloading.

\section*{Syntax:}
```

{attribute 'no_virtual_actions'}

```

Insert location: Top line in the declaration part of the function block


The exemplary implementation of this sequence is limited to the initial step, followed by a single step with a linked step action ActiveAction. This step with a linked step action takes care of the configuration of the output variables.
```

an_int:=an_int+1; // Counting the action calls
te\overline{st_act:=' father_action';}
METH(); // Call of the method METH in order to set the
string variable test_meth

```

In the case of the derived class POU_child the step action is replaced by a special implementation of ActiveAction. Active Āction differs from the original only by the assignment of the string 'child_action' in place of 'father_action' at the variable test_act.

Likewise, the method METH, which assigns the string 'father_method' to the variable test_meth in the base class, is overwritten so that test_meth now gets the value 'child_method'. The main program PLC_PRG calls an instance of the function block POU_child, named Child. As expected, the value of the strings reflects the call of the action and method of the derived class:

PLC_PRG [CoDe5ys_SP_f_r_Win32: PLC Logic: Application] \
CoDe5ys_SP_f_r_Winsz2Application.PLC_PRG
\begin{tabular}{|c|c|c|c|}
\hline Expression & Type & Walue & Prep \\
\hline - \({ }^{\text {a }}\) Child & POU_child & & \\
\hline F \({ }^{\text {\% }}\), test_meth & STRING & 'thild_method' & \\
\hline \% \(\%\) test_act & STRING & 'child_action' & \\
\hline  & INT & 33 & \\
\hline
\end{tabular}

Now, however, you place the pragma \{attribute 'no_virtual_actions'\} in front of the base class:
```

{attribute 'no_virtual_actions'}
FUNCTION_BLOCK POU_SFC...

```

This changes the behavior: While the implementation of the derived class is still used for the method METH, the call of the step action now results in a call of the action ActiveAction of the base class. Therefore test_act is now given the value 'father_action':
\begin{tabular}{|c|c|c|c|}
\hline E] PLC_PRG[Co & Ys_5P_ & [ Logic: Application] & 4 \\
\hline \multicolumn{4}{|l|}{CoDeSys_SP_f_r_Wini2.Application.PLC_PRG} \\
\hline Expression & Type & Value & Frepa \\
\hline - \({ }^{\text {a }}\) Ghild & POU_Cthild & & \\
\hline 5 \% test_meth & STRING & 'child_method' & \\
\hline \% test_act & STRING & 'Father_action' & \\
\hline  & INT & 120 & \\
\hline
\end{tabular}

\section*{Attribute 'pingroup'}

The effect of this pragma is that the input pins or output pins (parameters) are grouped in the declaration of a function block. In the FBD/LD editor a pin group defined in this way can be displayed as an enlarged or reduced unit on the inserted function block. Several groups are possible and are distinguished by their names. CODESYS saves the respective state (reduced) per function block box with the project options.

\section*{Syntax:}
```

{attribute 'pingroup' := '<group name>'}

```

Insertion position: line above the declaration of the input or output variables concerned in the declaration part of a function block.
```

Example Two groups are defined: general (i1, out1) and group1 (i2, g1). r1, r2, outRes1
and g2 are always displayed

```
```

FUNCTION BLOCK FB

```
FUNCTION BLOCK FB
VAR_INPUT
VAR_INPUT
    r1 : REAL;
    r1 : REAL;
    {attribute 'pingroup' := 'general'}
    {attribute 'pingroup' := 'general'}
    i1 : INT;
    i1 : INT;
    {attribute 'pingroup' := 'group1'}
    {attribute 'pingroup' := 'group1'}
    i2 : INT;
    i2 : INT;
    r2 : REAL;
    r2 : REAL;
END_VAR
END_VAR
VAR_OUTPUT
VAR_OUTPUT
    outRes1 : REAL;
    outRes1 : REAL;
    {attribute 'pingroup' := 'general'}
    {attribute 'pingroup' := 'general'}
    out1 : INT;
    out1 : INT;
            {attribute 'pingroup' := 'group1'}
            {attribute 'pingroup' := 'group1'}
        g1 : INT;
        g1 : INT;
    g2 : REAL;
    g2 : REAL;
END_VAR
```

END_VAR

```

\section*{Attribute 'pin_presentation_order_inputs/outputs'}

The pragmas are evaluated in the CFC, FBD, and LD graphical editors, causing the order of inputs/outputs of the affected function block to be displayed as specified. You program the order by assigning the names of the inputs/outputs to the attribute in the desired order.

\section*{Syntax}
```

{attribute 'pin_presentation_order_inputs' := '<First_Input_Name>,
(<Next_Input_Name>, )* ( *, ) ? (<Next_Input_Name>,)*
<Last_Input_NName>'}
{attribute 'pin_presentation order outputs' := '<First_Output Name>,
(<Next_Output_Name>,)* ( *, )? (<Next_Output_Name>,)*
<Last_\overline{Output_N}=\=>>'}

```
- *

The terminal character serves as a wildcard for all inputs/outputs that are not specified in the display order. If the terminal character is missing, then the missing inputs/outputs are appended at the end.
- ( . . . ) ?

The contents of the parentheses are optional.
- ( . . . )*

The contents of the parentheses are optional again and can therefore occur not at all, one time, or several times.
- Insert location: First line in the declaration part of a function block.

NOTICE!
This pragma is not evaluated when pragma \{attribute 'pingroup' := '<Group_Name>' \} is used.
```

Example

```
```

{attribute 'pin_presentation_order_inputs' := 'input_2,*,input_1'}

```
{attribute 'pin_presentation_order_inputs' := 'input_2,*,input_1'}
{attribute 'pin_presentation_order_outputs' := 'outpūt_2, outpūt_1'}
{attribute 'pin_presentation_order_outputs' := 'outpūt_2, outpūt_1'}
FUNCTION_BLOCK POU_BASE
FUNCTION_BLOCK POU_BASE
VAR_INPUT
VAR_INPUT
    input_1 : BOOL;
    input_1 : BOOL;
        input_2 : INT;
        input_2 : INT;
        input_3 : INT;
        input_3 : INT;
        input_4 : INT;
        input_4 : INT;
END_VAR
END_VAR
VAR_OUTPUT
VAR_OUTPUT
    output_1 : BOOL;
    output_1 : BOOL;
    output_-2 : INT;
    output_-2 : INT;
    output_3 : INT;
    output_3 : INT;
    output_4 : BOOL;
    output_4 : BOOL;
END_VAR
END_VAR
FUNCTION_BLOCK PLC_PRG
FUNCTION_BLOCK PLC_PRG
VAR INPUT
VAR INPUT
END_VAR
END_VAR
VAR_OUTPUT
VAR_OUTPUT
END VAR
END VAR
VAR
VAR
    pouBase_A: POU_BASE;
    pouBase_A: POU_BASE;
END_VAR
```

END_VAR

```

In the representation of function module instance pouBase_A, the pragmas result in the following arrangement of input and output pins:


See also
- ⓢ Chapter 6.4.1.20.6.3.35 "Attribute 'pingroup"' on page 2302

\section*{Attribute 'obsolete'}

The effect of this pragma is that a defined warning is displayed for a data type definition during compilation if the data type (structure, function block, etc.) is used in the project. This enables you, for example, to draw attention to the fact that a data type is no longer valid because, for example, an interface has changed and this should also be implemented in the project.
In contrast to a message pragma this warning is defined centrally for all instances of a data type.

\section*{Syntax:}
```

{attribute 'obsolete' := 'user defined text'}

```

Insertion position: line of the data type definition or in a line above it.

\section*{Example The pragma is inserted in the definition function block fb1:}
```

{attribute 'obsolete' := 'datatype fb1 not valid!'}
FUNCTION BLOCK fb1
VAR_INPUT
i:INT;
END_VAR

```

If you use fb1 as a data type, for example in fbinst: fb1, the following warning will be displayed when compiling the project: "datatype fb1 not valid".

\section*{See also}
- \(\quad>\) Chapter 6.4.1.20.6.2 "Message Pragmas" on page 2269

\section*{Attribute 'pack_mode'}

The pragma defines how a data structure is packed during the allocation. The attribute has to be inserted above the data structure and affects the packing of the entire structure.

\section*{Syntax:}
```

{attribute 'pack_mode' := ' <pack mode value>' }

```

Insert location: above the declaration of the data structure

Table 401: Possible values for <value>:
\begin{tabular}{|c|l|l|}
\hline \begin{tabular}{c} 
<pack mode \\
value>
\end{tabular} & \begin{tabular}{l} 
Associated \\
packing \\
method
\end{tabular} & Description \\
\hline 0 & Aligned & All variables are allocated to byte addresses. There are no memory gaps. \\
\hline 1 & 1-byte-aligned & \\
\hline 2 & 2-byte-aligned & \begin{tabular}{l} 
There are \\
- 1-byte variables at byte addresses \\
- 2-byte variables at addresses divisible by 2. A maximum gap of 1 byte \\
results. \\
- 4-byte variables at addresses divisible by 2. A maximum gap of 1 byte \\
results. \\
- 8-byte variables at addresses divisible by 2. A maximum gap of 1 byte \\
results. \\
- Strings always at byte addresses. No gaps result.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|}
\hline <pack mode value> & Associated packing method & Description \\
\hline 4 & 4-byte-aligned & \begin{tabular}{l}
There are \\
- 1-byte variables at byte addresses \\
- 2-byte variables at addresses divisible by 2. A maximum gap of 1 byte results. \\
- 4 byte variables at addresses divisible by 4 . A maximum gap of 3 byte results. \\
- 8-byte variables at addresses divisible by 4. A maximum gap of 3 byte results. \\
- Strings always at byte addresses. No gaps result.
\end{tabular} \\
\hline 8 & 8-byte-aligned & \begin{tabular}{l}
There are \\
- 1-byte variables at byte addresses \\
- 2-byte variables at addresses divisible by 2. A maximum gap of 1 byte results. \\
- 4 byte variables at addresses divisible by 4 . A maximum gap of 3 byte results. \\
- 8 byte variables at addresses divisible by 8 . A maximum gap of 7 byte results. \\
- Strings always at byte addresses. No gaps result.
\end{tabular} \\
\hline
\end{tabular}

Depending on the structure, there may be no difference in the memory mapping of the individual modes. Therefore, the memory allocation of a structure with <pack mode value> \(=4\) can correspond to that of <pack mode value> \(=8\).

Arrays of structures: If the structures are combined in arrays, then bytes are added at the end of the structure so that the next structure is aligned.

\section*{NOTICE!}

If the "Compatibility layout" option is selected in the symbol configuration and at the same time the attribute 'pack_mode' is used in the code, then problems can occur due to unintentional memory misalignment.

\section*{See also}
- ⓨ Chapter 6.4.1.21.2.27 "Object 'Symbol Configuration'" on page 2523

\section*{Example 1}

Example
```

{attribute 'pack_mode' := '1'}
TYPE myStruct:
STRUCT
Enable: BOOL;
Counter: INT;
MaxSize: BOOL;
MaxSizeReached: BOOL;
END STRUCT
END_TY}P\textrm{PE

```

The memory range for a variable of the data type myStruct is allocated 'aligned'. If the storage address of its component Enable is \(0 \times 0100\), for example, then the component Counter follows at the address \(0 \times 0101\), MaxSize at address \(0 \times 0103\) and MaxSizeReached at address \(0 \times 0104\). In the case of 'pack_mode':=2, Counter would be at \(0 \times 0102\), MaxSize at \(0 \times 0104\) and MaxSizeReached at \(0 \times 0106\).

\section*{Example 2}
```

Example

```
```

STRUCT

```
STRUCT
    Var1 : BOOL := 16#01;
    Var1 : BOOL := 16#01;
    Var2 : BYTE := 16#11;
    Var2 : BYTE := 16#11;
    Var3 : WORD := 16#22;
    Var3 : WORD := 16#22;
    Var4 : BYTE := 16#44;
    Var4 : BYTE := 16#44;
    Var5 : DWORD := 16#88776655;
    Var5 : DWORD := 16#88776655;
    Var6 : BYTE := 16#99;
    Var6 : BYTE := 16#99;
    Var7 : BYTE := 16#AA;
    Var7 : BYTE := 16#AA;
    Var8 : DWORD := 16#AA;
    Var8 : DWORD := 16#AA;
END_TYPE
```

END_TYPE

```
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline & \multicolumn{2}{|l|}{pack_mode = 0} & \multicolumn{2}{|l|}{pack_mode = 1} & \multicolumn{2}{|l|}{pack_mode = 2} & \multicolumn{2}{|l|}{pack_mode = 4} & \multicolumn{2}{|l|}{pack_mode = 8} \\
\hline & Variable & Value & Variable & Value & Variable & Value & Variable & Value & Variable & Value \\
\hline 0 & Var1 & 01 & Var1 & 01 & Var1 & 01 & Var1 & 01 & Var1 & 01 \\
\hline 1 & Var2 & 11 & Var2 & 11 & Var2 & 11 & Var2 & 11 & Var2 & 11 \\
\hline 2 & Var3 & 22 & Var3 & 22 & Var3 & 22 & Var3 & 22 & Var3 & 22 \\
\hline 3 & ... & 00 & ... & 00 & ... & 00 & ... & 00 & ... & 00 \\
\hline 4 & Var4 & 44 & Var4 & 44 & Var4 & 44 & Var4 & 44 & Var4 & 44 \\
\hline 5 & Var5 & 55 & Var5 & 55 & & & & & & \\
\hline 6 & \(\ldots\) & 66 & \(\ldots\) & 66 & Var5 & 55 & & & & \\
\hline 7 & ... & 77 & ... & 77 & \(\ldots\) & 66 & & & & \\
\hline 8 & ... & 88 & ... & 88 & ... & 77 & Var5 & 55 & Var5 & 55 \\
\hline 9 & Var6 & 99 & Var6 & 99 & ... & 88 & ... & 66 & ... & 66 \\
\hline 10 & Var7 & AA & Var7 & AA & Var6 & 99 & \(\ldots\) & 77 & ... & 77 \\
\hline 11 & Var8 & AA & Var8 & AA & Var7 & AA & ... & 88 & ... & 88 \\
\hline 12 & ... & 00 & ... & 00 & Var8 & AA & Var6 & 99 & Var6 & 99 \\
\hline 13 & ... & 00 & ... & 00 & ... & 00 & Var7 & AA & Var7 & AA \\
\hline 14 & ... & 00 & ... & 00 & ... & 00 & & & & \\
\hline 15 & & & & & \(\ldots\) & 00 & & & & \\
\hline 16 & & & & & & & Var8 & AA & Var8 & AA \\
\hline 17 & & & & & & & ... & 00 & ... & 00 \\
\hline 18 & & & & & & & \(\ldots\) & 00 & \(\ldots\) & 00 \\
\hline 19 & & & & & & & \(\ldots\) & 00 & \(\ldots\) & 00 \\
\hline 20 & & & & & & & & & & \\
\hline 21 & & & & & & & & & & \\
\hline 22 & & & & & & & & & & \\
\hline 23 & & & & & & & & & & \\
\hline 24 & & & & & & & & & & \\
\hline 25 & & & & & & & & & & \\
\hline 26 & & & & & & & & & & \\
\hline 27 & & & & & & & & & & \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|c|}
\hline & \multicolumn{2}{|c|}{ pack_mode = 0 } & \multicolumn{2}{c|}{ pack_mode = 1 } & \multicolumn{2}{c|}{ pack_mode = 2 } & \multicolumn{2}{c|}{ pack_mode = 4 } & \multicolumn{2}{c|}{ pack_mode = 8 } \\
\hline & Variable & Value & Variable & Value & Variable & Value & Variable & Value & Variable & Value \\
\hline 28 & & & & & & & & & & \\
\hline 29 & & & & & & & & & & \\
\hline 30 & & & & & & & & & & \\
\hline 31 & & & & & & & & & & \\
\hline
\end{tabular}

\section*{Example 3}
```

Example

```
```

STRUCT

```
STRUCT
    Var1 : BYTE := 16#01;
    Var1 : BYTE := 16#01;
    Var2 : LWORD := 16#11;
    Var2 : LWORD := 16#11;
    Var3 : BYTE := 16#22;
    Var3 : BYTE := 16#22;
    Var4 : BYTE := 16#44;
    Var4 : BYTE := 16#44;
    Var5 : DWORD := 16#88776655;
    Var5 : DWORD := 16#88776655;
    Var6 : BYTE := 16#99;
    Var6 : BYTE := 16#99;
    Var7 : BYTE := 16#AA;
    Var7 : BYTE := 16#AA;
    Var8 : WORD := 16#AA;
    Var8 : WORD := 16#AA;
END_TYPE
```

END_TYPE

```
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|c|}
\hline & \multicolumn{2}{|c|}{ pack_mode = 0 } & \multicolumn{2}{c|}{ pack_mode = 1 } & \multicolumn{2}{c|}{ pack_mode = 2 } & \multicolumn{2}{c|}{ pack_mode = 4 } & \multicolumn{2}{c|}{ pack_mode = 8 } \\
\hline & Variable & Value & Variable & Value & Variable & Value & Variable & Value & Variable & Value \\
\hline 0 & Var1 & 01 & Var1 & 01 & Var1 & 01 & Var1 & 01 & Var1 & 01 \\
\hline 1 & Var2 & 11 & Var2 & 11 & & & & & & \\
\hline 2 & \(\ldots\) & 00 & \(\ldots\) & 00 & Var2 & 11 & & & & \\
\hline 3 & \(\ldots\) & 00 & \(\ldots\) & 00 & \(\ldots\) & 00 & & & & \\
\hline 4 & \(\ldots\) & 00 & \(\ldots\) & 00 & \(\ldots\) & 00 & Var2 & 11 & & \\
\hline 5 & \(\ldots\) & 00 & \(\ldots\) & 00 & \(\ldots\) & 00 & \(\ldots\) & 00 & & \\
\hline 6 & \(\ldots\) & 00 & \(\ldots\) & 00 & \(\ldots\) & 00 & \(\ldots\) & 00 & & \\
\hline 7 & \(\ldots\) & 00 & \(\ldots\) & 00 & \(\ldots\) & 00 & \(\ldots\) & 00 & & \\
\hline 8 & \(\ldots\) & 00 & \(\ldots\) & 00 & \(\ldots\) & 00 & \(\ldots\) & 00 & Var2 & 11 \\
\hline 9 & Var3 & 22 & Var3 & 22 & \(\ldots\) & 00 & \(\ldots\) & 00 & \(\ldots\) & 00 \\
\hline 10 & Var4 & 44 & Var4 & 44 & Var3 & 22 & \(\ldots\) & 00 & \(\ldots\) & 00 \\
\hline 11 & Var5 & 55 & Var5 & 55 & Var4 & 44 & \(\ldots\) & 00 & \(\ldots\) & 00 \\
\hline 12 & \(\ldots\) & 66 & \(\ldots\) & 66 & Var5 & 55 & Var3 & 22 & \(\ldots\) & 00 \\
\hline 13 & \(\ldots\) & 77 & \(\ldots\) & 77 & \(\ldots\) & 66 & Var4 & 44 & \(\ldots\) & 00 \\
\hline 14 & \(\ldots\) & 88 & \(\ldots\) & 88 & \(\ldots\) & 77 & & & \(\ldots\) & 00 \\
\hline 15 & Var6 & 99 & Var6 & 99 & \(\ldots\) & 88 & & & \(\ldots\) & 00 \\
\hline 16 & Var7 & AA & Var7 & AA & Var6 & 99 & Var5 & 55 & Var3 & 22 \\
\hline 17 & Var8 & AA & Var8 & AA & Var7 & AA & \(\ldots\) & 66 & Var4 & 44 \\
\hline 18 & \(\ldots\) & 00 & \(\ldots\) & 00 & Var8 & AA & \(\ldots\) & 77 & & \\
\hline 19 & & & & & \(\ldots\) & 00 & \(\ldots\) & 88 & & \\
\hline 20 & & & & & & & Var6 & 99 & Var5 & 55 \\
\hline 21 & & & & & & & Var7 & AA & \(\ldots\) & 66 \\
\hline 22 & & & & & & & Var8 & AA & \(\ldots\) & 77 \\
\hline 23 & & & & & & & \(\ldots\) & 00 & \(\ldots\) & 88 \\
\hline 24 & & & & & & & & & Var6 & 99 \\
\hline 25 & & & & & & & & & Var7 & AA \\
\hline 26 & & & & & & & & & Var8 & AA \\
\hline 27 & & & & & & & & & \(\ldots\) & 00 \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|c|}
\hline & \multicolumn{2}{|c|}{ pack_mode = 0 } & \multicolumn{2}{c|}{ pack_mode = 1 } & \multicolumn{2}{c|}{ pack_mode \(=\mathbf{2}\)} & \multicolumn{2}{c|}{ pack_mode = 4 } & \multicolumn{2}{c|}{ pack_mode = 8 } \\
\hline & Variable & Value & Variable & Value & Variable & Value & Variable & Value & Variable & Value \\
\hline 28 & & & & & & & & & & \\
\hline 29 & & & & & & & & & & \\
\hline 30 & & & & & & & & & & \\
\hline 31 & & & & & & & & & & \\
\hline
\end{tabular}

Behavior without pack mode

If pack mode is not used, then the compiler typically uses pack mode 4 or 8 , depending on the device description. In each case, a pack mode which is particularly beneficial for the processor is used so that memory access can be performed. This is also called natural alignment or a natural alignment of data.

Negative effects when using pack mode

Unaligned memory access can be the result of using the attribute 'pack_mode '. This means, for example, that a data type with a size of 4 bytes is then located at an address which is not divisible by 4 . Normally, on a 32 -bit system a 32 -bit data type can be read and written with a single memory access. On some platforms, for example on ARM platforms, this is possible only when this value is aligned in the memory. On other platforms, it can be that the access is possible but it is performed much more slowly.

\section*{Example}
```

{attribute 'pack_mode':=1}
TYPE DUT
STRUCT
by1 : BYTE;
dw1 : DWORD;
END_STRUCT
END_TYPE

```

On an ARM platform, the value dw1 cannot be read with a single access. When an attempt is made to access this element directly, the ARM processor will throw an exception.
Assumption: The following read access is performed: dwTest := dut1.dw1;
For this access to the DWORD dw1, four memory accesses are required because each byte is read, shifted, and disjuncted individually. The flow is somewhat the same as in the following example in which a DWORD is generated from an array of four bytes:
```

dwHelp := bytes[0];
dwResult := dwHelp;
dwHelp := bytes[1];
dwHelp := SHL(dwHelp, 8);
dwResult := dwResult OR dwHelp;
dwHelp := bytes[2];
dwHelp := SHL(dwHelp, 16);
dwResult := dwResult OR dwHelp;
dwHelp := bytes[3];
dwHelp := SHL(dwHelp, 24);
dwResult := dwResult OR dwHelp;

```

Obviously, this kind of access is much slower than access to a DWORD, which is aligned appropriately in the memory.
```

pdw := ADR(dut1.dw1);
dwTest := pdw^;

```

However, the compiler will not generate the access of the example when this kind of member is accessed by means of a pointer. This means that the following code results in an exception on an ARM platform.
```

pdw := ADR(dut1.dw1);

```
dwTest : \(=\mathrm{pdw}^{\wedge}\);

For performance reasons, you should therefore avoid working with structures which are not naturally aligned.

A packed structure must not contain an unpacked structure.

\section*{Attribute 'ProcessValue'}

With the 'ProcessValue' attribute, you mark a component of a structure. In the CFC editor, you can then use the command "Use attributed member as input" in order to connect this structure to an input of scalar type.

\section*{Syntax:}
```

{attribute 'ProcessValue'}

```

Insert location: Line above the affected structure variable.

\section*{Example}
```

TYPE QINT :
STRUCT
Status : STRING;
{attribute 'ProcessValue'}
Value1 : INT;
Value2 : INT;
END STRUCT
END_TYPE

```

See also
- \({ }^{\text {® }}\) Chapter 6.4.1.21.3.13.36 "Command 'Use Attributed Member as Input"' on page 2698

\section*{Attribute 'qualified_only'}

The effect of this pragma is that variables of a global variable list are only addressed by specifying the global variable name, for example gvl.g_var. This also applies to variables of the type Enumeration and can be helpful in avoiding being mistaken for local variables.

\section*{Syntax:}
\{attribute 'qualified_only'\}
Insertion position: line above VAR_GLOBAL in a GVL

\section*{Example Global Variable List "GVL":}
```

{attribute 'qualified_only'}
VAR GLOBAL
iVār:INT;
END_VAR

```

Within a POU, for example "PLC_PRG", the global variable iVar can only be addressed using the prefix GVL:
```

GVL.iVar:=5;

```

Conversely, the following incomplete call of the variable will create an error:
```

iVar:=5;

```

\section*{Attribute 'reflection'}

The pragma is used to identify POUs in which some variables require special treatment and are tagged with a specific attribute for this purpose. Currently, this applies to the attributes 'instance-path' and 'is-connected' for function block variables. The compiler searches only blocks marked with 'reflection' for variables with these attributes and therefore needs less time.

\section*{Syntax:}
```

{attribute 'reflection'}

```

For examples, see the description of the attributes 'instance-path' and 'is-connected'.
See also
- \({ }^{*}\) Chapter 6.4.1.20.6.3.22 "Attribute 'instance-path'" on page 2292
- \({ }^{3} \gg\) Chapter 6.4.1.20.6.3.24 "Attribute 'is_connected"' on page 2293

\section*{Attribute 'subsequent'}

The pragma is used to allocate consecutive variables in memory. When the list changes, the entire variable list is allocated to a new memory area. This pragma is used in programs and global variable lists.

\section*{Syntax:}
\{attribute 'subsequent'\}

\section*{NOTICE!}

VAR_TEMP in a program with attribute ' subsequent' leads to a compiler error.

When a variable in the list is qualified with RETAIN, all variables of the declaration part are stored in the memory area for RETAIN.

\section*{Attribute 'symbol'}

The pragma \{attribute 'symbol'\} defines which variables of a program or a global variable list are to be adopted into the symbol configuration. This means that the variables are exported as symbols to a symbol list. This symbol list is then available for external access both as an XML file in the project directory and as a file that is invisible to the user on the target system. For example, the symbol list is then available for access by an OPC server. The variables thus equipped with a symbol are loaded by CODESYS to the controller, even if they are not explicitly configured or visible in the editor of the symbol configuration.
In any case, however, an object "Symbol configuration" must be created below the application concerned in the device tree.

\section*{Syntax:}
\{attribute 'symbol' := '<access possibilities>'\}
<access possibilities>: none, read, write, readwrite. The default value readwrite applies if no parameter is specified.
Insertion position:
- in order to affect only an individual variable, you must place the pragma in the line before the variable declaration.
- In order to be effective for all variables in the declaration part of a program, you must place the pragma in the first line of the declaration editor. In this case, too, you can still set instructions for individual variables explicitly in the respective line.

Example With the following configuration the variables \(A\) and \(B\) are exported with read and write permission. Variable \(D\) is exported with read permission.
```

{attribute 'symbol' := 'readwrite'}
PROGRAM PLC_PRG
VAR
A : INT;
B : INT;
{attribute 'symbol' := 'none'}
C : INT;
{attribute 'symbol' := 'read'}
D : INT;
END_VAR

```

See also
- \(\xi^{*}\) Chapter 6.4.1.9.8 "Using Pragmas" on page 1888
- \(\sum^{4}\) Chapter 6.4.1.10.3 "Symbol Configuration" on page 1941

\section*{Attribute 'to_string'}

The pragma affects how the result of converting an enumeration component with the TO_STRING operator is output. If the enumeration declaration has the pragma, then the name of the enumeration component appears as a string instead of the numeric value.

\section*{Syntax:}
```

{attribute 'to_string'}

```

Insert location: First line above the declaration part of the enumeration.
```

Example Declaration of the enumeration color:
{attribute 'to_string'}
TYPE color :
l
red := 0,
blue := 1,
green := 2
);
END_TYPE
Conversion with TO_STRING:

```
```

PROGRAM PLC_PRG

```
PROGRAM PLC_PRG
VAR
VAR
    i_color: Color;
    i_color: Color;
    s_show_color: STRING;
    s_show_color: STRING;
END_VA\overline{R}
END_VA\overline{R}
i_color := 1;
i_color := 1;
s_show_color := TO_STRING(i_color);
s_show_color := TO_STRING(i_color);
In this case, str_show_color gets the value 'blue' instead of ' 1 ' as the conversion result.
```

See also

-     * Chapter 6.4.1.20.5.18 "Enumerations" on page 2263

Attribute 'warning disable', attribute 'warning restore'
This pragma causes certain warnings to be suppressed. The warning restore pragma causes a suppressed message to be reactivated.

## Syntax:

```
{warning disable <compiler ID>}
{warning restore <compiler ID>}
```

<compiler ID>: ID located at the beginning of an error or a warning message.

## Example Compiler message:

```
typify code ...
C0196: Implicit conversion from unsigned Type 'UINT' to signed Type
'INT' : possible change of sign
Compile complete -- O errors
```

Applying the pragma to a variable declaration:

```
VAR
    {warning disable C0195}
    test1 : UINT := -1;
        {warning restore C0195}
        test2 : UINT := -1;
    END_VAR
test1 does not generate an error message, test2 generates an error message.
```


## Effects of Pragmas on Symbols

POUs and variables can change their behavior with respect to the symbol configuration as a result of pragmas. A detailed description can be found on the help page of each pragma.

| Pragma with attribute | Effect | See also |
| :---: | :---: | :---: |
| ```{attribute 'call_after_global_init_slot' := ' <slot>''}``` | None |  |
| \{attribute 'call_after_init'\} | None |  |
| ```{attribute 'call_after_online_change_slot ' := ' <slo\overline{t}> '}``` | None |  |
| ```{attribute 'call_before_global_exit_slot' := ' <<slot>``` | None |  |
| ```{attribute 'call_on_type_change':= ' comma separated list of referenced function blocks> '}``` | None |  |
| ```{attribute 'conditionalshow' := ' <some text> '} {attribute 'conditionalshow'} {attribute 'conditionalshow_all_locals' : = ' <some text> ''} {attribute 'conditionalshow_all_locals'}``` | The marked variables are hidden and therefore cannot be exported. <br> However, if the source code file from the compiled library is available, or if CODESYS has been started with the command-line option conditionalshowsymbols, then the marked variables are visible despite the pragma. |  |
| ```{attribute 'const_replaced'} {attribute 'const_non_replaced'}``` | Replaced constants are not available in the symbol configuration editor and therefore cannot be exported. <br> A constant being replaced depends on whether or not the "Replace constants" compiler option has been selected for all constants and whether or not pragmas overwrite the compiler option for individual constants. | « Chapt er 6.4.1.20.6. 3.10 "Attribute 'const_re placē', Attribute 'const_n on_repla ced'"' on page 2278 |
| \{attribute 'dataflow'\} | None |  |
| \{attribute 'displaymode':= <displaymode> \} | None |  |
| \{attribute <br> 'enable_dynamic_creation'\} | None |  |
| \{attribute 'estimated-stackusage' := ' <estimated stack size in bytes> '\} | None |  |
| \{attribute 'ExpandFully' \} | None |  |
| ```{attribute 'global init_slot' := '<slot>'}``` | None |  |


| Pragma with attribute | Effect | See also |
| :---: | :---: | :---: |
| \{attribute 'hide'\} | Variables are hidden and therefore cannot be exported. |  |
| \{attribute 'hide_all_locals'\} | Variables are hidden and therefore cannot be exported. | 乡̌ Chapt er 6.4.1.20.6. 3.18 "Attribute 'hide_all locals'" on page 2289 |
| \{attribute <br> 'initialize_on_call'\} | None |  |
| \{attribute 'init_namespace'\} | None |  |
| \{attribute 'init_on_onlchange' \} | None |  |
| \{attribute 'instance-path'\} | None |  |
| ```{attribute 'io_function_block'} {attribute 'io_function_block_mapping'}``` | None |  |
| \{attribute 'is connected' := ' <input variable> '\} | None |  |
| \{attribute 'linkalways'\} | POUs and library POUs are integrated in the compile list and therefore cannot be exported. | ② Chapt er 6.4.1.20.6. 3.25 "Attribute 'linkal- ways"" on page 294 |
| ```{attribute 'monitoring' := 'variable'} {attribute 'monitoring' := 'call'}``` | Properties PROPERTY or functions (FUNCTION) are provided as symbols. | ¿ュ Chapt <br> er <br> 6.4.1.20.6. 3.26 <br> "Attribute <br> 'monitor- <br> ing'" <br> on page 220 |
| $\begin{aligned} & \text { \{'no_assign' \}} \\ & \text { \{'no_assign_warning' \} } \end{aligned}$ | None |  |
| \{attribute 'no_check'\} | None |  |
| \{attribute 'no_copy'\} | None |  |
| \{attribute 'no-exit'\} | None |  |
| ```{attribute 'no_init'} {attribute 'no-init'} {attribute 'noinit'}``` | None |  |
| \{attribute <br> 'no_instance_in_retain'\} | None |  |
| ```{attribute 'no_virtual_actions'}``` | None |  |


| Pragma with attribute | Effect | See also |
| :---: | :---: | :---: |
| \{attribute 'pingroup' := <group name> '\} | None |  |
| ```{attribute 'pin_presentation_order_inputs := '< input name > '( , <next input name> )* } {attribute 'pin_presentation_order_output s' := '< output nāme > ' '( , <next output name> )* }``` | None |  |
| \{attribute 'obsolete' := 'user defined text'\} | None |  |
| \{attribute 'pack_mode' := ' <pack mode value> '\} | Can lead to intentional memory misalignment | 4y Chapt er 6.4.1.21.2. 27 "Object 'Symbol Configu- ration'" on page 2523 |
| \{attribute 'ProcessValue'\} | None |  |
| \{attribute 'qualified_only'\} | None |  |
| \{attribute 'reflection'\} | None |  |
| \{attribute 'subsequent'\} | None |  |
| \{attribute 'symbol' := '<access possibilities>'\} | Variable is exported as symbol. The variable is displayed in the symbol list only when the "View", "Symbols Exported via Attribute" option is selected in the symbol configuration editor. The access rights, which have been defined with the pragma, are displayed In the "Attribute" column. | en Chapt er 6.4.1.20.0. "Attribute 'symbol'" on page 2314 |
| \{attribute 'to_string'\} | None |  |
| ```{warning disable <compiler ID> } {warning restore <compiler ID> }``` | None |  |

## See also

- $\Leftrightarrow$ Chapter 6.4.1.10.3 "Symbol Configuration" on page 1941
- $\Leftrightarrow$ Chapter 6.4.1.21.2.27 "Object 'Symbol Configuration"' on page 2523


## Conditional Pragmas

The purpose of conditional pragmas is to influence the generation of code in the pre-compilation process or the compilation process. The ST implementation language supports these pragmas.

## NOTICE!

They use conditional pragmas in the implementations of POUs. CODESYS does not evaluate these conditional pragmas if you use them in the declaration part.

With conditional pragmas you affect whether implementation code is taken into account for the compilation. For example, you can make this dependent on whether a certain variable is declared, whether a certain function block exists, etc.

| Pragma | Description |
| :---: | :---: |
| \{define <identifi |  |
| \{undefine <identi | f雨er $>$ define $\}$ statement of the identifier <identifier> is canceled, and the identifier is 'undefined' again from now on. The pragma is ignored if the specified identifier is not defined at all. |
| $\begin{aligned} & \{\text { IF <expr> }\} \ldots \\ & \{\text { ELSIF } \\ & \text { <expr> } \ldots \ldots \\ & \{\text { ELSE }\} \ldots \\ & \text { END_IF }\} \end{aligned}$ | These are pragmas for the conditional compilation. <br> The specified expressions <expr> must be constant at the time of compilation; they are evaluated in the order in which they appear here until one of the expressions indicates a non-zero value. The text linked to the instruction is compiled; the other lines are ignored. The order of the sections is fixed. The ELSIF and ELSE sections are optional. The ELSIF-segments may occur any number of times. You can use several conditional compilation operators within the constants <expr>. |
| <expr> | You can use one or more operators within the constant expression <expr> within the conditional compilation pragma \{IF\} or \{ELSIF\}. |

You can enter expressions and define definitions as "compiler definitions" in the "Build" tab in the Properties dialog of POUs. If you enter define definitions in the properties dialog, you must omit the term \{define \}, contrary to the definition in the implementation code. In addition, you can specify several define definitions in the properties dialog, separated by commas.

Operator This operator causes the expression to be given the value TRUE. The requirement is that defined )

Example Requirement: The applications App1 and App2 exist. The variable pdef1 is defined by a \{define\} statement in App1, but not in App2.

```
{IF defined (pdef1)}
(* This code is processed in App1 *)
{info 'pdef1 defined'}
        hugo := hugo + SINT#1;
{ELSE}
(* the following code is only processed in App2 *)
{info 'pdef1 not defined'}
    hugo := hugo - SINT#1;
{END_IF}
```

This also contains an example of a message pragma: Only the message pdef1 defined is displayed in the message view when the application is compiled, because pdef1 is actually defined. The message pdef1 not defined is displayed if pdef1 is not defined.

```
Operator This operator causes the expression to be given the value TRUE if the variable
defined
(variable:
<variable_nam
e>)
```

```
Example Requirement: The two applications App1 and App2 exist. The variable g_bTest is declared in
App1, but not in App2.
```

```
{IF defined (variable: g_bTest)}
```

{IF defined (variable: g_bTest)}
(* the following code is only processed in App2*)
(* the following code is only processed in App2*)
g_bTest := x > 300;
g_bTest := x > 300;
{END_\overline{IF}}

```
{END_\overline{IF}}
```

Operator The operator causes the expression to be given the value TRUE if a data type is declared with defined the identifier <identifier>; otherwise FALSE is returned.
(type:
<identifier>)
Example Requirement: The two applications App1 and App2 exist. The data type DUT is declared in App1, but not in App2.

```
{IF defined (type: DUT)}
(* the following code is only processed in App1*)
        bDutDefined := TRUE;
{END_IF}
```

Operator The operator causes the expression to be given the value TRUE if a function block or an action defined (pou: with name <pou-name> exists; otherwise FALSE is returned.
<pou name>)

```
Example Requirement: The two applications App1 and App2 exist. The function block CheckBounds
exists in App1, but not in App2.
```

```
{IF defined (pou: CheckBounds)}
```

{IF defined (pou: CheckBounds)}
(* the following code is only processed in App1 *)
(* the following code is only processed in App1 *)
arrTest[CheckBounds(0,i,10)] := arrTest[CheckBounds(0,i,10)] +
arrTest[CheckBounds(0,i,10)] := arrTest[CheckBounds(0,i,10)] +
1;
1;
{ELSE}
{ELSE}
(* the following code is only processed in App2 *)
(* the following code is only processed in App2 *)
arrTest[i] := arrTest[i]+1;
arrTest[i] := arrTest[i]+1;
{END_IF}

```
{END_IF}
```

| Operator <br> defined <br> (task: | Not yet implemented! <br> <task_name>) |
| :--- | :--- |
| The operator causes the expression to be given the value TRUE if a task is defined with the <br> name <task_name>; otherwise FALSE is returned. |  |
| Syntax | $\{$ IF \| ELSIF defined (task: <task name> \} |

```
Example {IF defined (task: Task_D) }
```

```
Example Requirement: The two applications App1 and App2 exist. The task PLC_PRG_Task is defined
in App1, but not in App2.
```

```
{END_IF}
```

{END_IF}

```
{END_IF}
{IF \overline{defined (task: PLC_PRG_Task)}}
{IF \overline{defined (task: PLC_PRG_Task)}}
{IF \overline{defined (task: PLC_PRG_Task)}}
(* the following code is on
(* the following code is on
(* the following code is on
    erg := plc_prg.x;
    erg := plc_prg.x;
    erg := plc_prg.x;
{ELSE }
{ELSE }
{ELSE }
(* the following code is only processed in App2 *)
(* the following code is only processed in App2 *)
(* the following code is only processed in App2 *)
    erg := prog.x;
    erg := prog.x;
    erg := prog.x;
{END_IF}
```

{END_IF}

```
{END_IF}
```

Operator
defined (resource:
<identifier>)

Operator defined (IsSimulation Mode)

Not yet implemented!
The operator causes the expression to be given the value TRUE if a resource object with the name <identifier> exists for the application; otherwise FALSE is returned.

```
Example Requirement: The two applications App1 and App2 exist. A resource object glob_var1 of the
Example Requirement: The two applications App1 and App2 exist. A resource object glob_var1 of the
```

global variable list exists for App1, but not for App2.

```
global variable list exists for App1, but not for App2.
{IF defined (resource:glob_var1)}
{IF defined (resource:glob_var1)}
{IF defined (resource:glob_var1)}
{IF defined (resource:glob_var1)}
{IF defined (resource:glob_var1)}
(* the following code is only processed in Appl *)
(* the following code is only processed in Appl *)
(* the following code is only processed in Appl *)
(* the following code is only processed in Appl *)
(* the following code is only processed in Appl *)
gvar_x := gvar_x + ivar;
gvar_x := gvar_x + ivar;
gvar_x := gvar_x + ivar;
gvar_x := gvar_x + ivar;
gvar_x := gvar_x + ivar;
{ELSE }
{ELSE }
{ELSE }
{ELSE }
{ELSE }
(* the following code is only processed in App2 *)
(* the following code is only processed in App2 *)
(* the following code is only processed in App2 *)
(* the following code is only processed in App2 *)
(* the following code is only processed in App2 *)
x := x + ivar;
x := x + ivar;
x := x + ivar;
x := x + ivar;
x := x + ivar;
{END_IF}
{END_IF}
{END_IF}
{END_IF}
{END_IF}
G
```

G

```
    

The operator causes the expression to be given the value TRUE if the application runs on a simulated device, i.e. in simulation mode.
See also
- \({ }^{*}\) Chapter 6.4.1.12.2 "Testing in simulation mode" on page 1980

\section*{Operator defined (IsLittleEndi an)}

Operator defined
(IsFPUSupport ed)

The operator causes the expression to be given the value FALSE, if the CPU memory is organized in Big Endian (Motorola byte order).

If the expression returns the value TRUE, then the code generator produces an FPU code (for the floating-point unit processor) when calculating with REAL values. Otherwise CODESYS emulates FPU operations, which is much slower.

This operator causes the expression to be given the value TRUE if the attribute <attribute> is specified in the first line of the declaration part of the function block <pou name>; otherwise FALSE is returned.
```

Example Requirement: The two applications App1 and App2 exist. The function fun1 is declared
in App1 and App2. However, in App1 it is also provided with the pragma {attribute
'vision'}.
In App1:

```
```

{attribute 'vision'}

```
{attribute 'vision'}
FUNCTION fun1 : INT
FUNCTION fun1 : INT
VAR_INPUT
VAR_INPUT
    i : INT;
    i : INT;
END_VAR
END_VAR
VAR
VAR
END_VAR
END_VAR
In App2:
```

```
FUNCTION fun1 : INT
```

FUNCTION fun1 : INT
VAR_INPUT
VAR_INPUT
i : INT;
i : INT;
END VAR
END VAR
VAR
VAR
END_VAR
END_VAR
Pragma
instruction: {IF hasattribute (pou: fun1, 'vision')}
(* the following code is only processed in App1 *)
ergvar := fun1(ivar);
{END_IF}

```

See also
- «Chapter 6.4.1.20.6.3.2 "User-defined attributes" on page 2272

Operator This operator causes the expression to be given the value TRUE if the pragma \{attribute hasattribute '<attribute>' \} is assigned to the variable in the line before the variable declaration; other(variable:
<variable>,
'<attribute>'
)
```

Example
Requirement: The two applications App1 and App2 exist. The variable g_global Int is used in App1 and App2, but in App1 the attribute 'DoCount' is assigned to it in addition.

```
```

Declaration of

```
Declaration of
g_GlobalInt
g_GlobalInt
in App1
in App1
```

VAR_GLOBAL

```
VAR_GLOBAL
    {attribute 'DoCount'}
    {attribute 'DoCount'}
    g_globalInt : INT;
    g_globalInt : INT;
    g_multiType : STRING;
    g_multiType : STRING;
END_VAR
END_VAR
Declaration
Declaration
g_GlobalInt
g_GlobalInt
in App2:
in App2:
VAR_GLOBAL
VAR_GLOBAL
    g_globalInt : INT;
    g_globalInt : INT;
    g_multiType : STRING;
    g_multiType : STRING;
END_VAR
END_VAR
Pragma
Pragma
instruction:
instruction:
{IF hasattribute (variable: g_globalInt, 'DoCount')}
{IF hasattribute (variable: g_globalInt, 'DoCount')}
    (* the following code is only processed in App1 *)
    (* the following code is only processed in App1 *)
    g_globalInt := g_globalInt + 1;
    g_globalInt := g_globalInt + 1;
{END_IF}
```

{END_IF}

```

See also
- \(\underbrace{*}\) Chapter 6.4.1.20.6.3.2 "User-defined attributes" on page 2272

Operator hasconstantty pe (constant name, boolean literal)

The operator checks whether or not the constant, which is identified with <constant name>, has been replaced. The second parameter (Boolean value) controls what is checked:
- TRUE: Checks if the constant has been replaced
- FALSE: Checks if the constant has not been replaced When the respective case occurs, the operator returns TRUE.

\section*{Syntax}
\{ IF hasconstanttype( <constant namne> , <boolean literal> ) \}
```

{ ELSIF hasconstanttype( <constant namne> , <boolean literal> ) }

```

\section*{Example}
```

{IF hasconstanttype(PLC_PRG.aConst, TRUE)}

```

The automatic replacement of constants in principle depends on the following:
- Compile option
- Replace constants
- Constant type (For example, STRING types are never replaced.)
- Usage of the attribute \{attribute 'const_non_replaced'\}
- Usage of the attribute \{attribute 'const_replaced'\}

\section*{Example}
```

VAR
iCntMAXIsReplaced: INT;
xErrorOccured : BOOL;
END_VAR
VAR CONSTANT
c iMAX: INT := 99;
END_VAR
{IF hasconstanttype(c iMAX, TRUE)}
iCntMAXIsReplaced := iCntMAXIsReplaced + 1;
{ELSE }
xErrorOccured := FALSE;
{END_IF}

```

Operator hasconstantva

The operator compares the value of the constant, which is identified with <constant name>, with lue ( the value of the second parameter. The second parameter can be specified either as a literal name,
variable
name, comparison operator)

Comparison operators <comparison operator>:
- Greater than (>)
- Greater than or equal to (>=)
- Equal to (=)
- Not equal to (<>)
- Less than or equal to (<=)
- Less than (<)

\section*{Syntax}
```

{ IF hasconstantvalue( <constant name> , <variable name> ,
<comparison operator> )
{ IF hasconstantvalue( <constant name> , <literal> , <comparison
operator> )
{ ELSIF hasconstantvalue( <constant name> , <variable name> ,
<comparison operator> )
{ ELSIF hasconstantvalue( <constant name> , <literal> , <comparison
operator> )

```

\section*{Example}
```

{IF hasconstantvalue(PLC_PRG.aConst, 99, >) }

```
\{ELSIF hasconstantvalue (PLC_PRG.aConst, GVL.intconst99, =) \}

\section*{Example}
```

PROGRAM PRG_ConditionConstantValue
VAR
iCntMAX: INT;
iCntGlobalMAX : INT;
iCntABC: INT;
iCntGlobalABC : INT;
xErrorOccured : BOOL;
END_VAR
VAR CONSTANT
c_iMAX: INT := 999;
c_sABC: STRING := 'ABC';
{attribute 'const_non_replaced'}
c_iNonReplaceable: IN\overline{T}}:=888
END_VĀR
{IF hasconstantvalue(c_iMAX, 999, =) }
iCntMAX := iCntMAX + 1;
{ELSE }
xErrorOccured := FALSE;
{END_IF}
{IF hasconstantvalue(c_iMAX, GVL.gc_iMAX, =) }
iCntGlobalMAX := i\overline{CntGlobalMAX + 1;}
{ELSE }
xErrorOccured := FALSE;
{END_IF}
{IF hasconstantvalue(c_sABC, 'ABC', =) }
iCntABC := iCntMAX + 1;
{ELSE }
xErrorOccured := FALSE;
{END_IF}
{IF hasconstantvalue(c_sABC, GVL.gc_sABC, =) }
iCntGlobalABC := i\overline{CntMAX + 1;}
{ELSE}
xErrorOccured := FALSE;
{END_IF}

```
```

Operator This operator causes the expression to be given the value TRUE if the variable <variable> is
hastype
(variable:
<variable>,
<type-spec>)
This operator causes the expression to be given the value TRUE if the variable <variable> is of the data type <type-spec>; otherwise FALSE is returned.
Possible data types for <type-spec>:

- BOOL
- BYTE
- DATE
- DATE_AND_TIME (DT)
- DINT
- DWORD
- INT
- LDATE
- LDATE_AND_TIME (LDT)
- LINT
- LREAL
- LTime
- LTIME_OF_DAY (LTOD)
- LWORD
- REAL
- SINT
- STRING
- TIME
- TIME_OF_DAY (TOD)
- ULINT
- UDINT
- UINT
- USINT
- WORD
- WSTRING

```

```

{IF (hastype (variable: g_multitype, LREAL))}

```
{IF (hastype (variable: g_multitype, LREAL))}
(* the following code is only processed in App1 *)
(* the following code is only processed in App1 *)
    g_multitype := (0.9 + g_multitype) * 1.1;
    g_multitype := (0.9 + g_multitype) * 1.1;
{ELSIF (hastype (variable: g_multitype, STRING))}
{ELSIF (hastype (variable: g_multitype, STRING))}
(* the following code is only processed in App2 *)
(* the following code is only processed in App2 *)
    g_multitype := 'this is a multitalent';
    g_multitype := 'this is a multitalent';
{END_\overline{IF}}
{END_\overline{IF}}
Operator The checked pack mode depends on the device description, not on the pragma that can be hasvalue
(PackMode, '<pack mode value>')
```

Operator hasvalue (RegisterSize , '<register size>')

``` specified for individual DUTs.
<register size>: Size of a CPU register (in bits) The operator causes the expression to return the value TRUE when the size of a CPU register is equal to <register size>.
Possible values for <register size>
- 16 for C 16 x
- 64 for X86-64 bit
- 32 for X86-32 bit
```

```
Operator
hasvalue
(<define-
ident>,
'<char-
string>')
This operator causes the expression to be given the value TRUE if a variable is defined with the hasvalue
(<defineident>,
'<char-
string>') identifier <define-ident> and has the value <char-string>; otherwise FALSE is returned.
Example Requirement: The two applications App1 and App2 exist. The variable test is used in the applications App1 and App2; in App1 it is given the value 1, in App2 the value 2.
```

```
{IF hasvalue(test,'1')}
```

{IF hasvalue(test,'1')}
(* the following code is only processed in App1 *)
(* the following code is only processed in App1 *)
x := x + 1;
x := x + 1;
{ELSIF hasvalue(test,'2')}
{ELSIF hasvalue(test,'2')}
(* the following code is only processed in App2 *)
(* the following code is only processed in App2 *)
x := x + 2;
x := x + 2;
{END_IF}

```
{END_IF}
```

Operator NOT <operator>

The expression is given the value TRUE if the reverse value of <operator> returns the value TRUE. <operator> can be one of the operators described in this chapter.

Example Requirement: The two applications App1 and App2 exist.PLC_PRG1 exists in App1 and App2, and the POU CheckBounds exists only in App1.

```
{IF defined (pou: PLC_PRG1) AND NOT (defined (pou: CheckBounds))}
(* the following code is only processed in App2 *)
        bANDNotTest := TRUE;
{END_IF}
```

Operator The expression is given the value TRUE if the two specified operators return TRUE.
<operator>
AND
<operator>

```
Example Requirement: The applications App1 and App2 exist.PLC_PRG1 exists in App1 and App2, the
POU CheckBounds only in App1.
{IF defined (pou: PLC_PRG1) AND (defined (pou: CheckBounds)) }
    (* the following \overline{code is only processed in App1 *)}
    bANDTest := TRUE;
{END_IF}
```

Operator

The expression returns TRUE if one of the two specified operators returns TRUE. <operator> R can be one of the operators described in this chapter.

Example Requirement: The two applications App1 and App2 exist. The POU PLC_PRG1 exists in App1 and App2, and the POU CheckBounds exists only in App1.

```
{IF defined (pou: PLC_PRG1) OR (defined (pou: CheckBounds))}
```

{IF defined (pou: PLC_PRG1) OR (defined (pou: CheckBounds))}
(* the following code is only processed in App1 and in App2 *)
(* the following code is only processed in App1 and in App2 *)
bORTest := TRUE;
bORTest := TRUE;
{END_IF}

```
{END_IF}
```


## Operator (<operator>)

See also

- ${ }^{\circ} \boldsymbol{y}$ Chapter 6.4.1.20.6.3.2 "User-defined attributes" on page 2272


## Region Pragma

This pragma is used for grouping several lines into one block in a text editor. The block can be named. Region pragmas can also be nested.
Code with region pragma: Expanded and collapsed views


The pragma can be used in the ST editor and all declaration editors. Syntax highlighting can be customized in the options.

See also

- ${ }^{4} \boldsymbol{y}$ Chapter 6.4.1.21.3.3.18 "Command 'Collapse All Folds'" on page 2568
- ${ }^{\circ} \boldsymbol{y}$ Chapter 6.4.1.21.3.3.17 "Command 'Expand All Folds'" on page 2568


### 6.4.1.20.7 Identifiers

Rules for identi- Rules for identifiers of variables

## fier designation

- An identifier must not contain spaces or special characters.
- Capitalization is ignored. For example, VAR1 and var1 refer to the same variable.
- The underscore is recognized. For example, A_BCD and AB_CD are treated as two different identifiers. Multiple consecutive underscores are not permitted.
- The length of an identifier is unrestricted.

Rules for multiple use of identifiers (namespaces)

- An identifier must not be declared two times locally.
- An identifier can be used more than one time globally. If a local variable has the same name as a global variable, then the local variable has priority within the POU.
- An identifier must not be identical to a keyword, such as the scope VAR_Global.
- A variable that is declared in a global variable list can have the same name as a variable defined in another GVL. CODESYS provides features that extend the standard for the namespace or scope of variables:
- Global namespace operator:

An instance path that begins with a dot always opens a global namespace. If there is a local variable (for example, ivar) that has the same name as a global variable, then you refer to the global variable as .ivar.

- The name of a global variable list can define the namespace uniquely for the include variables. Therefore, you can declare variables with the same name in different global variables list and still uniquely reference by prepending the list name.
For example, globlist1.ivar := globlist2.ivar; (* ivar from GVL globlist2 is copied to ivar in GVL globlist1 *).
- Variables that are defined in the global variable list of a library included in the project can be addressed uniquely according to the following syntax:
<name scope library>.< GVL name>.<variable name>
For example, globlist1.ivar := libl.globlist1.ivar (* ivar from GVL globlist1 in library lib1 is copied to ivar in GVL globlist1 *).
- When inserting a library, you also use the Library Manager to define a namespace. In this way, you can make unique references to a library block or library variable by <namespace library>.<block name|variable name>. Note that when libraries are nested, you have to reference the namespaces of all libraries are in succession
Example: If Lib1 is referenced by Lib0, then the POU func in Lib1 is addressed by Lib0.Lib1.fun: ivar := Lib0.Lib1.fun(4, 5); (* return value from func is copied to variable ivar in the project *)
We recommend that you apply the following rules in addition to the items that you have to consider specifically for variables declaration. By doing this, you get the best possible harmonization when assigning names.

Recommenda- Whenever possible, you should name variables in Hungarian notation in applications and librations for variable names ries. Find a meaningful, short, English name for each variable as a base name, which can consist of several words. Write the first letter of each word in uppercase, the remaining letters in lowercase. In front of the base name, append a prefix in lowercase to indicate the data type of the variable.

Example: iFileSize : INT;

| Data Type | Prefix | Description |
| :--- | :--- | :--- |
| BOOL | x | We expressly recommend x as the prefix for Boolean <br> variables in order to distinguish them from identifiers of <br> the data type BYTE. The prefix indicates the view of an <br> IEC programmer. |
|  | b | Reserved |
| BYTE | by | Bit string; not for arithmetic operations |
| WORD | w | Bit string; not for arithmetic operations |
| DWORD | dw | Bit string; not for arithmetic operations |
| LWORD | lw | Bit string; not for arithmetic operations |
|  | si |  |
| SINT | usi | Arithmetic integer data type, 8-bit |
| USINT | i | Arithmetic integer data type, 16-bit |
| INT | ui | Arithmetic integer data type, 16-bit |
| UINT | di | Arithmetic integer data type, 32-bit |
| DINT |  |  |


| Data Type | Prefix | Description |
| :---: | :---: | :---: |
| UDINT | udi | Arithmetic integer data type, 32-bit |
| LINT | li | Arithmetic integer data type, 64-bit |
| ULINT | uli | Arithmetic integer data type, 64-bit |
| REAL | r | Arithmetic floating-point data type, 32-bit |
| LREAL | $1 r$ | Arithmetic floating-point data type, 64-bit |
| STRING | S | Single-byte character string of variable length (default setting: 80 characters) |
| WSTRING | ws | Double-byte character string of variable length (default setting: 80 characters) |
| TIME | tim | Time duration, 32-bit |
| LTIME | ltim | Time duration, 64-bit |
| - TIME_OF_DAY <br> - TOD | tod | Time of day, 32-bit |
| - LTIME_OF_DAY <br> - LTOD | ltod | Time of day, 64-bit |
| - DATE_AND_TIME <br> - DT | $d t$ | Date and time |
| - LDATE_AND_TIME <br> - LDT | ldt |  |
| DATE | - dat <br> - d | Calender date |
| LDATE | - ldat <br> - ld | Calender date |
| POINTER | p |  |
| ARRAY | a |  |
| Enumeration | e |  |

## Example

```
VAR
    bySubIndix: BYTE;
    xFlag: BOOL;
    udiCounter: UDINT;
    END VAR
```

| Identifier | Description | Example |
| :---: | :---: | :---: |
| Nested declaration | Prefixes are attached successively in the order of declaration. | pabyTelegramData: POINTER TO ARRAY [0..7] OF BYTE; |
| Function block instance Variable of user-defined data type | Prefix: Abbreviation for the name of the function block or data type | ```cansdoReceivedTelegram: CAN_SDOTelegram; TYPE CAN_SDOTelegram : (* prefix: sdo *) STRUCT \squarewIndex: WORD; \squarebySubIndex:BYTE; \squarebyLen:BYTE; \squareaby: ARRAY [0..3] OF BYTE; END_STRUCT END TYPE``` |
| Local constant <br> Local constant variable | Prefix: c_, followed by the type prefix and the variable name | VAR CONSTANT $\begin{aligned} & \text { ■c_uiSyncID: UINT := 16\#80; } \\ & \text { END_VAR } \end{aligned}$ |
| Global variable | An additional prefix is appended to the library prefix. \| 9_ | $\begin{aligned} & \text { VAR_GLOBAL } \\ & \text { ロCAN_g_iTest: INT; } \\ & \text { END_VAR } \end{aligned}$ |
| Global constants <br> Global constant variable | An additional prefix is appended to the library prefix. \| 9c_ | ```VAR_GLOBAL CONSTANT \squareCAN_gc_dwExample: DWORD; END_VAR``` |

## Recommenda-

tions for vari-
able names
CODESYS V3.x
libraries

| Identifier | Description | Example |
| :--- | :--- | :--- |
| Variable | Corresponds to the <br> description for variable <br> names, with the exception <br> that global variables and <br> constants do not require <br> library prefixes because <br> the namespace replaces <br> the function. | g_iTest: INT; // Declaration <br> CAN.g_iTest; // Implementation; call in <br> the program |

## Recommenda-

tions for identi-
fiers for user-
defined data
types (DUT)

| Identifier for | Description | Example |
| :---: | :---: | :---: |
| Structures | Library prefix followed by an underscore and a short, informative description of the structure. The associated prefix for created variables of this structure should follow the colon as a comment. | ```TYPE CAN_SDOTelegram : (* prefix: sdo *) STRUCT \squarewIndex : WORD; \squarebySubIndex : BYTE; \squarebyLen : BYTE; \squareabyData: ARRAY [0..3] OF BYTE; END_STRUCT END_TYPE``` |
| Enumerations | Library prefix followed by an underscore and the identifier in uppercase. <br> Note: In past CODESYS versions, enumeration values > 16\#7FFF caused errors because they were not automatically converted to INT. For this reason, enumerations should always be declared with correct INT values. | TYPE CAL_Day : <br> ( <br> $\square C A L \_M O N D A Y$, <br> पCAL_TUESDAY, <br> ロCAL_WEDNESDAY, <br> पCAL_THURSDAY, <br> -CAL_FRIDAY, <br> पCAL_SATURDAY, <br> -CAL_SUNDAY <br> ); <br> Declaration: <br> eToday: CAL_Day; |

## Recommenda-

tions for identi-
fiers for user-
defined data
types (DUT) in
CODESYS V3
libraries

| Identifier for | Description | Example |
| :---: | :---: | :---: |
| DUT names in CODESYS V3 libraries | The namespace replaces the need for the library prefix. Therefore, it is omitted. Enumeration values are also defined without a library prefix. | Library with namespace CAL <br> TYPE DAY : <br> ( <br> GMONDAY <br> ロTUESDAY, <br> पWEDNESDAY, <br> $\square T H U R S D A Y$, <br> ロFRIDAY, <br> -SATURDAY, <br> QSUNDAY <br> ); <br> Declaration: <br> eToday: CAL.Day; <br> Usage in the application <br> IF eToday = CAL.Day.MONDAY THEN |

## Recommenda-

tions for identi-
fiers for POUs,
functions, func-
tion blocks, pro-
grams

| Identifier for | Description | Example |
| :--- | :--- | :--- |
| POUs: Func- <br> tions, function <br> blocks, pro- <br> grams | Library prefix followed by an underscore and <br> a short, informative POU name. Like for varia- <br> bles, the first letter of each word is uppercase <br> and all other letters are lowercase. We recom- <br> mend that you compose the POU name from a <br> verb and a noun. <br> For function blocks, the associated prefix for | FUNCTION_BLOCK CAN_SendTelegram (* <br> prefix: canst *) |
| created instances should follow the name as a |  |  |
| comment. |  |  |$\quad$| Actions |
| :--- |
| Only actions that the block itself calls, beginning <br> with prv_. Otherwise, actions do not have a <br> prefix. |

## Recommenda-

tions for identi-
fiers for POUs in CODESYS V3
libraries

| Identifier for | Description | Example |
| :--- | :--- | :--- |
| POU | The library prefix is omitted because the name-- <br> space replaces the function of the library prefix. | FUNCTION_BLOCK SendTelegram (* <br> prefix: canst *) |
| Method | Only methods that the block itself calls, begin- <br> ning with prv_. <br> Otherwise, methods do not have a prefix. |  |
| Interface <br> Interface | I | ICANDevice |

## Recommenda-

 tions for identifiers for visualizations
## NOTICE!

Note that a visualization is not named the same as another block in the project because this may cause problems when changing visualizations.

See also

- $\&$ Chapter 6.4.1.9.4 "Declaration of Variables " on page 1847
- $\quad$ Chapter 6.4.1.20.5 "Data Types" on page 2234
- $\quad$ « Chapter 6.4.1.20.2 "Variables" on page 2113


### 6.4.1.20.8 Shadowing Rules

In CODESYS, you are generally allowed to use the same identifier for different elements. For example, a POU and a variable can be named the same. However, you should avoid this practice in order to prevent confusion.
Negative example: In the following code snippet, a local function block instance has the same name as a function:

## Example

```
FUNCTION YYY : INT
;
END_FUNCTION
FUNCTION_BLOCK XXX
;
END_FUNCTION_BLOCK
PROGRAM PLC PRG
VAR
    YYY : XXX;
END VAR
YYY();
END_PROGRAM
```

In such a case as this, it is unclear whether the instance or the function is called in the program.

To make sure that names are always unique, you should follow naming conventions, such as certain prefixes for variables. Rules for assigning identifiers can be found in the "Identifiers" chapter of the help.
Naming conventions can be checked automatically using the static code analysis of CODESYS. Static code analysis could also detect the duplicate use of the name YYY and report it as an error.

The consistent use of the attribute qualified_only for enumerations and global variable lists and the use of qualified libraries can also prevent ambiguous situations.

To make sure that a POU of the same name in the "Devices" view is not called when a POU in the "POUs" view is called, the operator $\qquad$ POOL should be prepended (for example, svar_pou := __POOL.POU () ;) when the name of the POU is called.
Shadowing: The compiler does not report any errors or warnings if the same identifier is used for different elements. Instead, the compiler searches the code in a specific order for the declaration of the identifier. If a declaration is found, then the compiler does not search for any other declarations elsewhere. If other declarations do exist, then they are "shadowed" for the compiler. The following section describes the shadowing rules (that is, the search order that the compiler uses when searching for the declaration for identifiers). The section "Ambiguous access and qualified access" provides ways to prevent ambiguous access and bypass shadowing rules.

Search order in When the compiler encounters a single identifier in the code of an application, it searches for the application the corresponding declaration in the following order:

1. Local variables of a method
2. Local variables in the function block, program, or function, and in any base function blocks
3. Local methods of the POU
4. Global variables in the application, if the qualified_only attribute is not set in the variable list where the global variables are declared
5. Global variables in a parent application, if the qualified_only attribute is not set in the variable list where the global variables are declared
6. Global variables in referred libraries when neither the library nor the variable list requires qualified access
7. POU or type names from the application (that is, names of global variable lists, function blocks, and so on)
8. POU or type names from a parent application
9. POU or type names from a library
10. Namespaces of locally referred libraries and libraries that are published by libraries
11. Global variables in the "POUs" view, unless the qualified_only attribute is set in the variable list where they are declared
12. POU or type names from the "POUs" view (that is, names of global variable lists, function blocks, and so on)

Libraries that are inserted in the Library Manager of the "POUs" view are mirrored in the Library Manager in all applications in the project with the appropriate placeholder resolution. These libraries then form a common namespace with the libraries in the application. Therefore, there is no shadowing of libraries in the pool by libraries in the application.

Search order in When the compiler encounters a single identifier in the code of a library, it searches for the the library corresponding declaration in the following order:

1. Local variables of a method
2. Local variables in the function block, program, or function, and in any base function blocks
3. Local methods of the POU
4. Global variables in the local library, if the qualified_only attribute is not set in the variable list where the global variables are declared
5. Global variables in referred libraries when neither the library nor the variable list requires qualified access
6. POU or type names from the local library (that is, names of global variable lists, function blocks, and so on)
7. POU or type names from a referred library
8. Namespaces of locally referred libraries and libraries that are published by locally refereed libraries

Ambiguous Despite these search orders, ambiguous access can still occur. For example, this is the case
access and qualified access when a variable with the same name exists in two global variable lists that do not require qualified access. Such a case is reported by the compiler as an error (for example: ambiguous use of the name XXX).

This kind of ambiguous usage can be made unique by means of qualified access, for example by accessing via the name of the global variable list (example: GVL. XXX).

Qualified access can also always be used to avoid shadowing rules.

- The name of the global variable list can be used to uniquely access a variable in the list.
- The name of a library can be used to uniquely access elements in the library.
- The THIS pointer be used to uniquely access variables in a function block, even if a local variable with the same name exists in a method of the function block.

To find the declaration location of an identifier at any time, use the command "Edit $\rightarrow$ Browse $\rightarrow$ Go to Definition". This can be especially helpful if the compiler produces an apparently obscure error message.

Searching in The search orders described above do not apply to identifiers that exist as components in an instance paths instance path or to identifiers that are used as inputs in calls.

For access of the following type yy. component, it depends on the entity described by yy where the declaration of component is searched for.
If yy denotes a variable with a structured data type (that is, type STRUCT or UNION), then component is searched for in the following order:

- Local variables of the function block
- Local variables of the base function block
- Methods of the function block
- Methods of the base function block

If $y y$ denotes a global variable list or a program, then component is searched for in this list only.
If yy denotes a namespace of a library, then component is searched for in this library exactly as described in the section above "Search order in the library".
Only in the second instance does the compiler decide whether access to the found element is granted (that is, whether the variable is only locally accessible, or whether a method is private). If access is not allowed, an error is issued.

See also

- Ey Chapter 6.4.1.20.7 "Identifiers" on page 2327
- « ${ }^{*}$ Chapter 6.4.1.20.6.3.40 "Attribute 'qualified_only'" on page 2312
- $\Leftrightarrow$ Chapter 6.4.1.20.2.16 "THIS" on page 2126
- ${ }^{4}$ Chapter 6.4.1.20.3.74 "Operator '__POOL" on page 2217


### 6.4.1.20.9 Keywords

In all editors, you must capitalize keywords that for example denote scopes, data types, or operators.
Keywords cannot be used as variable names.

## Examples

```
VAR
END_VAR
BOOİ_TO_INT
IF
THEN
ELSE
LTIME
MUL
XOR
PERSISTENT
PROGRAM
```

CODESYS checks the correct use of keywords automatically and highlights errors immediately during input with a wavy underline.

When CODESYS creates implicit code, variables and functions are generally given a name that is prepended with two underscores (_). The use of double underscores in the implementation code is prevented automatically. This eliminates conflicts between internal system identifiers and identifiers assigned by the programmer.

The following keywords are used in the CODESYS export format. Therefore, you may not use them as identifiers:

- ACTION
- END_ACTION
- END_FUNCTION
- END_FUNCTION_BLOCK
- END_PROGRAM

Other valid keywords:

- VAR_ACCESS
- READ_ONLY
- READ_WRITE
- PARAMS


### 6.4.1.20.10 Methods 'FB_Init', 'FB_Reinit', and 'FB_Exit'

You can declare the methods explicitly in order to influence the initialization of function block variables, as well as the behavior when exiting function blocks.

The type of the return value for the implicit methods is BOOL. The value is not evaluated by the system, but the type should not be changed.

FB_Init is always available implicitly and it is used primarily for initialization. For a specific influence, you can also declare the methods explicitly and provide additional code there with the standard initialization code.
FB_Reinit must be implemented explicitly. If this method exists, then it is called after the instance of the affected function block is copied. That happens during an online change after changes to the function block declaration (signature change) in order to reinitialize the new instance module. To reinitialize the basic implementation of the function block, you must call FB_Reinit explicitly.

FB Exit must be implemented explicitly. If there is an implementation, then the method is called before the controller removes the code of the function block instance (implicit call).
The following shows some use cases of these methods for different operating conditions.

Operating con- When downloading an application to a PLC with factory settings, the memory of all variables dition "First download"

## Operating condition "Online Change"

 must be offset to the required initial state. In this way, the data areas of function block instances are assigned the required values. By the explicit implementation of FB_Init for function blocks, you can react specifically to this situation in the application code. By evaluating the method parameters bInCopyCode (FALSE) and bInitRetains (TRUE), you can detect this operating condition clearly. (See "Operating condition "Online Change"" and "Operating condition "Redownload"".)Within the scope of the online change, you can influence the initialization of function block instances by means of the methods FB Exit, FB_Init, and FB_Reinit. During the online change, the changes to the application that were made in offline mode are applied in the running PLC. This is the reason that the old instances of the function blocks are replaced by new instances as much as possible without incident. If no changes were made to the declaration part of a function block in the application before login, but in the implementation only, then the data areas are not replaced. Only code blocks are replaced. Then the methods FB_Exit, FB_Init, and FB_Reinit are not called.

If you have made changes to the declaration of a function block that lead to the copying operation described above, then you receive a message during the online change about possible unintended effects. In the "Details" of the message view, you see a list of all instances to be copied.

In the code of the FB_Init method, the parameter bInCopyCode (TRUE) can be evaluated to detect whether or not an online change is being executed.

The following calls occur in succession during an online change:

1. FB_Exit
old_inst.FB_Exit(bInCopyCode := TRUE);
You can call FB_Exit when exiting the old instance in order to trigger specific cleanup tasks before the copy operation. In this way, you can prepare the data for the following copy operation and influence the state of the new instance. You can notify other parts of the application about the pending change in location in the memory. Pay special attention to the variables of type POINTER and REFERENCE. These may no longer refer to the required memory locations after the online change. Interface variables (INTERFACE) are handled separately by the compiler and they are adapted accordingly during the online change. External resources such as sockets, files, or other handles can be applied by the new instance, in some case unchanged. Often they do not have to be treated specially during an online change. (See "Operating condition "Re-download"")
2. FB_Init
new_inst.FB_Init(bInitRetains := FALSE, bInCopyCode := TRUE);
FB_Init is called before the copy operation and can be used in order to execute specific operations for the online change. For example, you can initialize variables accordingly at the new location in the memory, or notify other parts of the application about the new location of specific variables in the memory.
3. Copy operation: copy
copy(\&old_inst, \&new_inst);
Existing values remain unchanged. For this purpose, they are copied from the old instance into the new instance.
```
4. FB_Reinit
    new_inst.FB_Reinit();
```

This method is called after the copy operation and should set defined values for the variables of the instance. For example, you can initialize variables accordingly at the new location in the memory, or notify other parts of the application about the new location of specific variables in the memory. Design the implementation independent of the online change. The method can also be called from the application at any time in order to reset a function block instance to its original state.


With the \{attribute 'no_copy'\} attribute, you can prevent that this is copied during the online change for a single variable of the function block. It always retains the initial value.

## See also

- ${ }^{\text {® }} \boldsymbol{y}$ Chapter 6.4.1.21.3.4.19 "Command 'Settings of Memory Reserve for Online Change' " on page 2595

Operating con- When downloading an application, an existing application may be replaced on the PLC. Theredition "New download" fore, the provision of memory for the present function blocks must be regulated. You can use the FB_Exit method for implementing the required steps for this. For example, you can offset external resources (with socket and file handles) in a defined state.
You can detect this operating condition by checking whether or not the parameter bInCopyCode = FALSE for the FB_Exit method.

Operating condition "Start of application"

```
Example T1 : TON := (PT:=t#500ms);
```

These kinds of assignments are executed only after calling FB_Init. In order to control the effects of these assignments, you can provide a function block or a method of a function block with the \{attribute 'call_after_init'\} attribute. You must add the attribute above the declaration part of the function block body and above the declaration part of the corresponding method. A POU that extends another POU which uses the \{attribute 'call_after_init'\} attribute must also have the attribute. For the benefit of clarity, we recommend that the corresponding methods are overwritten with the same name, the same signature, and the same attribute. This requires calling SUPER^. My Init. The name of the method can be chosen without restriction. (Exceptions: FB_Init, FB_Reinit, and FB_Exit). The method is called after processing the initial assignments and before starting the application tasks. Therefore, the method can react to user input.

When using FB_Init or \{attribute 'call_after_init'\}, remember that detecting errors in the FB_Init method or in methods decorated with the \{attribute 'call_after_init'\} attribute is tedious, because the setting of breakpoints may not have the expected effect.

## NOTICE!

If the explicitly defined initialization code is reached during execution, then the function block instance is already completely initialized via the implicit initialization code. Therefore, there must not be a SUPER^. FB_Init call.

## NOTICE!

FB_Init replaces the INI operator used in CoDeSys V2.3. The methods cannot be compared to the design of a constructor, such as in C\#, C++, or Java. This has consequences for function blocks that extend other function blocks. (See below: "Derived function blocks")

## Interface of method FB_Init

```
METHOD FB_Init : BOOL
VAR INPUT
    bİnitRetains : BOOL; // TRUE: the retain variables are initialized
(reset warm / reset cold)
    bInCopyCode : BOOL; // TRUE: the instance will be copied to the
copy code afterward (online change)
END_VAR
```

You can declare additional function block inputs in an FB_init method. Then you have to set these inputs in the declaration of the function block instance.

```
Example Method FB_Init for the serialdevice function block
METHOD PUBLIC FB_Init : BOOL
VAR INPUT
    bInitRetains : BOOL; // initializing of retain variable
    bInCopyCode : BOOL; // instance is copied to copy code
    iCOMnum : INT; // additional input: number of the COM
interface, that is to be observed
END_VAR
```

Instantiation of the serialdevice function block:

```
com1: serialdevice(iCOMnum:=1);
com0: serialdevice(iCOMnum:=0);
```


## Interface of method FB_Reinit

Interface o method FB_Exit

METHOD FB_Reinit : BOOL

There is the mandatory parameter bInCopyCode.

```
METHOD FB Exit : BOOL
VAR_INPUT
    bInCopyCode : BOOL; // TRUE: the exit method is called in order to
leave the instance which will be copied afterwards (online change).
END_VAR
```

Behavior for If a function block is derived from another function block, then the FB_Init method of the derived function derived function block must define the same parameters as the FB_Init method of the basic blocks function block. However, you can add further parameters in order to set up a special initialization for the instance.
Example $\quad$ The function blocks MainFB, SubFB, and SubSubFB are derived from each other. Therefore,
SubFB EXTENDS MainFB and SubSubFB EXTENDS SubFB apply.

Calling order of methods FB_Exit and FB_Init:

1. fbSubSubFb.FB_Exit(...);
2. fbSubFb.FB_Exit(...);
3. fbMainFb.FB_Exit(...);
4. fbMainFb.FB_Init(...);
5. fbSubFb.FB_Init(...);
6. fbSubSubFb.FB_Init(...);

## See also

- $\quad>$ Chapter 6.4.1.21.2.21.6 "Object 'Method'" on page 2485
- ${ }^{\text {• }}$ - Chapter 6.4.1.20.6.3.4 "Attribute 'call_after_init"' on page 2273
- $\quad$ - Chapter 6.4.1.20.6.3.30 "Attribute 'no_copy'" on page 2299
- \& Chapter 6.4.1.20.2.15 "SUPER" on page 2125


## Error Messages and Warnings

6.4.1.20.11.1 Compiler error C0001 ..... 2344
6.4.1.20.11.2 Compiler error C0002 ..... 2344
6.4.1.20.11.3 Compiler error C0003 ..... 2344
6.4.1.20.11.4 Compiler Error C0004 ..... 2345
6.4.1.20.11.5 Compiler error C0005 ..... 2345
6.4.1.20.11.6 Compiler error C0006 ..... 2346
6.4.1.20.11.7 Compiler error C0007 ..... 2346
6.4.1.20.11.8 Compiler error C0008 ..... 2346
6.4.1.20.11.9 Compiler error C0009 ..... 2347
6.4.1.20.11.10 Compiler error C0010 ..... 2347
6.4.1.20.11.11 Compiler error C0011 ..... 2347
6.4.1.20.11.12 Compiler error C0013 ..... 2348
6.4.1.20.11.13 Compiler error C0016 ..... 2348
6.4.1.20.11.14 Compiler error C0018 ..... 2348
6.4.1.20.11.15 Compiler error C0020 ..... 2349
6.4.1.20.11.16 Compiler error C0022 ..... 2349
6.4.1.20.11.17 Compiler error C0023 ..... 2349
6.4.1.20.11.18 Compiler error C0026 ..... 2350
6.4.1.20.11.19 Compiler error C0027 ..... 2350
6.4.1.20.11.20 Compiler error C0030 ..... 2350
6.4.1.20.11.21 Compiler error C0031 ..... 2351
6.4.1.20.11.22 Compiler error C0032 ..... 2351
6.4.1.20.11.23 Compiler Error C0033 ..... 2351
6.4.1.20.11.24 Compiler error C0035 ..... 2352
6.4.1.20.11.25 Compiler Error C0036 ..... 2352
6.4.1.20.11.26 Compiler error C0037 ..... 2352
6.4.1.20.11.27 Compiler error C0038 ..... 2353
6.4.1.20.11.28 Compiler error C0039 ..... 2353
6.4.1.20.11.29 Compiler error C0040 ..... 2354
6.4.1.20.11.30 Compiler error C0041 ..... 2354
6.4.1.20.11.31 Compiler Error C0042 (Compiler Version <= 3.4.10) ..... 2355
6.4.1.20.11.32 Compiler error C0043 ..... 2355
6.4.1.20.11.33 Compiler error C0044 ..... 2356
6.4.1.20.11.34 Compiler error C0045 ..... 2356
6.4.1.20.11.35 Compiler error C0046 ..... 2356
6.4.1.20.11.36 Compiler error C0047 ..... 2357
6.4.1.20.11.37 Compiler error C0048 ..... 2357
6.4.1.20.11.38 Compiler error C0049 ..... 2358
6.4.1.20.11.39 Compiler error C0050 ..... 2358
6.4.1.20.11.40 Compiler Error C0051 ..... 2358
6.4.1.20.11.41 Compiler Error C0053 ..... 2359
6.4.1.20.11.42 Compiler error C0061 ..... 2359
6.4.1.20.11.43 Compiler error C0062 ..... 2359
6.4.1.20.11.44 Compiler error C0064 ..... 2360
6.4.1.20.11.45 Compiler Error C0065 ..... 2360
6.4.1.20.11.46 Compiler error C0066 ..... 2360
6.4.1.20.11.47 Compiler error C0068 ..... 2361
6.4.1.20.11.48 Compiler error C0069 ..... 2361
6.4.1.20.11.49 Compiler error C0070 ..... 2362
6.4.1.20.11.50 Compiler error C0072 ..... 2362
6.4.1.20.11.51 Compiler error C0074 ..... 2362
6.4.1.20.11.52 Compiler error C0075 ..... 2363
6.4.1.20.11.53 Compiler error C0076 ..... 2363
6.4.1.20.11.54 Compiler error C0077 ..... 2363
6.4.1.20.11.55 Compiler Error C0078 ..... 2364
6.4.1.20.11.56 Compiler error C0080 ..... 2364
6.4.1.20.11.57 Compiler error C0081 ..... 2365
6.4.1.20.11.58 Compiler error C0082 ..... 2365
6.4.1.20.11.59 Compiler error C0084 ..... 2365
6.4.1.20.11.60 Compiler Error C0085 ..... 2366
6.4.1.20.11.61 Compiler error C0086 ..... 2366
6.4.1.20.11.62 Compiler error C0087 ..... 2367
6.4.1.20.11.63 Compiler error C0089 ..... 2367
6.4.1.20.11.64 Compiler error C0090 ..... 2368
6.4.1.20.11.65 Compiler error C0091 ..... 2368
6.4.1.20.11.66 Compiler error C0094 ..... 2368
6.4.1.20.11.67 Compiler error C0096 ..... 2369
6.4.1.20.11.68 Compiler error C0097 ..... 2369
6.4.1.20.11.69 Compiler error C0098 ..... 2370
6.4.1.20.11.70 Compiler Error C0099 (Compiler Version < 3.5.7.0). ..... 2370
6.4.1.20.11.71 Compiler error C0101 ..... 2371
6.4.1.20.11.72 Compiler error C0102 ..... 2371
6.4.1.20.11.73 Compiler error C0104 ..... 2371
6.4.1.20.11.74 Compiler error C0114 ..... 2371
6.4.1.20.11.75 Compiler Error C0115 ..... 2372
6.4.1.20.11.76 Compiler error C0116 ..... 2372
6.4.1.20.11.77 Compiler error C0117 ..... 2372
6.4.1.20.11.78 Compiler error C0118 ..... 2372
6.4.1.20.11.79 Compiler error C0119 ..... 2373
6.4.1.20.11.80 Compiler error C0120 ..... 2373
6.4.1.20.11.81 Compiler error C0122 ..... 2374
6.4.1.20.11.82 Compiler error C0124 ..... 2374
6.4.1.20.11.83 Compiler error C0125 ..... 2374
6.4.1.20.11.84 Compiler error C0126 ..... 2375
6.4.1.20.11.85 Compiler error C0130 ..... 2375
6.4.1.20.11.86 Compiler error C0131 ..... 2376
6.4.1.20.11.87 Compiler error C0132 ..... 2376
6.4.1.20.11.88 Compiler error C0136 ..... 2376
6.4.1.20.11.89 Compiler Error C0138 ..... 2377
6.4.1.20.11.90 Compiler error C0139 ..... 2377
6.4.1.20.11.91 Compiler error C0140 ..... 2377
6.4.1.20.11.92 Compiler error C0141 ..... 2378
6.4.1.20.11.93 Compiler error C0142 ..... 2378
6.4.1.20.11.94 Compiler error C0143 ..... 2378
6.4.1.20.11.95 Compiler error C0144 ..... 2379
6.4.1.20.11.96 Compiler error C0145 ..... 2379
6.4.1.20.11.97 Compiler error C0149 ..... 2380
6.4.1.20.11.98 Compiler error C0161 ..... 2380
6.4.1.20.11.99 Compiler error C0162 ..... 2380
6.4.1.20.11.100 Compiler Error C0164 ..... 2381
6.4.1.20.11.101 Compiler Error C0165 ..... 2381
6.4.1.20.11.102 Compiler error C0168. ..... 2382
6.4.1.20.11.103 Compiler error C0169. ..... 2382
6.4.1.20.11.104 Compiler Error C0173 ..... 2383
6.4.1.20.11.105 Compiler error C0174. ..... 2383
6.4.1.20.11.106 Compiler error C0175. ..... 2383
6.4.1.20.11.107 Compiler error C0177. ..... 2384
6.4.1.20.11.108 Compiler error C0178. ..... 2384
6.4.1.20.11.109 Compiler Error C0179 ..... 2385
6.4.1.20.11.110 Compiler Error C0180. ..... 2385
6.4.1.20.11.111 Compiler error C0182. ..... 2385
6.4.1.20.11.112 Compiler Error C0183. ..... 2386
6.4.1.20.11.113 Compiler error C0185. ..... 2386
6.4.1.20.11.114 Compiler Error C0186 ..... 2386
6.4.1.20.11.115 Compiler Error C0188. ..... 2387
6.4.1.20.11.116 Compiler error C0189. ..... 2387
6.4.1.20.11.117 Compiler error C0190. ..... 2388
6.4.1.20.11.118 Compiler error C0191. ..... 2388
6.4.1.20.11.119 Compiler error C0195. ..... 2388
6.4.1.20.11.120 Compiler error C0196. ..... 2388
6.4.1.20.11.121 Compiler error C0197. ..... 2389
6.4.1.20.11.122 Compiler error C0198. ..... 2389
6.4.1.20.11.123 Compiler error C0199. ..... 2389
6.4.1.20.11.124 Compiler error C0201 ..... 2390
6.4.1.20.11.125 Compiler error C0203. ..... 2390
6.4.1.20.11.126 Compiler error C0204. ..... 2391
6.4.1.20.11.127 Compiler error C0205 ..... 2391
6.4.1.20.11.128 Compiler error C0206. ..... 2391
6.4.1.20.11.129 Compiler Error C0207 ..... 2391
6.4.1.20.11.130 Compiler error C0208. ..... 2392
6.4.1.20.11.131 Compiler Error C0209 ..... 2392
6.4.1.20.11.132 Compiler error C0211. ..... 2392
6.4.1.20.11.133 Compiler error C0212. ..... 2393
6.4.1.20.11.134 Compiler Error C0215 ..... 2393
6.4.1.20.11.135 Compiler error C0216. ..... 2393
6.4.1.20.11.136 Compiler error C0217. ..... 2393
6.4.1.20.11.137 Compiler error C0218. ..... 2394
6.4.1.20.11.138 Compiler error C0219. ..... 2394
6.4.1.20.11.139 Compiler error C0221. ..... 2395
6.4.1.20.11.140 Compiler error C0222. ..... 2395
6.4.1.20.11.141 Compiler error C0224. ..... 2395
6.4.1.20.11.142 Compiler Error C0225 ..... 2396
6.4.1.20.11.143 Compiler error C0227. ..... 2396
6.4.1.20.11.144 Compiler error C0228. ..... 2397
6.4.1.20.11.145 Compiler Error C0230 ..... 2397
6.4.1.20.11.146 Compiler Error C0232. ..... 2397
6.4.1.20.11.147 Compiler Error C0233 ..... 2398
6.4.1.20.11.148 Compiler error C0234. ..... 2398
6.4.1.20.11.149 Compiler error C0235. ..... 2399
6.4.1.20.11.150 Compiler error C0236. ..... 2399
6.4.1.20.11.151 Compiler error C0237. ..... 2399
6.4.1.20.11.152 Compiler error C0238. ..... 2400
6.4.1.20.11.153 Compiler error C0239. ..... 2400
6.4.1.20.11.154 Compiler error C0240. ..... 2401
6.4.1.20.11.155 Compiler error C0241 ..... 2401
6.4.1.20.11.156 Compiler error C0242 ..... 2401
6.4.1.20.11.157 Compiler error C0243. ..... 2402
6.4.1.20.11.158 Compiler Error C0380 ..... 2402
6.4.1.20.11.159 Compiler error C0509. ..... 2403
6.4.1.20.11.160 Compiler error C0511 ..... 2404
6.4.1.20.11.161 Compiler Error C0542 ..... 2404
6.4.1.20.11.162 Compiler Error C0543 ..... 2405
6.4.1.20.11.163 Compiler Error C0549 ..... 2405
6.4.1.20.11.164 Compiler Error C0550 ..... 2406
6.4.1.20.11.165 Compiler Error C0554 ..... 2406
6.4.1.20.11.166 Compiler Error C0555 ..... 2407

## Compiler error C0001

Message: Constant '<constant value>' too large for type '<data type>'
Possible error cause: A typed constant is too large for the given data type or a constant is too large for each possible data type.
Error correction: Use smaller constants or an appropriate data type for a typed constant.

```
Example of the PROGRAM PLC_PRG
error:
VAR
            test1: INT;
            test2: INT;
            test3: LREAL;
END_VAR
test1 := 12345678912345566991923939292939911;
test2 := INT#123456;
test3 := 10E500;
--> C0001: Constant '12345678912345566991923939292939911' too large
for type 'ANY_INT'
--> C0001: Constant 'INT#123456' too large for type 'INT'
--> C0001: Constant '10E500' too large for type 'ANY_REAL'
```


## Compiler error C0002

Message: '<operator 1>' or '<operator 2>' expected instead of '<tag>'
Possible error cause: Syntax error
Error correction: Use the correct syntax.

| Example of the <br> error: | PROGRAM PLC_PRG <br>  <br> Fun (1; |
| :--- | :--- |
|  | $-->C 0002: \quad$ ',' or ')' expected instead of ';' |

## Compiler error C0003

Message: '<value>' is not a valid bit number for '<variable>'

Possible error cause: Attempted access to a bit that is outside of the range for a data type.
Error correction: Use a bit value for the bit access that is lower than the number of bits in the data type of the variable.

```
Example of the PROGRAM PLC_PRG
error:
VAR
        test1: WORD;
        test2: BOOL;
END VAR
test1 := test2.17;
--> C0003: '17' is not a valid bit number for 'w'
```


## Compiler Error C0004

Message: '<variable>' is not a component of '<structure>'
Possible error cause: Component access with "." to a variable that is not a structured value or does not exist as a component of the structure.

Error correction: Access a defined component, or change the definition of the component in the data type. The input assistance "List components" provides all valid access to this position.

```
Example of the TYPE DUT:
error: STRUCT
    x, y : INT;
    END STRUCT
END_TYPE
PROGRAM PLC_PRG
VAR
        test1 : DUT;
        test2 : INT;
    END VAR
test2 := test1.z;
--> C0032: Type 'Unknown type: 'test1.z'' cannot be converted to
type 'INT'
--> C0004: 'z' is to a component of 'DUT'
```


## Compiler error C0005

Message: Constant overflow in address '<address>'
Possible error cause: At least one component in the address does not fit into a 32-bit integer value.

Error correction: Use a valid address expression.

```
Example of the PROGRAM PLC_PRG
error:
VAR
        X: BYTE;
END VAR
X := %QB5555555555;
--> C0005: Constant overflow in address '%??'
```

```
Compiler error C0006
    Message: '<operator>' expected instead of '<token>'
    Possible error cause: Syntax error
    Error correction: Use the correct syntax.
Example of the PROGRAM PLC_PRG
error: VAR
        x: INT;
            bTest : BOOL;
    END_VAR
    IF bTest
        x := 9;
    END_IF
    --> C0006: 'THEN' expected instead of 'x'
```


## Compiler error C0007

Message: Expression expected instead of '<token>'
Possible error cause: Syntax error
Error correction: Use the correct syntax.

```
Example of the PROGRAM PLC_PRG
error: VAR
            x: INT;
            bTest : BOOL;
    END_VAR
    IF THEN
            x := 9;
    END_IF
    --> C0007: Expression expected instead of 'THEN'
```


## Compiler error C0008

Message: Unexpected end-of-file found: '<operator 1>', '<operator 2>', or '<operator 3>' expected

Possible error cause: Syntax error
Error correction: Use the correct syntax.

```
Example of the PROGRAM PLC_PRG
error:
VAR
            x: INT;
            bTest : BOOL;
        END
            VAR
        IF bTest THEN
            x := 9;
        --> C0008: Unexpected end-of-file found: 'ELSIF', 'ELSE' or 'END_IF'
        expected
```

```
Compiler error C0009
Message: Unexpected token '<token>' found
Possible error cause: Syntax error
Error correction: Use the correct syntax.
```



## Compiler error C0010

Message: Unexpected end-of-file found: '<token>' expected
Possible error cause: Syntax error
Error correction: Use the correct syntax.

```
Example of the PROGRAM PLC_PRG
error:
VAR
        i: INT;
    END VAR
    FOR i := 0 TO 2 DO
    ;
    --> C0010: Unexpected end-of-file 'END_FOR' found
```


## Compiler error C0011

Message: No 'CASE' label found
Possible error cause: Syntax error in a CASE statement. A statement in a CASE statement is not assigned to a CASE label.
Error correction: Add a CASE label.


## Compiler error C0013

Message: At least one statement is expected.
Possible error cause: At least one statement is expected at some positions in the code. For example, in the THEN and ELSE part of an IF statement, or in the body of a FOR loop.
Error correction: Add at least one statement at the selected position. It is enough to write a blank statement ";".

```
Example of the PROGRAM PLC_PRG
error:
VAR
        bTest: BOOL;
    END_VAR
    IF bTest THEN
END_IF
--> C0013: At least one statement is expected
```


## Compiler error C0016

Message: Counter initialization expected
Possible error cause: Syntax error in a FOR loop. The counter variable is not initialized correctly.
Error correction: Pay attention to the correct syntax of the FOR loop.

```
FOR i := 0 TO 10 DO
    ;
END_FOR
```

```
Example of the PROGRAM PLC_PRG
error:
VAR
            i: INT;
END_VAR
FOR i TO 10 DO
END_FOR
--> C0015: Counter initialization expected
Error correc- FOR i := 0 TO 10 DO
tion:
END_FOR
```


## Compiler error C0018

Message: <expression> is not a valid assignment target
Possible error cause: An expression with no write permission is on the left side of an assignment. Example: a constant.
Error correction: Assign only to variables that have write access.

| Example of the | PROGRAM PLC_PRG |
| :--- | :--- |
| error: | VAR $\quad$ i: INT; |
|  |  |
|  | END_VAR |
|  | VAR CONSTANT |

```
    j: INT := 0;
END_VAR
j := i;
--> C0018: 'j' is not a valid assignment target
```


## Compiler error C0020

Message: '<statement>' is no valid statement
Possible error cause: Syntax error (for example, too few or too many characters) Error correction: Make sure that the syntax is correct.

```
Example of the PROGRAM PLC_PRG
error:
VAR
            \(x\) : INT;
    END_VAR
\(x=2 ;\)
--> C0020: ' \((x=2) ;\) ' is no valid statement
Error correc- Example:
tion: \(\quad \mathrm{x}:=2\);
```


## Compiler error C0022

Message: '<operator>' needs exactly '<number of operands>' operands
Possible error cause: Too many or too few operands are assigned to an operator.
Error correction: Assign the required number of operands to the operator.

```
Example of the PROGRAM PLC_PRG
error: VAR
            i : INT;
            pt: POINTER TO INT;
END
            VAR
pt := ADR(i,1);
--> C0022: 'ADR' needs exactly '1' operands
Error correc- Example:
tion: pt := ADR(i);
```


## Compiler error C0023

Message: '<operator>' needs at least '<number of operands>' operands
Possible error cause: Too few operands are assigned to an operator.
Error correction: Assign the required number of operands to the operator.

```
Example of the PROGRAM PLC_PRG
error: VAR
            i : INT;
END VAR
i := MUX (30,40);
--> C0023: 'MUX' needs at least '3' operands
Error correc- Example:
tion: i := MUX (30,40,50);
```


## Compiler error C0026

Message: Identifier expected instead of '<invalid identifier>'
Possible error cause: An invalid identifier is passed to a method.
Error correction: Use valid identifiers.

| Example of the | METHOD 123 |
| :--- | :--- |
| error: | VAR_INPUT |
|  | END_VAR |
|  | --> C0243: The name used in the signature is not identical to the <br>  <br> object name <br>  <br> $-->C 0026: ~ I d e n t i f i e r ~ e x p e c t e d ~ i n s t e a d ~ o f ~ ' 123 ' ~$ |
| Error correc- | Example: <br> tion: |

## Compiler error C0027

Message: size of string expected after '('
Possible error cause: The length of the string is not specified.
Error correction: Specify a string length between the parentheses.

```
Example of the PROGRAM PLC_PRG
error:
VAR
        str : STRING();
    END_VAR
--> C0027: size of string expected after '('
--> C0006: ';, :=, REF=, ( or [' expected instead of ')'
Error correc- Example:
tion: str : STRING(100);
```


## Compiler error C0030

Message: Direct Address expected after 'AT' instead of '<identifier>'
Possible error cause: Either an invalid address or no address is assigned after 'AT'.
Error correction: Specify a valid address.


```
--> C0030: Direct Address expected after 'AT' instead of 'ABC'
Error correc- Example:
tion: i AT %IWO : INT;
```


## Compiler error C0031

Message: Type definition expected instead of '<no data type>'
Possible error cause: An invalid type definition is assigned to the identifier.
Error correction: Specify a valid type definition.

```
Example of the PROGRAM PLC_PRG
error: VAR
    i : 0;
END_VAR
--> C0031: Type definition expected instead of '0'
Error correc- Example:
tion: i : INT;
```


## Compiler error C0032

Message: Type '<type 1>' can not be converted to type '<type 2>'
Possible error cause: A variable is assigned to another variable with an incompatible type.
Error correction: Use a type conversion.

```
Example of the PROGRAM PLC_PRG
error: VAR
            test1: INT;
            test2: STRING;
    END_VAR
    test1 := test2;
    -->C0032: Type 'STRING' cannot be converted to type 'INT'
Error correc- Example:
tion: test1 := TO_INT(test2);
```


## Compiler Error C0033

Message: Type '<pointer type>' possibly not convertible to type '<data type>' .
Possible error cause: This error occurs only when checking pool objects. An attempt was made to convert a pointer to an integer. Because the size of pointers in a library is unknown, errors may occur when using the library.
Error correction: Use the type $\qquad$ UXINT or $\qquad$ XWORD for platform-independent calculations with pointers.

```
Example of the PROGRAM PLC_PRG
error: VAR
            ptr : POINTER TO INT;
            dw : DWORD;
END_VAR
            dw := ptr;
--> C0033: Type 'POINTER TO INT' possibly not convertible to type
'DWORD'.
```


## Compiler error C0035

Message: Program name, function or function block instance expected instead of '<invalid function>'

Possible error cause: A function is called that does not exist.
Error correction: Make sure that only program names, functions, and function Block Instances that exist are called.

```
Example of the PROGRAM PLC_PRG
error: VAR
END_VAR
PLC_PRG.METH1();
METHOD METH
VAR_INPUT
END VAR
--> C0004: 'METH1' is no component of 'PLC PRG'
--> C0035: Program name, function or functíion block instance expected
instead of 'PLC_PRG.METH1'
Error correc- Example:
tion: PLC_PRG.METH();
```


## Compiler Error C0036

Message: Cannot call object of type <type>
Possible error cause: An attempt has been made to call an object that does not support any calls.

Error correction: Only functions, function blocks, programs, methods, and actions can be called.

```
Example of the VAR_GLOBAL GVL
error:
```

```
        value : INT;
```

        value : INT;
    END_VAR
END_VAR
PRO\overline{GRAM PLC_PRG}
PRO\overline{GRAM PLC_PRG}
GVL();
GVL();
--> C0036: Cannot call object of type 'VAR_GLOBAL`.

```
--> C0036: Cannot call object of type 'VAR_GLOBAL`.
```


## Compiler error C0037

Message: '<invalid input>' is no input of '<function name>'

Possible error cause: A local variable is defined in a function call.
Error correction: Declare the variable as an input parameter.

```
Example of the PROGRAM PLC_PRG
error: VAR
    i : INT;
END_VAR
i := TEST(iVar := 1);
FUNCTION TEST : INT
VAR
    iVar : INT;
END_VAR
--> C0037: 'iVar' is no input of 'TEST'
Error correc- Example:
tion: VAR INPUT
-iVar : INT;
END_VAR
```


## Compiler error C0038

Message: '<invalid output>' is no output of '<function name>'
Possible error cause: A local variable is handled as an output in a function call.
Error correction: Declare the variable as an output parameter.

```
Example of the PROGRAM PLC_PRG
error: VAR
            i : INT;
            x : INT;
END VAR
i := TEST(iVar => x);
FUNCTION TEST : INT
VAR
            iVar : INT;
END_VAR
--> C0038: 'iVar' is no output of 'TEST'
Error correc-
tion:
Example:
VAR_OUTPUT
    iVar : INT;
END VAR
```


## Compiler error C0039

Message: VAR_IN_OUT '<invalid variable>' must be assigned in call of '<function block name>' Possible error cause: An IN_OUT variable is not passed to a function block that requires an IN_OUT variable.

Error correction: Assign the IN_OUT variable.

```
Example of the PROGRAM PLC_PRG
error: VAR
    inst : FB;
    END VAR
    inst();
    FUNCTION_BLOCK FB
VAR_IN_OUT
        inout : INT;
END VAR
--> C0039: VAR_IN_OUT 'inout' must be assigned in call of 'FB'
Error correc- Example:
tion: inst(inout := i);
```


## Compiler error C0040

Message: Function '<function name>' requires exactly '<number of inputs>' input
Possible error cause: Too many or too few parameters are passed to the called function.
Error correction: Pass exactly as many parameters to the function as are expected.

```
Example of the PROGRAM PLC_PRG
error:
VAR
            i : INT;
END VAR
i := TEST(1,2);
FUNCTION TEST : INT
VAR_INPUT
            IN: INT;
END VAR
--> C0040: Function 'TEST' requires exactly '1' inputs
Error correc- Example:
tion: i := Test(1);
```


## Compiler error C0041

Message: VAR_IN_OUT parameter '<parameter name>' of '<function name>' needs variable with write access as input

Possible error cause: The passed parameter is not a variable with write access (but a constant for example).

Error correction: Pass a VAR_IN_OUT parameter with write access to the function.

| Example of the error: | PROGRAM PLC_PRG VAR |
| :---: | :---: |
|  | i : INT; |
|  | x : INT; |
|  | END_VAR |
|  | i : = Test(31415); |
|  | FUNCTION TEST : INT |

```
VAR_IN_OUT
    in_out: INT;
END VAR
--> C0041: VAR_IN_OUT' parameter 'in_out' of 'TEST' needs variable
with write access as input
```

Error correc- Example:
tion: i $:=$ Test $(x)$;

## Compiler Error C0042 (Compiler Version <= 3.4.10)

Message: Either all or none formal parameter have to be denoted in function call
Possible error cause: The parameters are explicitly assigned to the function in the wrong order.

Error correction: Use uniform formal parameters or implicit parameters.

```
Example of the PROGRAM PLC_PRG
error:
VAR
    i : INT;
END_VAR
    i := Test(iPar1:=2, 5);
FUNCTION Test : INT
VAR_INPUT
            iPar1 : INT;
            iPar2 : INT;
END_VAR
--> Either all or none formal parameter have to be denoted in
function call
```


## Compiler error C0043

Message: Wrong formal parameter: '<parameter name>' expected in this place
Possible error cause: The parameters are assigned to the function explicitly in the wrong order.

Error correction: Specify the parameters in the correct order.

```
Example of the PROGRAM PLC_PRG
error:
```

```
VAR
```

VAR
i : INT;
i : INT;
END_VAR
END_VAR
i := Test(iPar2 := 2, 5);
i := Test(iPar2 := 2, 5);
FUNCTION Test : INT
FUNCTION Test : INT
VAR_INPUT
VAR_INPUT
iPar1 : INT;
iPar1 : INT;
iPar2 : INT;
iPar2 : INT;
END_VAR
END_VAR
--> C0043: Wrong formal parameter: 'iPar1' expected in this place
--> C0043: Wrong formal parameter: 'iPar1' expected in this place
--> C0412: Multiple input assignments for parameter ''

```
--> C0412: Multiple input assignments for parameter ''
```

```
Error correc- Example:
tion: i := Test(5, iPar2 := 2);
```


## Compiler error C0044

Message: Assignment to input missing for parameter '<input variable name>' in call of '<function block name>'

Possible error cause: A parameter is passed although an input variable is not declared.
Error correction: Declare an input variable.

```
Example of the PROGRAM PLC_PRG
error:
VAR
        inst : FB;
    END VAR
    inst(1);
FUNCTION BLOCK FB
VAR_INPUT
END_VAR
--> C0044: Assignment to input missing for parameter '1' in call of
'FB'
Error correc- Example:
tion: VAR_INPUT
VAR_INPUT
END_VAR
```


## Compiler error C0045

Message: Use of 'THIS' is not allowed in this context
Possible error cause: In order to be assigned to the current instance, THIS can be used only in a method, action, transition, or in the body of a function block. This error message appears for all other positions.

Error correction: Use THIS in an allowed context only.

```
Example of the PROGRAM PLC_PRG
error: VAR
        test1: INT;
    END VAR
    THIS^.test1 := 19;
    --> C0018: 'THIS^.test1' is not a valid assignment target
    --> C0062: 'THIS^' is not a structure variable
    --> COO45: Use of 'THIS' is not allowed in this context
```


## Compiler error C0046

Message: Identifier '<identifier name>' not defined
Possible error cause: An identifier is used that is not declared.

Error correction: Declare the variables that you want to use.

| Example of the error: | PROGRAM PLC_PRG VAR |
| :---: | :---: |
|  | END_VAR |
|  | i $:=1$; |
|  | --> C0018: 'i' is no valid assignment target <br> --> C0046: Identifier 'i' not defined |
| Error correction: | Example: VAR |
|  | $\begin{aligned} & \text { i : INT; } \\ & \text { END VAR } \end{aligned}$ |

## Compiler error C0047

Message: Cannot apply indexing with '[]' to an expression of type '<data type>'
Possible error cause: A data type that is not an array is indexed with '[]'.
Error correction: Index data types with '[]' only if they are declared as arrays.

```
Example of the PROGRAM PLC_PRG
error: VAR
            i : INT;
    END VAR
    i[1];
    --> C0047: Cannot apply indexing with '[]' to an expression of type
    'INT'
```


## Compiler error C0048

Message: Array requires exactly '<number>' indexes
Possible error cause: Too many or too few indexes are specified when using an array.
Error correction: Specify as many indexes as there are dimensions assigned to the array.

| Example of the error: | PROGRAM PLC_PRG VAR |
| :---: | :---: |
|  | arr1 : ARRAY[1..2,1..3] OF INT; |
|  | END VAR |
|  | arrl$[1] \quad:=5 ;$ |
|  | --> C0048: Array requires exactly 2 indexes |
| Error correc- | Example: |
| tion: | arrl[1,2] := 5; |

## Compiler error C0049

Message: The constant index '<index value>' is not within the range from '<start index>' to '<end index>'

Possible error cause: An index is specified that is outside the size of the array.
Error correction: Use only indexes that are within the size of he array.

```
Example of the PROGRAM PLC_PRG
error:
VAR
    arr1 : ARRAY[1..2] OF INT;
END_VAR
arr\overline{1}[3] := 1;
--> C0049: The constant index '3' is not within the range from '1' to
'2'
Error correc- Example:
tion: arrl[2] := 1;
```


## Compiler error C0050

Message: Bitaccess requires literal or symbolic integer constant
Possible error cause: No literal or an integer constant is specified in a bit access.
Error correction: Use a literal or an integer constant.

```
Example of the PROGRAM PLC_PRG
error: VAR
            i : INT;
            x : INT;
END VAR
i.x := FALSE;
--> C0018: 'i.x' is no valid assignment target
--> C0050: Bitaccess requires literal or symbolic integer constant
Error correc- Example:
tion: i := Test(x);
```


## Compiler Error C0051

Message: Single byte string expected for an attribute value instead of '<value>'.
Possible error cause: A character string does not appear at the displayed location as expected.
Error correction: Replace the current value with a string.

```
Example of the PROGRAM PLC_PRG
error: {IF hasattríbute(pou: MyPOU, MyAttribute)}
    {END_IF}
    --> C0051: Single byte string expected for an attribute value instead
    of 'MyAttribute'.
```

```
Error correc- PROGRAM PLC_PRG
tion: {IF hasattribute(pou: MyPOU, 'MyAttribute')}
{END IF}
```


## Compiler Error C0053

Message: Compiler version <version> has been withdrawn. Please use a higher compiler version instead.

Possible error cause: The current compiler version cannot ne used.
Error correction: Adapt the current compiler version in the project (Project Environment, Project Settings).

## Compiler error C0061

Message: Bitaccess on function call is not allowed
Possible error cause: Bit access is performed on a function.
Error correction: Use bit access only for supported data types.


## Compiler error C0062

Message: '<variable name>' is no structured variable
Possible error cause: A variable that is not a structure variable is treated like a structure variable.

Error correction: Make sure that the variable is a structure variable.

| Example of the error: | ```PROGRAM PLC_PRG VAR pt : PUNKT; i : INT; END VAR``` |
| :---: | :---: |
|  | i.x := 1024; |
|  | TYPE Punkt : STRUCT |
|  | $x$ : REAL; <br> y : REAL; <br> END STRUCT |

```
--> C0018: 'i.x' is no valid assignment target
```

--> C0062: 'Variable' is no structured variable

| Error correc- | Example: |
| :--- | :--- |
| tion: | pt.x $:=1024 ;$ |

## Compiler error C0064

Message: Dereferencing requires a pointer
Possible error cause: A variable that is not a pointer variable is dereferenced.
Error correction: Dereference only variables that are pointer variables.

```
Example of the PROGRAM PLC_PRG
error: VAR
            i : INT;
            pi : POINTER TO INT;
END_VAR
i^:=1;
--> C0018: 'i^' not a valid assignment target
--> C0064: Dereferencing requires a pointer
Error correc- Example:
tion: pi := ADR(i);
pi^ := 1;
```


## Compiler Error C0065

Message: There is no global definition for '<name>'.
Possible error cause: The value searched for is not a global variable, global POU, or other value that can be accessed globally.

Error correction: Declare '<name>' as a global variable.

| Example of the error: | PROGRAM PLC_PRG <br> .someValue := 5; |
| :---: | :---: |
|  | --> C0065: There is no global definition for 'someValue' |
| Error correc- | Example: |
| tion: | VAR_GLOBAL |
|  | someValue : INT; |
|  | END_VAR |

## Compiler error C0066

Message: Cannot compare type '<data type>' with type '<data type>'
Possible error cause: Two data types are compared which cannot be compared with each other.

Error correction: Compare only data types that can be compared with each other.

```
Example of the PROGRAM PLC_PRG
error: VAR
            i : INT;
            re : REAL;
            str: STRING;
            b : BOOL;
END_VAR
b := i > str;
--> C0066: Cannot compare type 'INT' with type 'STRING'
Error correc- Example:
tion: b := i > re;
```


## Compiler error C0068

Message: Compare not possible on objects of type '<data type>'
Possible error cause: Objects are being compared in which a comparison is not possible.
Error correction: Compare only data types in which a comparison is possible (INT, REAL, etc.).

```
Example of the PROGRAM PLC_PRG
error: VAR
            b : BOOL;
            arr1 : ARRAY [1..2] OF INT;
            arr2 : ARRAY [1..2] OF INT;
END VAR
b := arr1 > arr2;
--> C0068: Compare not possible on objects of type 'ARRAY [1..2]'
```


## Compiler error C0069

Message: Compare not possible on objects of type '<data type>' or '<data type>'
Possible error cause: Two different objects are being compared in which a comparison is not possible.
Error correction: Compare only data types in which a comparison is possible (INT, REAL, etc.).

```
Example of the PROGRAM PLC_PRG
error:
VAR
            b : BOOL;
            arr1 : ARRAY [1..2] OF INT;
            arr2 : ARRAY [1..3] OF INT;
    END_VAR
b := arr1 > arr2;
--> C0069: Compare not possible on objects of type 'ARRAY [1..2]' or
'ARRAY [1..3]'
```


## Compiler error C0070

Message: 'INI' operator needs function block instance or data unit type instance
Possible error cause: Neither a function block instance nor a DUT instance is applied to the INI operator.
Error correction: Pass only function block instances or DUT instances to the INI operator.

```
Example of the PROGRAM PLC_PRG
error: VAR
            b : BOOL;
            inst : FB;
    END_VAR
    b := INI(b, TRUE);
    FUNCTION_BLOCK FB
    VAR
END_VAR
    --> C0070: 'INI' operator needs function block instance or data unit
    type instance
Error correc- Example:
tion: b := INI(inst, TRUE);
```


## Compiler error C0072

Message: Operator <operator name>' is not possible on type '<data type>'
Possible error cause: An operator is applied to an incompatible type.
Error correction: Apply operators only on compatible types.


## Compiler error C0074

Message: Unexpected array initialisation
Possible error cause: Syntax error in the array initialization
Error correction: Correct the syntax

```
Example of the PROGRAM PLC_PRG
error: VAR
        arr1 : INT := [1,2,3,4,5,6];
    END VAR
--> C0074: Unexpected array initialisation
--> C0032: Cannot convert type 'Unknown type: [1,2,3,4,5,6]' to type
'INT'
Error correc- Example:
tion: arr1 : ARRAY [1..6] OF INT := [1,2,3,4,5,6];
```


## Compiler error C0075

Message: Too many initializers for array
Possible error cause: Too many values are specified for the size of the array.
Error correction: The number of assigned values must correspond to the size of the array.


## Compiler error C0076

Message:Unexpected structure initialisation
Possible error cause: Syntax error in the structure initialization
Error correction: Make sure that the syntax is correct.

```
Example of the PROGRAM PLC_PRG
error:
VAR
    st1 : INT := (p1 := 1);
END VAR
--> C0076: Unexpected structure initialisation
--> C0032: Cannot convert type 'STRUCT(p1:=1)' to type 'INT'
--> C0046: Identifier 'p1' not defined
--> C0018: 'p1' is no valid assignment target
Error correc- Example:
tion: st1 : STRUCT1 := (p1:=1,p2:=10);
```


## Compiler error C0077

Message: Unknown type: '<data type>'
Possible error cause: Invalid data type in the declaration (maybe a syntax error)
Error correction: Specify valid data types only.

```
Example of the PROGRAM PLC_PRG 
--> C0077: Unknown type: 'INTEGER'
Error correc- Example:
tion: arr1 : ARRAY[1..2] OF STRUCT1 := (p1:=1,p2:=10);
```


## Compiler Error C0078

Message: Unsupported type: '<data type>'
Possible error cause: The used type is not supported by the current device and therefore cannot be used.

Error correction: If possible, use a different type. For example, REAL instead of LREAL.

```
Example of the PROGRAM PLC_PRG
error:
VAR
    value : LREAL;
END_VAR
--> C0078: Unsupported type: 'LREAL'
Error correc- Example:
tion: PROGRAM PLC_PRG
VAR
    value : REAL;
END_VAR
```


## Compiler error C0080

Message: Functionblock '<function block name>' must be instantiated to be accessed Possible error cause: Missing function Block Instantiation

Error correction: Instantiate the function block.


## Compiler error C0081

Message: Unexpected Pragma: '<pragma name>' found without matching 'if' Possible error cause: The IF condition is missing when using the pragma.

Error correction: Complete the IF condition of the pragma.

```
Example of the PROGRAM PLC_PRG
error: VAR
            i : INT;
    END VAR
    i := 5;
    {END_IF}
    --> C0081: Unexpected Pragma: 'END IF' found without matching 'if'
Error correc- Example:
tion: {IF <expression>}
    i := 5;
    {END_IF}
```


## Compiler error C0082

Message: '<invalid pragma>' is no valid condition for pragma
Possible error cause: When using a pragma, an invalid expression is used in the IF condition.
Error correction: Use valid pragma conditions.

```
Example of the PROGRAM PLC_PRG
error: VAR
            i : INT;
    END_VAR
{IF abc}
i := 5;
{END_IF}
--> C0082: '!!!ERROR!!!' is no valid condition for pragma
Error correc- Example:
tion: {IF defined (abc)}
```


## Compiler error C0084

Message: '<pragma operand>' is no valid operand for pragma
Possible error cause: Syntax error
Error correction: Use valid pragma operands.

```
Example of the PROGRAM PLC_PRG
error: VAR
END_VAR
{IF defined(0) }
{END_IF}
--> C0084: 'defined(null)' is no valid operand for pragma
Error correc- Example:
tion: {IF defined (abc)}
```


## Compiler Error C0085

Message: Define value expected instead of '<value>'.
Possible error cause: A string is expected instead of the current value at the displayed location of the pragma.
Error correction: Replace the current value with a string.

Example of the PROGRAM PLC_PRG
error:

```
{IF hasvalue(define, defineValue)}
```

\{END_IF\}
--> C0086: C:0085: Define value expected instead of 'defineValue'
Error correc- Example:
tion: PROGRAM PLC_PRG
\{IF hasvalue(define, '120')\}
\{END_IF \}

## Compiler error C0086

Message: No definition found for interface '<interface name>'
Possible error cause: An undefined interface is used.
Error correction: Define the interface.

```
Example of the PROGRAM PLC_PRG
error: VAR
        inst : FB;
    END_VAR
    FUNCTION_BLOCK FB IMPLEMENTS XY
    VAR
    END_VAR
    --> C0086: No definition found for interface 'XY'
    Error correc- Example:
tion: INTERFACE XY
```


## Compiler error C0087

Message: There is no implementation for method '<method name>' defined in interface '<interface name>'.

Possible error cause: One of the methods specified by the interface has not be provided by the implemented function block.
Error correction: Implement all methods that are specified by the interface.

## Example of the <br> error:

```
PROGRAM PLC_PRG
VAR
            inst : FB;
END_VAR
INTERFACE XY
METHOD METH1
VAR INPUT
END - VAR
FUNCTION_BLOCK FB IMPLEMENTS XY
VAR
END_VAR
MET\overline{H}OD METH2
VAR_INPUT
END__VAR
--> C0087: There is no implementation for method 'METH1' defined in
interface 'XY'
```


## Compiler error C0089

Message: Interface of overridden method ' $\{0\}$ ' of interface ' $\{1\}$ ' does not match declaration
Possible error cause: The signature of the implemented method does not match the signature of the method in the interface.

Error correction: Make sure that the same return types and parameters are declared.

## Example of the error:

```
PROGRAM PLC_PRG
VAR
            inst : FB;
END_VAR
INTERFACE XY
METHOD METH1
VAR INPUT
            iPar : INT;
END_VAR
FUNCTION_BLOCK FB IMPLEMENTS XY
VAR
END VAR
MET\overline{HOD METH1}
VAR_INPUT
```

```
END_VAR
--> C0089: Interface of overridden method 'METH1' of interface 'XY'
does not match declaration
```


## Compiler error C0090

Message: No definition found for base class '<function name>'
Possible error cause: The function block specified as the base does not exist or is not a function block.

Error correction: Use a function block as the base.

```
Example of the PROGRAM PLC_PRG
error:
VAR
            inst : FB;
    END_VAR
    FUNCTION_BLOCK FB EXTENDS POU
    VAR
    END VAR
    FUNCTION POU
    VAR
    END_VAR
    --> C00090: No definition found for base class 'POU'
```


## Compiler error C0091

Message: Recursion in base function block list: <function name>
Possible error cause: A base function block is extended by itself.
Error correction: Recursion in base function block lists is not possible.

```
Example of the PROGRAM PLC_PRG
error:
VAR
        inst : FB;
    END_VAR
```

    FUNCTION_BLOCK FB EXTENDS FB
    VAR
    END_VAR
    --> C00091: Recursion in base function block list: FB -> FB
    
## Compiler error C0094

Message: Interface of overridden method '<method name>' of interface '<function block name>' doesn't match declaration

Possible error cause: The signature of the method of the first interface does not match the signature of the method in the second interface, which is extended by the first.

Error correction: Align the signatures.

```
Example of the PROGRAM PLC_PRG
error:
VAR
        Xyz : XY2;
END_VAR
FUNCTION_BLOCK XY
METHOD METH1
VAR_INPUT
END_VAR
FUNCTION BLOCK XY2 EXTENDS XY
METHOD METH1
VAR INPUT
    iPar : BOOL;
END_VAR
--> C00094: Interface of the overridden method METH1 of interface XY
doesn't match declaration
```


## Compiler error C0096

Message: Only one base function block may be defined in EXTENDS-list.
Possible error cause: Two or more base function blocks are defined in the EXTENDS list.
Error correction: Define only one base function block in the EXTENDS list

```
Example of the PROGRAM PLC_PRG
error:
VAR
    fb : FB;
END_VAR
FUNCTION BLOCK FB EXTENDS FB2, FB3
VAR
END_VAR
FUNCTION BLOCK FB2
VAR
END VAR
FUNCTION_BLOCK FB3
VAR
END_VAR
--> C00096: Only one base function block may be defined in EXTENDS-
list
```


## Compiler error C0097

Message: Duplicate definition of variable '<variable name>' in function block '<function block name>' and in base '<base function block name>'

Possible error cause: A variable is declared with the same name in a function block and its base.

Error correction: Use different variable names.

```
Example of the PROGRAM PLC_PRG
error: VAR
|AR fb : FB;
END_VAR
FUNCTION_BLOCK FB EXTENDS FB2
VAR
    i : INT;
END_VAR
FUNCTION_BLOCK FB2
VAR
    i : INT;
END_VAR
--> C00097: Duplicate definition of variable 'i' in function block
'FB' and in base 'FB2'
```


## Compiler error C0098

Message: The keyword "FUNCTIONBLOCK" is no longer supported. Use "FUNCTION_BLOCK" instead.
Possible cause of error: Syntax error
Error correction: Use the keyword "FUNCTION_BLOCK".

```
Example of the PROGRAM PLC_PRG
error:
VAR
            inst : FB;
END_VAR
FUNCTIONBLOCK FB
VAR
END_VAR
--> C00098: The keyword "FUNCTIONBLOCK" is no longer supported. Use
"FUNCTION_BLOCK" instead. Use "FUNCTION_BLOCK" instead.
```


## Compiler Error C0099 (Compiler Version < 3.5.7.0)

Message: Local defined enumeration are no longer supported. Use datatype definition instead.
Possible error cause: A local enumeration declaration was used together with a compiler version that does not support this.

Error correction: Use a later compiler version, or define the enumeration in a DUT.

```
Example of the PROGRAM PLC_PRG
error: VAR
    localEnumVar : (RED, GREEN, BLUE) := RED;
END_VAR
--> C0099: Local defined enumeration are no longer supported. Use
datatype definition instead.
```


## Compiler error C0101

Message: Data Recursion: '<recursion>'
Possible error cause: Recursive data initialization over two function blocks
Error correction: Avoid recursions for data initialization.

```
Example of the PROGRAM PLC_PRG
error: VAR
        inst0 : FB1;
    END VAR
    FUNCTION_BLOCK FB1
VAR
        inst1 : FB2;
END_VAR
    FUNCTION_BLOCK FB2
VAR
        inst2 : FB1;
    END_VAR
    --> C0101: Data Recursion: FB1->FB2->FB1
```


## Compiler error C0102

Message: Out of retain memory: Variable '<variable name>', <byte size> bytes.
Possible error cause: More retain memory is used than is available on the PLC. It is also possible that the retain memory is too fragmented due to incremental builds.

Error correction: Use the "Clean" for fragmenting the memory. This will force the reallocation of all data at the next build.

## Compiler error C0104

Message: 'Out of global data memory: Variable '<variable name>', <byte size> bytes.
Possible error cause: More memory for data is used than is available on the PLC. It is also possible that the memory is too fragmented due to incremental builds.

Error correction: Use the "Clean" for fragmenting the memory. This will force the reallocation of all data at the next build.

## Compiler error C0114

Message: Invalid destination <jump label> for 'JMP'
Possible error cause: Syntax error or typographical error in the JMP destination
Error correction: Correct the typographical or syntax error.

| Example of the | PROGRAM PLC_PRG |
| :--- | :--- |
| error: | VAR |
|  | END_VAR |
|  | JMP 0; |

## Compiler Error C0115

Message: The second parameter of a conditional call (????ALWAYS CALC???? ) has to be a valid call statement.

Error correction: Specify the call of a function, method, or function block in the second parameter of the conditional ????CALC??? call.

```
Example of the PROGRAM PLC_PRG
error: VAR
        condition : BOOL;
    END VAR
CAL\overline{C}(condition, 1+2);
--> C0115: The second parameter of a conditional call has to be a
valid call statement.
Error correc- Example:
tion: CALC(condition, MyFunction(1,2))
```


## Compiler error C0116

Message: The label '<jump label>' is a duplicate
Possible error cause: A label is defined multiple times.
Error correction: Define each label one time only.


## Compiler error C0117

Message: No such label '<jump label>' within the scope of the 'JMP' statement
Possible error cause: A jump is made to a label that does not exist.
Error correction: Define the label that you specify as the destination.

```
Example of the PROGRAM PLC_PRG
error:
VAR
END_VAR
JMP A;
--> C0117: No such label 'A' within the scope of the 'JMP' statement
```


## Compiler error C0118

Message: The label '<jump label>' has not been referenced.

Possible error cause: A jump label is defined that is not referenced.
Error correction: Remove the unused jump labels.

```
\begin{tabular}{ll} 
Example of the & PROGRAM PLC_PRG \\
error: & VAR \\
& END VAR \\
& LABEL:
\end{tabular}
--> C0118: The label ' LABEL' has not been referenced
```


## Compiler error C0119

Message: An 'FB_init'-Method of a functionblock or struct needs two inputs 'blnitRetains' and 'bInCopyCode' of type BOOL
Possible error cause: One or both of the inputs 'blnitRetains' and 'bInCopyCode' of type BOOL is missing.
Error correction: Define the missing inputs.

```
Example of the PROGRAM PLC_PRG
error: VAR
        inst : FB;
END_VAR
FUNCTION BLOCK FB
METHOD F\overline{B}}\mathrm{ init
VAR_INPUT
END_VAR
--> C0119: An 'FB_init'-Method of a functionblock or struct needs two
inputs 'bInitRetains' and 'bInCopyCode' of type BOOL
Error correc- Example:
tion: METHOD FB init
VAR INPUT
    bInitRetains : BOOL;
    bInCopyCode : BOOL;
END_VAR
```


## Compiler error C0120

Message: An 'FB_Exit'-Method of a functionblock or struct needs an input 'blnCopyCode' of type BOOL.
Possible error cause: The input 'bInCopyCode' of type BOOL is missing.
Error correction: Define the input.

| Example of the | PROGRAM PLC_PRG |
| :--- | :--- |
| error: | VAR $\quad$ inst : FB; |
|  | END_VAR |
|  |  |
|  | FUNCTION_BLOCK FB |

```
METHOD FB_exit
VAR_INPUT
END_VAR
--> C0120: An 'FB Exit'-Method of a functionblock or struct needs an
input 'bInCopyCode' of type BOOL.
Error correc- Example:
tion: METHOD FB exit
VAR_INPUT
    bInCopyCode : BOOL;
END_VAR
```


## Compiler error C0122

Message: Expression 'SUPER' is not allowed in this context
Possible error cause: "SUPER^" is used outside of derived function blocks.
Error correction: Use "SUPER^" in function blocks only.

```
Example of the PROGRAM PLC_PRG
error: VAR
    END_VAR
SUPER^.METH(TRUE, TRUE);
--> C0122: Expression 'SUPER' is not allowed in this context
```


## Compiler error C0124

Message: 'Initialization' is no valid initialization for an enumeration
Possible error cause: A data type that is not ANY_INT is used for the enum initialization.
Error correction: Use only ANY_INT for enum initializations.

```
Example of the PROGRAM PLC_PRG
error:
VAR
        inst : DUT;
END_VAR
TYPE DUT :
(
    enum member := 1.5
) DWORD;
END_TYPE
--> C0032: Cannot convert type 'LREAL' to type 'DUT'
--> C0124: 'Initialization' is no valid initialization for an
enumeration
```


## Compiler error C0125

Message: The constant <constant value> is assigned to more than one enumeration. Possible error cause: The same value is assigned to two or more enumerations.

Error correction: Assign different values to the enumerations.

```
Example of the PROGRAM PLC_PRG
error:
VAR
        inst : DUT;
END VAR
TYPE DUT :
(
    enum_member := 0,
        enum_member2 := 0
);
END_TYPE
--> C0125: The constant 0 is assigned to more than one enumeration
```


## Compiler error C0126

Message: Variable of type '<data type>' requires exactly 1 Index
Possible error cause: Multiple indexes are assigned to a variable with one index.
Error correction: Assign only one index.

```
\begin{tabular}{ll} 
Example of the & PROGRAM PLC_PRG \\
error: & VAR \\
& pi \(: ~ P O I N T E R ~ T O ~ I N T ; ~\) \\
& END_VAR \\
& pi \([0,1]:=0 ;\)
\end{tabular}
--> C0126: Variable of type 'POINTER TO INT' requires exactly 1 Index
Error correc- Example:
tion: pi[0] := 0;
```


## Compiler error C0130

Message: <object> '<object name>' referenced without parentheses '()'
Possible error cause: A method is referenced without parentheses.
Error correction: Always reference methods by means of parentheses.


```
Compiler error C0131
    Message: '<value>' is not allowed as operand for 'ADR'
    Possible error cause: A constant is passed as an operand to the operator ADR.
    Error correction: Use only valid operands for ADR.
Example of the PROGRAM PLC_PRG
error: VAR
            i : INT;
            pt : POINTER TO INT;
    END_VAR
    pt := ADR(1);
    --> C0131: '1' is not allowed as operand for 'ADR'
Error correc- Example:
tion: pt := ADR(i);
```


## Compiler error C0132

Message: No enclosing loop of which to EXIT
Possible error cause: EXIT is used outside of a loop.
Error correction: Use EXIT inside of a loop only.

```
Example of the PROGRAM PLC_PRG
error:
VAR
    END_VAR
    EXIT
    ;
    --> C0132: No enclosing loop of which to EXIT
```


## Compiler error C0136

Message: ambiguous use of name '<variable name>'
Possible error cause: A variable is declared in multiple GVLs.
Error correction: Qualify the variable with the desired GVL.

Example of the PROGRAM PLC_PRG
error:
VAR
j : INT $:=g_{-}{ }^{i} ; ~$
GVL1:
VAR_GLOBAL
g_i : INT;
END_VAR
GVL2 :

```
VAR_GLOBAL
            g_i : INT;
END VAR
--> C0136: ambiguous use of name 'g_i'
```

Error correc- Example:
tion: $\quad j:$ INT $:=$ GVL1.g_i;

## Compiler Error C0138

Message: No matching 'FB_Init' method found for instantiation of POU.
Possible error cause: No FB_Init method exists that accepts the passed parameters.
Error correction: Check which arguments FB_Init has to receive and adjust the passed arguments

```
Example of the PROGRAM PLC_PRG
error: VAR
    myPOU : POU(arg1 := 1, arg2 := 2);
    END VAR
    --> C0138: No matching 'FB Init' method found for instantiation of
    POU .
```


## Compiler error C0139

Message: The code <code> has no effect. Is this the intent?
Possible error cause: The written code is syntactically correct but does not do anything.
Error correction: Write code that has a purpose.

```
Example of the PROGRAM PLC_PRG
error: VAR
            i : INT;
END VAR
i;
--> C0139: The code 'i;' has no effect. Is this the intent?
```


## Compiler error C0140

Message: Reference assign is only allowed to variables of Reference type
Possible error cause: An attempt is made to assign a reference value to a variable not defined as a reference type.

Error correction: Define the variable as a reference type.

| Example of the error: | PROGRAM <br> VAR |
| :---: | :---: |
|  | i : |
|  | I_r |
|  | END_VAR |

```
I_r REF= i;
--> C0140: Reference assign is only allowed to variables of Reference
type
Error correc- Example:
tion: I_r : REFERENCE TO INT;
```


## Compiler error C0141

Message: Reference assign needs variable with write access
Possible error cause: A constant is assigned to the reference assignment.
Error correction: Assign a writable variable.

```
Example of the PROGRAM PLC_PRG
error: VAR
            i : INT;
            I_r : REFERENCE TO INT;
END_V\overline{A}R
I_r REF= 314;
--> C0141: Reference assign needs variable with write access
Error correc- Example:
tion: I_r REF= i;
```


## Compiler error C0142

Message: A local variable named '<variable name>' is already defined in '<pou name>'
Possible error cause: The same variable name is used two times.
Error correction: Use different variable names.

```
Example of the PROGRAM PLC_PRG
error: VAR
            i : INT;
            i : INT;
END_VAR
--> C0142: A local variable named 'i' is already defined in 'PLC_PRG'
```


## Compiler error C0143

Message: The property '<property name>' cannot be used in this context because it lacks the get accessor
Possible error cause: The property does not have Get access.
Error correction: Make sure that the property has a Get access definition.

```
Example of the PROGRAM PLC_PRG
error: VAR
            i : INT;
            inst: FB;
END_VAR
i := inst.Prop;
FUNCTION_BLOCK FB
VAR
END_VAR
PROPERTY Prop : INT
Set;
--> C0143: The property 'Prop' cannot be used in this context because
it lacks the get accessor
```


## Compiler error C0144

Message: Inheritance only allowed in Functionblocks, Interfaces and Structures
Possible error cause: An attempt is made to use inheritance in an object that does not permit inheritance

Error correction: Use EXTENDS in function blocks, interfaces, and structures only.

```
Example of the PROGRAM PLC_PRG
error:
VAR
        inst : DUT_1;
END_VAR
TYPE DUT:
(
    enum member := 0
) ;
END_TYPE
TYPE DUT_1 EXTENDS DUT:
(
    enum_memberX := 0
);
END_TYPE
--> C0144: Inheritance only allowed in Functionblocks, Interfaces and
Structures
```


## Compiler error C0145

Message: Interfaces can only be implemented by Functionblocks
Possible error cause: An attempt is made to implement an interface outside of a function block.

Error correction: Implement interfaces only in function blocks.

```
Example of the PROGRAM PLC_PRG
error: VAR
END_VAR
POU();
FUNCTION POU IMPLEMENTS ITF
VAR
END_VAR
--> C0145: Interfaces can only be implemented by Functionblocks
```


## Compiler error C0149

Message: Variable declarations are not allowed in interfaces
Possible error cause: An attempt is made to define a variable in an interface.
Error correction: Do not define variables in interfaces.

```
Example of the PROGRAM PLC_PRG
error: VAR
            inst : ITF;
END_VAR
INTERFACE ITF
VAR_INPUT
    i : INT;
END VAR
--> C0149: Variable declarations are not allowed in interfaces
```


## Compiler error C0161

Message: Border <array bound> of array is no constant value Possible error cause: A variable is specified as an array bound.
Error correction: Use constants for the array bounds.

```
Example of the PROGRAM PLC_PRG
error: VAR
            i : INT := 3;
            arr1 : ARRAY[1..i] OF INT;
END_VAR
--> C0161: Border 'i' of array is no constant value
Error correc- Example:
tion: arr1 : ARRAY[1..3] OF INT;
```


## Compiler error C0162

Message: Number <number of array values> of array initialisations is no constant value Possible error cause: The initialization [Wert1, Anzah1Wert2 (Wert2)] works only with a constant for AnzahlWert2.

Error correction: Use constants only.

```
Example of the PROGRAM PLC_PRG
error: VAR
i : INT := 3;
    arr1 : ARRAY[1..4] OF INT := [1,i(7)];
END_VAR
--> C0162: Number 'i' of array initialisations is no constant value
Error correc- Example:
tion: arr1 : ARRAY[1..4] OF INT := [1,3(7)];
```


## Compiler Error C0164

Message: POU <name> writes to output <name> and is called in several tasks.
Possible error cause: The device setting codegenerator \check-multiple-task-output-write is set and multiple tasks access the same output.
Error correction: Do not call a program that changes outputs in multiple tasks.

## Example of the error:

PROGRAM PLC_PRG
VAR
Output AT \%QB7 : BYTE
END_VAR

Output $:=0$;
--> C0164: POU 'PLC_PRG' writes to output 'QB7' and is called in several tasks

## Compiler Error C0165

Message: Variable '<variable name>', which is mapped on address '<address>' is written in different tasks.

Possible error cause: The device setting codegenerator\check-multiple-task-output-write is set and multiple tasks access the same output.
Error correction: Write an output in one fixed task only. If multiple tasks need to calculate data for one output, then you should try to transfer this information by means of global variables to one fixed task, which then writes the data to one output.

```
                                    PLC_PRG_1 (PRG)
                                    PLC_PRG_2 (PRG)
G Taskkonfiguration
    -* MainTask (IEC-Tasks)
        田 PLC_PRG_2
        Task (IEC-Tasks)
        田 PLC_PRG_1
```

```
Example of the PROGRAM PLC_PRG_1
```

Example of the PROGRAM PLC_PRG_1
error:
error:
VAR
VAR
Output AT %QB7 : BYTE;
Output AT %QB7 : BYTE;
END VAR
END VAR
Output := 0;
Output := 0;
PROGRAM PLC_PRG_2
PROGRAM PLC_PRG_2
VAR
VAR
Output AT %QB7 : BYTE;
Output AT %QB7 : BYTE;
END VAR
END VAR
Output := 1;
Output := 1;
--> C0165: Variable 'Output', which is mapped on address 'QB7' is
--> C0165: Variable 'Output', which is mapped on address 'QB7' is
written in different tasks.

```
written in different tasks.
```


## Compiler error C0168

Message: 'VAR_CONFIG' declaration only allowed in VAR_CONFIG - list Possible error cause: 'VAR_CONFIG' is used outside of a VAR_CONFIG list.
Error correction: Use 'VAR_CONFIG' only in VAR_CONFIG lists.

```
Example of the PROGRAM PLC_PRG
error:
VAR_CONFIG
        i : INT;
END_VAR
--> C0168: 'VAR_CONFIG' declaration only allowed in VAR_CONFIG - list
```


## Compiler error C0169

Message: 'VAR_GLOBAL' declaration only allowed in Global variable list
Possible error cause: 'VAR_GLOBAL' is used outside of global variable lists.
Error correction: Use 'VAR_GLOBAL' in global variable lists only.

```
Example of the PROGRAM PLC_PRG
error:
VAR_GLOBAL
END_VAR
--> C0169: 'VAR_GLOBAL' declaration only allowed in Global variable
list
```


## Compiler Error C0173

Message: '<keyword>' not allowed in this place
Possible error cause: A declaration keyword (example: VAR_INPUT, VAR_OUTPUT, or VAR) is not allowed at this location.

Error correction: Correct the declaration: Inputs and outputs are not useful or necessary in type definitions or global variable lists.

```
TYPE DUT :
STRUCT
    member : INT;
    END_STRUCT
END_TYPE
```

| Example of the error: | TYPE DUT : STRUCT |
| :---: | :---: |
|  | VAR_INPUT |
|  | END VAR |
|  | END_STRŪCT |
|  | END_TYPE |

    --> C0173: 'VAR_INPUT' not allowed in this place.
    Error correction Example:
TYPE DUT :
STRUCT
member : INT;
END STRUCT
END_TYPE

## Compiler error C0174

Message: 'VAR_TEMP' declaration not allowed in this place
Possible error cause: 'VAR_TEMP' is used outside of a program or function block.
Error correction: Use 'VAR_TEMP' inside of programs and function blocks only.

```
Example of the PROGRAM PLC_PRG
error: VAR
    END_VAR
    FUN();
    FUNCTION FUN
    VAR_TEMP
    END_VAR
    --> C0174: 'VAR_TEMP' declaration not allowed in this place
```


## Compiler error C0175

Message: 'RETAIN' or 'PERSISTENT' not allowed in this place
Possible error cause: 'RETAIN' or 'PERSISTENT' is used in a function.
Error correction: Use 'RETAIN' or 'PERSISTENT' at the intended locations.

```
Example of the PROGRAM PLC_PRG
error: VAR
    END_VAR
    POU_1();
    FUNCTION POU_1
    VAR RETAIN
    END_VAR
    --> C0175: 'RETAIN' or 'PERSISTENT' not allowed in this place
See also
- \(\boldsymbol{y y}^{*}\) Chapter 6.4.1.20.2.14 "Retain Variable - RETAIN" on page 2124
- \(\boldsymbol{y y}^{4}\) Chapter 6.4.1.20.2.14 "Retain Variable - RETAIN" on page 2124
```


## Compiler error C0177

Message: '<object>' is of type 'type' and cannot be instantiated
Possible error cause: An attempt is made to instantiate a function.
Error correction: Instantiate only objects that can be instantiated.

```
Example of the PROGRAM PLC_PRG
error: VAR
            inst : POU;
    END_VAR
    FUNCTION POU
    VAR
    END_VAR
    --> C0177: 'POU' is of type 'FUNCTION' and cannot be instantiated
```


## Compiler error C0178

Message: No external access to 'VAR_IN_OUT' parameter '<parameter name>' of '<object name>'
Possible error cause: An attempt is made to remotely access a 'VAR_IN_OUT' parameter.
Error correction: Do not remotely access 'VAR_IN_OUT' parameters.

```
Example of the PROGRAM PLC_PRG
error: VAR
            inst : FB;
            i : INT;
END_VAR
i := inst.in_out;
FUNCTION_BLOCK FB
VAR_IN_OUT
            in_out : INT;
END_VAR
--> C0178: No external access to 'VAR_IN_OUT' parameter 'in_out' of
'FB'
```


## Compiler Error C0179

Message: '<identifier>' is no output of 'Function block'
Possible error cause: The initialization of a function block instance must not contain VAR_IN_OUT variables.

Error correction: Use VAR_IN_OUT variables in function block calls only. When initializing a function block instance, only assign the inputs of a function block.

```
Example of the Example:
error: FUNCTION_BLOCK MyFB
VAR_IN_OUT
    inOut : INT;
END VAR
PROGRAM PLC_PRG
VAR
        iValue : INT;
        fb : MyFB := (inOut := iValue);
END_VAR
--> C0179: 'inOut' is no output of 'MyFB'
```


## Compiler Error C0180

Message: Ambiguous namespace '<library 1>' defined by library '<library 2>'
Possible error cause: The namespace of the library <library $1>$ is not unique. It is already used for <library 2>.
Error correction: Change the namespace of the library accordingly ("Properties" button in the Library Manager).

| Example of the error: | IoStandard = IoStandard, 3.5.15.0 (System) | Standard | 3.5.15.0 |
| :---: | :---: | :---: | :---: |
|  | $\square$ Standard $=$ Standard, 3.5.15.0 (System) | Standard | 3.5.15.0 |

## Compiler error C0182

Message: Return type is only possible for POUs of Type FUNCTION and METHOD Possible error cause: An attempt is made to define a return value in a program.
Error correction: Define a return value only in methods and functions.

```
Example of the PROGRAM PLC_PRG: BOOL
error: VAR
END_VAR
--> C0182: Return type is only possible for POUs of Type FUNCTION and
METHOD
```


## Compiler Error C0183

Message: Global scope operation '.' is not valid on expression '<expression>'
Possible error cause: The '.' operator is used to access a global variable. However, at this location it is not followed by a valid IEC identifier, but for example a character such as ";" or a reserved identifier such as FUNCTION, or an operator such as TO_STRING.
Error correction: Use a valid IEC identifier for a global variable.

```
Example of the PROGRAM PLC_PRG
error:
    iVar := .FUNCTION;// ERROR: C0183 because ; is not a valid identifier
    strVar := .TO STRING;
    --> C0183: Global scope operation '.' is not valid on expression
    '<expression>'
Error correction Example: globalValue is declared in a GVL.
    PROGRAM PLC_PRG
    iVar := .globalValue;
```


## Compiler error C0185

Message: It is not possible to perform component access '.', index access '[]' or call '()' on result of function call. Assign result to help variable first.
Possible error cause: Component or index access to the result of a function call is performed.
Error correction: Assign the result to a variable in order to access.

| Example of the | PROGRAM PLC_PRG |
| :--- | :--- |
| error: | VAR it : ITF; |
|  | END_VAR |
|  | POU_1()[0].METH1(); |

FUNCTION POU_1 : ARRAY[0..0] OF ITF
INTERFACE ITF
METHOD METH1
--> C0185: It is not possible to perform component access '.', index access '[]' or call '()' on result of function call. Assign result to help variable first.

## Compiler Error C0186

Message: It is not possible to compare interface that is return value of call. Assign to variable first.

Possible error cause: A comparison operation is applied to an interface that is returned by a function.

Error correction: First assign the result of the function call to a variable and then compare the value of the variable. This will also reduce the number of function calls that are required.

```
Example of the INTERFACE MyInterface
error:
FUNCTION GetInterface : MyInterface
PROGRAM PLC_PRG
IF GetInterface() <> 0 THEN
    // ...
END_IF
--> C0186: It is not possible to compare interface that is return
value of call. Assign to variable first.
Error correc- Example:
tion: PROGRAM PLC_PRG
VAR_TEMP
    tempInterface : MyInterface;
END VAR
tempInterface := GetInterface();
IF tempInterface <> O THEN
    // ...
END_IF
```


## Compiler Error C0188

Message: Device not installed to the system. No code generation possible.
Possible error cause: The desired device is not installed.


Error correction: Install the missing device in the device repository, or replace the existing device already inserted in the device tree with another existing device ("Update Device").

## Compiler error C0189

Message: ';' expected instead of '<token>'
Possible error cause: Syntax error
Error correction: Make sure that the syntax is correct.

```
Example of the PROGRAM PLC_PRG
error: VAR
INT
END_VAR
--> C0009: Unexpected Token '<Token>' found
--> C0189: ';' expected instead of 'INT'
```


## Compiler error C0190

Message: ';' expected instead of end of POU
Possible error cause: Syntax error in the POU
Error correction: Make sure that the syntax is correct.

```
Example of the PROGRAM PLC_PRG
error:
VAR
    i : INT;
END VAR
i := 5
_-> C0190: ';' expected instead of end of POU
```


## Compiler error C0191

Message: The operator 'INDEXOF' is no longer supported. Use ADR instead. ADR on a POUName returns a Pointer to a Pointer to the function code.
Possible error cause: The outdated operator 'INDEXOF' is used.
Error correction: Use the operator 'ADR'.

## Compiler error C0195

Message: Implicit conversion from signed Type '<data type 1>' to unsigned Type '<data type $2>$ ' : possible change of sign

Possible error cause: A sign conflict may have been missed in the implicit conversion.
Error correction: Convert only data types with the same sign implicitly.

```
Example of the PROGRAM PLC_PRG
error:
VAR
    i : INT;
    b : UINT;
END VAR
b := i;
--> C0195: Implicit conversion from signed Type 'INT' to unsigned
Type 'UINT' : possible change of sign
```


## Compiler error C0196

Message: Implicit conversion from unsigned Type '<data type 1>' to signed type '<data type $2>$ ' : possible change of sign

Possible error cause: A sign conflict may have been missed in the implicit conversion.
Error correction: Use explicit conversions.

```
Example of the PROGRAM PLC_PRG
error: VAR
    i : INT;
    b : UINT;
END_VAR
i := b;
--> C0196: Implicit conversion from unsigned Type 'UINT' to signed
type 'INT' : possible change of sign
```


## Compiler error C0197

Message: Implicit conversion from '<data type 1>' to '<data type 2>': possible loss of information

Possible error cause: An attempt is made to convert a variable from data type DINT or LINT to data type REAL.
Error correction: For DINT, use the data type LREAL, and when converting from LINT to LREAL make sure that the value of LINT does not exceed the capacity of LREAL.

```
Example of the PROGRAM PLC_PRG
error: VAR
            i : DINT;
            b : REAL;
END_VAR
b := i;
--> C0197: Implicit conversion from 'DINT' to 'REAL': possible loss
of information
```


## Compiler error C0198

Message: String constant '<string value>' too long for destination type '<data type>'
Possible error cause: The string constant has too many characters.
Error correction: Use shorter string constants or declare larger strings.

```
Example of the PROGRAM PLC_PRG
error:
VAR
    str : STRING(4) := '12345';
END_VAR
--> C0198: String constant '12345' too long for destination type
'STRING(4)
```


## Compiler error C0199

Message: Interface '<interface name>' must be instantiated to be accessed
Possible error cause: An attempt is made to access an interface method without the interface being instantiated.

Error correction: Instantiate the interface.


## Compiler error C0201

Message: Type '<data type 1>' is not equal to type '<data type 2>' of VAR_IN_OUT 'Variable'
Possible error cause: The data type that is passed to the function as a VAR_IN_OUT parameter does not match the data type defined in it.

Error correction: Pass a variable with the correct data type.

```
Example of the PROGRAM PLC_PRG
error:
VAR
        Inst: POU;
        b : BOOL;
    END VAR
inst(in_out := b);
FUNCTION BLOCK POU
VAR_IN_OUT
        in-out : INT;
END_VAR
--> C0201: Type 'BOOL' is not equal to type 'INT' of VAR_IN_OUT
'Variable'
```


## Compiler error C0203

Message: Only Structures and Function Blocks can contain variables of type BIT.
Possible error cause: An attempt is made to declare a variable of type BIT outside of structures and function blocks.

Error correction: Declare variables of type BIT only in structures and function blocks.

```
Example of the PROGRAM PLC_PRG
error:
```

```
VAR
```

VAR
b : BIT;
b : BIT;
END_VAR
END_VAR
--> C0203: Only Structures and Function Blocks can contain variables
--> C0203: Only Structures and Function Blocks can contain variables
of type BIT.

```
of type BIT.
```


## Compiler error C0204

Message: Variables of type BIT must be declared within a VAR_INPUT-, VAR_OUTPUT or VAR-block

Possible error cause: An attempt is made to define a variable of type BIT or as a VAR_IN_OUT parameter.
Error correction: Define variables of type BIT only within a VAR_INPUT, VAR_OUTPUT or VAR block.

```
Example of the PROGRAM PLC_PRG
error: VAR
            inst : FB;
    END_VAR
FUNCTION BLOCK FB
VAR_IN_OŪT
END VAR
--> C0204: Variables of type BIT must be declared within a
VAR_INPUT-, VAR_OUTPUT or VAR-block
```


## Compiler error C0205

Message: POINTER TO BIT is not allowed
Possible error cause: An attempt is made to declare a POINTER TO BIT.
Error correction: Do not declare POINTER TO BIT.

```
Example of the PROGRAM PLC_PRG
error:
VAR
    pt : POINTER TO BIT;
    END VAR
    --> C0205: POINTER TO BIT is not allowed
```


## Compiler error C0206

Message: BIT is not allowed as base type of an array
Possible error cause: An attempt is made to declare a BIT array.
Error correction: Do not declare BIT arrays.

```
Example of the PROGRAM PLC_PRG
error: VAR
            arr : ARRAY[1..2] OF BIT;
    END_VAR
    --> C0206: BIT is not allowed as base type of an array
```


## Compiler Error C0207

Message: There is no system definition for '<identifier>'

Possible error cause: An attempt was made to access a variable in $\qquad$ SYSTEM that does not exist.

Error correction: Check and correct the specified identifier of the respective variable.

```
Example of the PROGRAM PLC_PRG
error:
VAR
            Value : INT;
END_VAR
Value := __SYSTEM.UnknownVariable;
--> C0207: There is no system definition for 'UnkownVariable'
```


## Compiler error C0208

Message: 'MOD' is not defined for 'REAL'
Possible error cause: An attempt is made to perform a modulo operation with a variable of type REAL.

Error correction: Modulo operations are only possible with variables of type ANY_INT.

```
Example of the PROGRAM PLC_PRG
error:
VAR
    r1 : REAL;
END_VAR
r1 := r1 MOD 2;
--> C0208: 'MOD' is not defined for 'REAL'
```


## Compiler Error C0209

Message: You have defined '<number>' applications for device '<device name>'. The maximum number is '<number>'. So you will not be able to download all applications.

Possible error cause: Some devices only support a specific number of applications (device description). If a project contains more applications, then not all will be downloaded to the device.

Error correction: Remove applications from your project or use another device.

## Compiler error C0211

Message: Variable declaration expected instead of <expression>
Possible error cause: Syntax error
Error correction: Make sure that the syntax is correct.

```
Example of the PROGRAM PLC_PRG
error: VAR
    VAR
    END_VAR
END VAR
--> C0211: Variable declaration expected instead of VAR END_VAR
```


## Compiler error C0212

Message: VAR, VAR_INPUT, VAR_OUTPUT or VAR_INOUT expected instead of <expression> Possible error cause: Syntax error

Error correction: Make sure that the syntax is correct.

```
Example of the PROGRAM PLC_PRG
error: i : INT;
--> C0212: VAR, VAR_INPUT, VAR_OUTPUT or VAR_INOUT expected instead
of i : INT;
```


## Compiler Error C0215

Message: Direct address declaration is not possible in persistent list
Possible error cause: Persistent variables are not allowed to have a direct address.
Error correction: Remove the direct address assignment in the persistent variable list.

```
Example of the VAR_GLOBAL PERSISTENT RETAIN
error: -directAddressVar AT %QB7 : BYTE;
END_VAR
--> C0215: Direct address declaration is not possible in persistent
list.
```


## Compiler error C0216

Message: Case label duplicate
Possible error cause: A CASE label is used multiple times.
Error correction: Use each CASE label only one time.

```
Example of the PROGRAM PLC_PRG
error: VAR
            i : INT;
    END VAR
    CASE i OF
        1: i := i+1;
        1: i := i+2;
    ELSE
        i := i+10;
    END_CASE;
    --> C0216: Case label duplicate
```


## Compiler error C0217

Message: Case label <case label> also contained in range <case range begin> .. <case range end>
Possible error cause: A CASE label is part of the range of another CASE label.
Error correction: Make sure that there is no intersecting.

```
Example of the PROGRAM PLC_PRG
error: VAR
            i : INT;
END VAR
CASE i OF
    3..5: i := i+2;
    4: i := i+2;
ELSE
    i := i+10;
END_CASE;
--> C0217: Case label 4 also contained in range 3 .. 5
```


## Compiler error C0218

Message: Case label requires literal or symbolic integer constant
Possible error cause: An attempt is made to use a variable as a CASE label.
Error correction: Use only literals and symbolic integer constants.

```
Example of the PROGRAM PLC_PRG
error:
VAR
            i : INT;
            a : INT := 2;
END_VAR
CASE i OF
    1: i := i+1;
    a: i := i+2;
ELSE
    i := i+10;
END_CASE;
--> C0218: Case label requires literal or symbolic integer constant
```


## Compiler error C0219

Message: Case contains overlapping range <case range 1 begin> .. <case range 1 end> and <case range 2 begin> .. <case range 2 end>
Possible error cause: Two branches of CASE markers have the same elements or subsets.
Error correction: Make sure that there is no intersecting.

```
Example of the PROGRAM PLC_PRG
error:
VAR
            i : INT;
END VAR
CASE i OF
    3..5: i := i+2;
    1..4: i := i+2;
ELSE
    i := i+10;
END CASE;
--> C0219: Case contains overlapping range 1 .. 4 and 3 .. 5
```


## Compiler error C0221

Message: Direct Address '<address>' malformed
Possible error cause: An address is not displayed completely.
Error correction: Make sure that the address is displayed correctly.

```
Example of the PROGRAM PLC_PRG
error: VAR
            xVar : BOOL;
    END VAR;
    xVar := %IX0;
    --> C0221: Direct Address '%IXO' malformed
Error correc- Example:
tion: xVar := %IX0.2;
```


## Compiler error C0222

Message: Outputs can't be of type 'REFERENCE TO'
Possible error cause: An attempt is made to define REFERNCE TO as an output parameter.
Error correction: Do not use REFERENCE TO as an output parameter.

```
Example of the PROGRAM PLC_PRG
error:
VAR
        inst : FB;
END_VAR
FUNCTION BLOCK FB
VAR_OUTPUT
    re : REFERENCE TO INT;
END VAR
--> C0222: Outputs can't be of type 'REFERENCE TO'
```


## Compiler error C0224

Message: Call Recursion: <recursion>
Possible error cause: A function calls itself.
Error correction: Make sure that functions are not recursive.

| Example of the | PROGRAM PLC_PRG |
| :--- | :--- |
| error: | VAR |
|  | END_VAR |
|  | POU (); |
|  | FUNCTION POU |
|  | VAR |
|  | END_VAR |
|  | POU(); |
|  | $-->C 0224:$ Call Recursion: POU -> POU |

## Compiler Error C0225

Message: '<name>' is not an instance of '<name>'
Possible error cause: A function block in a graphical programming language has been assigned with an explicitly specified type that does not match the declared type.
Error correction: Replace the explicit type with the one used in the declaration part, or remove the specification of the explicit type from the POU.

## Example of the

 error:

PROGRAM PLC_PRG
VAR
fbVar : MyFB;
END_VAR
--> C0225: 'fbVar' is not an instance of 'MyFB2'

## Error correction:


or


## Compiler error C0227

Message: Initialisation of constant variable <constant name> not constant
Possible error cause: A constant is initialized with a variable.
Error correction: Initialize constants only with constant values.

```
Example of the PROGRAM PLC_PRG
error:
VAR
            i : INT;
    END_VAR
    VAR CONSTANT
    k : INT := i;
    END_VAR
    --> C0227: Initialisation of constant variable 'k' not constant
```


## Compiler error C0228

Message: No initial value for constant variable '<constant name>'
Possible error cause: A constant is not initialized.
Error correction: Initialize the constants.

| Example of the | PROGRAM PLC_PRG |
| :--- | :--- |
| error: | VAR |
|  | END_VAR |
|  | VARCONSTANT |
|  | EN: INT; |
|  |  |
|  | $-->C 0228: ~ N o ~ i n i t i a l ~ v a l u e ~ f o r ~ c o n s t a n t ~ v a r i a b l e ~ ' k ' ~$ |

## Compiler Error C0230

Message: Type name '<data type>' not expected in this place
Possible error cause: The data type name of an enumeration is used at an invalid position.
Error correction: Check whether the data type name is used correctly at this location. Maybe there is a spelling error.

```
Example of the TYPE MyEnum :
error: (
    enum_member := 0
    );
    END_TYPE
    PROGRAM PLC_PRG
VAR
    value : INT;
END_VAR
value := MyEnum;
MyEnum := value;
--> For PLC_PRG, the error message is issued 2x:
C0230: Type name 'MyEnum' not expected in this place
Error correc- Example:
tion: value := MyEnum.enum member;
MyEnum.enum_member := value;
```


## Compiler Error C0232

Message: Array initialisation expected
Possible error cause: An array of arrays is initialized, but the initialization values are not nested.

Error correction: Use a nested array initialization as shown in the example below.

```
Example of the PROGRAM PLC_PRG
error: VAR
    value : ARRAY[0..2] OF ARRAY[0..2] OF INT := [1,2,3];
END_VAR
--> C0232: Array initialisation expected
Error correc- Example:
tion: value : ARRAY[0..2] OF ARRAY[0..2] OF INT := [
    [1,2,3],
    [4,5,6],
[7,8,9]];
```


## Compiler Error C0233

Message: Initialisation list for $\{0\}$ <data type> expected
Possible error cause: An array of the type of a structure is initialized with elements that are not structure initializations or variables.

Error correction: As shown in the example below, use structure initializations or existing variables to initialize arrays of structures.

| Example of the error: | ```PROGRAM PLC_PRG VAR values : ARRAY[0..2] OF COLOR := [1,2,3]; END_VAR``` |
| :---: | :---: |
|  | --> C0233: Initialisation list for COLOR expected |
| Error correction: | ```Example: PROGRAM PLC_PRG VAR colorVariable : COLOR := (red:=0, green:=0, blue:=255); value : ARRAY[0..2] OF COLOR := [ colorVariable, (red:=255, green:=0, blue:=0), (red:=0, green:=255, blue:=0)];``` |
|  | END_VAR |

## Compiler error C0234

Message: First Operand of $\qquad$ QueryInterface must be an interface reference or the instance of a function block

Possible error cause: Incorrect operands are passed to the operator $\qquad$ QueryInterface.

Error correction: Pass an interface reference or the instance of a function block.

| Example of the error: | ```PROGRAM PLC_PRG VAR a : INT; ITFref, ITFref2 : ITF; ITFref2 : ITF2; END_VAR``` |
| :---: | :---: |
|  | QueryInterface(a ,ITFref) |

```
INTERFACE ITF EXTENDS SYSTEM.IQueryInterface
INTERFACE ITF2 EXTENDS ITF
--> C0234: First Operand of QueryInterface must be an interface
reference or the instance of a function block
Error correc- Example:
tion:
__QueryInterface(ITFref2, ITFref);
```


## Compiler error C0235

Message: Second Operand of $\qquad$ QueryInterface must be an interface reference Possible error cause: Incorrect operands are passed to the operator $\qquad$ QueryInterface. Error correction: Pass an interface reference.

```
Example of the PROGRAM PLC_PRG
error:
VAR
        a : INT;
        ITFref, ITFref2 : ITF;
            ITFref2 : ITF2;
    END_VAR
    __QueryInterface(ITFref2, a);
    INTERFACE ITF EXTENDS __SYSTEM.IQueryInterface
    INTERFACE ITF2 EXTENDS ITF
--> C0235: Second Operand of __QueryInterface must be an interface
reference
Error correc- Example:
tion: __QueryInterface(ITFref2, ITFref);
```


## Compiler error C0236

Message: Wrong type definition for VAR_EXTERNAL <variable name>
Possible error cause: The variable is declared in VAR_GLOBAL / VAR_EXTERNAL as different types.
Error correction: Use the same type definition in VAR_GLOBAL and VAR_EXTERNAL.

```
Example of the PROGRAM PLC PRG
error:
VAR_EXTERNA\overline{L}
            ig : STRING;
END VAR
VAR_GLOBAL
            ig : INT;
END_VAR
--> C0236: Wrong type definition for VAR_EXTERNAL ig
```


## Compiler error C0237

Message: No global definition found for VAR_EXTERNAL '<variable name>'

Possible error cause: An attempt is made to declare a variable in VAR_EXTERNAL which does not exist in VAR_GLOBAL.
Error correction: Make sure that the identifiers match.

```
Example of the PROGRAM PLC_PRG
error:
VAR_EXTERNA\overline{L}
END_VAR
VAR_GLOBAL
            ig : INT;
END_VAR
--> C0237: No global definition found for VAR_EXTERNAL 'i'
```


## Compiler error C0238

Message: No initial value allowed for VAR_EXTERNAL <variable name>
Possible error cause: An attempt is made to initialize a variable in VAR_EXTERNAL.
Error correction: Do not initialize variables in VAR_EXTERNAL.

```
Example of the PROGRAM PLC_PRG
error:
VAR_EXTERNA\overline{L}
    -ig : INT := 2;
    END_VAR
VAR_GLOBAL
        ig : INT;
END_VAR
--> C0238: No initial value allowed for VAR_EXTERNAL ig
```


## Compiler error C0239

Message: Interface <interface name 1> does not extend <interface name 2>
Possible error cause: The used interface does not extend another interface.
Error correction: Extend the interface.

```
Example of the PROGRAM PLC_PRG
error:
VAR
        ITFref : ITF;
        ITFref2 : ITF2;
END_VAR
__QueryInterface(ITFref2,ITFref);
INTERFACE ITF
INTERFACE ITF2 EXTENDS ITF
--> C0239: Interface ITF__Union does not extend
___System.IQueryInterface
Error correc- Example:
tion: INTERFACE ITF EXTENDS __System.IQueryInterface
```


## Compiler error C0240

Message: First Operand of $\qquad$ QueryPointer must be an interface reference or the instance of a function block

Possible error cause: Incorrect operands are passed to the operator $\qquad$ QueryPointer.

Error correction: Pass an interface reference or the instance of a function block.

```
Example of the PROGRAM PLC_PRG
error: VAR
        a : INT;
        ITFref : ITF;
        pt : POINTER TO FB;
    END VAR
    __QueryPointer(a,pt);
    --> C0240: First Operand of __QueryPointer must be an interface
    reference or the instance of a function block
Error correc- Example:
tion: __QueryPointer (ITFref, pt);
```


## Compiler error C0241

Message: Second Operand of __QueryPointer must be pointer
Possible error cause: Incorrect operands are passed to the operator $\qquad$ QueryPointer.

Error correction: Pass a pointer.

```
Example of the PROGRAM PLC_PRG
error: VAR
            b : INT;
            ITFref : ITF;
            pt : POINTER TO FB;
    END_VAR
    __QueryPointer(ITFref,b);
    INTERFACE ITF EXTENDS __System.IQueryInterface
--> C0241: Second Operand of __QueryPointer must be pointer
Error correc- Example:
tion: __QueryPointer (ITFref, pt);
```


## Compiler error C0242

Message: Operand of $\qquad$ DELETE must be pointer
Possible error cause: An incorrect operand is passed to the operator $\qquad$ DELETE.

Error correction: Pass a pointer.

```
Example of the PROGRAM PLC_PRG
error: VAR
            a : INT;
            pt : POINTER TO INT;
END_VAR
__DELETE(a);
--> C0242: Operand of __DELETE must be pointer
Error correc- Example:
tion:
```

```
__DELETE (pt);
```

```
__DELETE (pt);
```


## Compiler error C0243

Message：The name used in the signature is not identical to the object name Possible error cause：The object name differs from the name used in the code．

Error correction：make sure that the names are the same．

## Example of the

 error：| Devices $\quad 7 \times$ | 速 PLC＿PRG $\times$ |  | 㦴 FB $\times$ |
| :---: | :---: | :---: | :---: |
| ```- - Unbenamont204 - - Device (CODESYS Control Win V3) = 酋\| PLC Logic Application (ifl) Library Manager 目 FB(FB) # PLC_PRG (PRG) - Task Configuration - MainTask (IEC-Tasks) PLC_PRG``` | $=\begin{array}{r} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \end{array}$ | ```PROGRAM PLC_PRG VAR inst : F; END_VAR``` | 1 FUNCTION＿BLOCK F <br> 2 VAR＿INPUT <br> 3 BID＿VAR <br> 4 VAR＿OUTPUT <br> 5 EID＿VAR <br> 6 VAR <br> 7 EID＿VAR <br> 8  <br>   |

## Compiler Error C0380

Message：The Operators LOWER＿BOUND and UPPER＿BOUND are only supported for arrays of variable length．

Possible error cause：One of the two operators LOWER＿BOUND or UPPER＿BOUND is not used for an array of variable length．

Error correction：Use the operators LOWER＿BOUND and UPPER＿BOUND only for an array of variable length．

For compiler version 3．5．14．0 and higher，the operators can also be used for static arrays．As a result，the error C0380 occurs only in the case of earlier compiler versions．

```
Example of the FUNCTION_BLOCK POU
error:
VAR_IN_OUTT
            arrin : ARRAY [*] OF INT;
END VAR
VAR
        arrtest : ARRAY [0..5] OF INT;
        test1: DINT;
        test2: DINT;
```

```
END_VAR
test1 := UPPER_BOUND(arrin, 1);
test2 := UPPER_BOUND(arrtest, 1);
--> C0380: The operators LOWER_BOUND and UPPER_BOUND are supported
only for arrays with variable I
```


## Compiler error C0509

Message: Multiple assignments for operator '__New' not allowed
Possible error cause: In one line of code, the assignment operator " $:=$ " is called a multiple number of times with the $\qquad$ New operator.

Error correction: Program the memory allocation with the __ New operator in a separate line of code for each pointer that points to dynamically allocated memory.

```
Example of the PROGRAM PLC_PRG
error: VAR
    pbAlpha : POINTER TO BYTE; // Typed pointer to Alpha
    pbBeta: POINTER TO BYTE; // Typed pointer to Beta
    xInit : BOOL := TRUE;
    xDelete : BOOL;
END_VAR
IF (xInit) THEN
    pbBeta := p.bAlpha := __NEW(BYTE); // Incorrect code for memory
allocation
END_IF
pbBeta := pbAlpha := 16#01;
IF (xDelete) THEN
        DELETE(pbAlpha); // Frees memory of pointer
END_\overline{IF}
--> C0509: Multiple assignments for operator '___New' not allowed
```


## Error correc-

## tion:

```
PROGRAM PLC_PRG
VAR
pbAlpha : POINTER TO BYTE; // Pointer to Alpha
    pbBeta: POINTER TO BYTE; // Pointer to Beta
    xInit : BOOL := TRUE;
    xDelete : BOOL;
END VAR
IF (xInit) THEN
    pbAlpha := __NEW(BYTE); // Allocates memory for Alpha
    pbBeta := __NEW(BYTE); // Allocates memory for Beta
END
pbBeta := p.BAlpha := 16#01; // Multiple assignment
IF (xDelete) THEN
_DELETE(pbAlpha); // Frees memory of pointer
```

See also

- $\quad 4$ Chapter 6.4.1.20.3.59 "Operator '__NEW" on page 2201
- $\quad$ Chapter 6.4.1.20.6.3.13 "Attribute 'enable_dynamic_creation'" on page 2281


## Compiler error C0511

Message: The function block '<function block name>' is ABSTRACT and cannot be used as a target for an assignment.

Possible error cause: A value was assigned to an abstract function block. The concrete function blocks may have different types and therefore cannot be copied.

Error correction: In order to copy the data of the function block, concrete function blocks have to be used.

```
Example of the PROGRAM PLC_PRG
error:
VAR
refAbstract1 : REFERENCE TO AbstractPOU;
refAbstract2 : REFERENCE TO AbstractPOU;
END_VAR
refAbstract1 := refAbstract2;
--> C0511: The function block 'refAbstract1' is ABSTRACT and cannot
be used as a target for an assignment.
```


## Error correction:

```
Use the reference assignment \(\mathrm{REF}=\) to assign the reference refAbstract1 to the same function block as refAbstract 2 .
```


## Compiler Error C0542

Message: Inheritance is not intended for the data type "UNION" <data type name>.
Possible error cause: A structured data type (DUT) is derived from a UNION by extending with EXTENDS, or a UNION is derived from a DUT. This kind of derivation is not permitted. However, for reasons of compatibility only a warning is issued.

## Example of the error

```
TYPE U_StringExt EXTENDS U_StringBase :
UNION
    str10 : STRING(10);
END_UNION
END_TYPE TYPE U_StringBase :
UNIŌN
            str20 : STRING(20);
END UNION
END_TYPE PROGRAM PLC_PRG
VAR
            uStringExt : U_StringExt;
END_VAR
uStringExt.str20 := 'a234567890.b234567890'; -> C0542
```


## Compiler Error C0543

Message: The name <keyword> is a reserved keyword in the IEC 1131-3 standard. An error will be issued in future versions.

Possible error cause: A reserved keyword was assigned as the name of a variable.
Error correction: Rename the variable.

```
Example of the PROGRAM PLC_PRG
error:
VAR
    char : BYTE;
END_VAR
--> C0543: The name 'char' is a reserved keyword in the IEC 1131-3
standard. An error will be issued in future versions.
```

Note: For violations in compiled libraries, only a text message (information) will be issued instead of a warning.

The following keywords are reserved:

- CHAR
- WCHAR
- ANY_DERIVED
- ANY_ELEMENTARY
- ANY_MAGNITUDE
- ANY_SIGNED
- ANY_DURATION
- ANY_CHARS
- ANY_CHARS
- CHAR_TO
- TO_CHAR
- WCHAR_TO
- TO_WCHAR
- ATAN2
- USING
- CLASS

See also

- ๕ Chapter 6.4.1.20.9 "Keywords" on page 2335


## Compiler Error C0549

Message: Initialization of the static variable '<variable name>' is not constant, or replaced constants is disabled.

Possible error cause: VAR_STAT and CONSTANT declarations are used together in a function block declaration. The Replaces constants compile option is disabled.

## Example The Replace constants compile option is disabled.

```
VAR CONSTANT
    iMaxInst : INT := 2;
END_VAR
VAR_STAT
    iIDs : INT := iMaxInst - 1;
END_VAR
```

Message: C0549: Initialization of the static variable 'ilDs' is not constant, or replaced constants is disabled.

Error correction: In Project $\rightarrow$ Project Settings, on the "Compile Options" tab, enable the "Replace constants " compile option.

## Compiler Error C0550

Message: Attribute 'pack_mode' for 'FUNCTION' ( 'METHOD') not allowed
Possible error cause: Functions or methods which are attributed with the pragma \{attribute 'pack_mode' := ' <pack mode value>'\}

## Example

```
{attribute 'pack_mode' := '2'}
METHOD METH : INT
VAR INPUT
END_VAR
{attribute 'pack_mode' := '1'}
FUNCTION FunPackēd : DINT
VAR_INPUT
        by1: BYTE;
END VAR
VAR
    by2: BYTE;
END_VAR
```


## Message:

C0550: Attribute 'pack_mode' for 'METHOD' not allowed C0550: Attribute 'pack_mode' for 'FUNCTION' not allowed

Error correction: Delete the attribute pragma.

## Compiler Error C0554

Message: No explicit calls of '<FB name>' allowed. <value of attribute 'no_explicit_call'> Possible error cause: A function block with the no_explicit_call attribute has been called.

## Example

```
{attribute 'no_explicit_call' := 'Call of this FB is not allowed'}
FUNCTION BLOCK FB1
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
END_VAR
PROGRAM PLC_PRG
VAR
    myFB: FB1;
END_VAR
myFB();
```


## Message：

C0554：No explicit calls of＇FB1＇allowed．Call of this FB is not allowed．
Error correction：Delete the attribute pragma．

## Compiler Error C0555

Message：The string literal＜literal＞contains characters that cannot be displayed．The project option＇UTF－8 encoding for STRING＇could be used．

Possible error cause：Input of unknown characters which cannot be transformed to Latin－1．

## Example

```
PROGRAM PLC_PRG
VAR
    str1: STRING(200);
    str2: STRING := '1 €';
    str3: STRING(200) := ' !$"#$$%&$'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~';
END_VAR
str1 := '你好,世界';
str1 := 'ABC abc 123 äöü ß#~=" §% /(!)[]{}\ `^** }€\mu *__.; 中文字
\Omega ';
str1 := '1 £';
str1 := '1 €';
str1 := ' !$"#$$%&$'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~';
```


## Message：

C0555：The string literal＇你好，世界＇contains characters that cannot be displayed．The project option＇UTF－8 Encoding for STRING＇could be used．
Error correction：Select the option．

### 6.4.1.21 Reference, User Interface

6.4.1.21.1 Notifications ..... 2408
6.4.1.21.2 Objects. ..... 2409
6.4.1.21.3 Menu Commands ..... 2551
6.4.1.21.4 Dialogs. ..... 2745

### 6.4.1.21.1 Notifications

Notifications inform you about important information, such as available updates or security notices.

To open the "Notifications" view, click the $\boldsymbol{\Gamma}$ icon in the upper right corner of the frame window of CODESYS. All received notifications are displayed in this view. Notifications marked as "read" are deleted from the list the next time CODESYS is started.

The red icon $\boldsymbol{~}^{3}$ indicates that new notifications are available, as well as how many.
6.4.1.21.2.1 General ..... 2409
6.4.1.21.2.2 Object 'Application'. ..... 2410
6.4.1.21.2.3 Object 'POU Locations' ..... 2411
6.4.1.21.2.4 Objects for Alarm Management. ..... 2412
6.4.1.21.2.5 Object 'Data Source Manager' ..... 2412
6.4.1.21.2.6 Object 'Data Source'. ..... 2414
6.4.1.21.2.7 Object 'DeviceTrace' ..... 2426
6.4.1.21.2.8 Object 'Device' and Generic Device Editor ..... 2427
6.4.1.21.2.9 Object 'DUT'. ..... 2461
6.4.1.21.2.10 Object 'External File' ..... 2464
6.4.1.21.2.11 Object 'GlobalTextList' ..... 2465
6.4.1.21.2.12 Object 'GVL' - Global Variable List. ..... 2465
6.4.1.21.2.13 Object 'GVL' - Global Variable List (task-local) ..... 2466
6.4.1.21.2.14 Object: IEC Symbol Publishing ..... 2466
6.4.1.21.2.15 Object 'Image Pool'. ..... 2468
6.4.1.21.2.16 Object 'Library Manager' ..... 2469
6.4.1.21.2.17 Object 'OPC UA Information Model'.... ..... 2472
6.4.1.21.2.18 Object 'Network Variable List (Sender)' ..... 2475
6.4.1.21.2.19 Object 'Network Variable List (Receiver)' ..... 2475
6.4.1.21.2.20 Object 'Persistent variable list'. ..... 2476
6.4.1.21.2.21 Object 'POU' ..... 2477
6.4.1.21.2.22 Object 'POUs for Implicit Checks' ..... 2500
6.4.1.21.2.23 Object 'Project Settings'. ..... 2514
6.4.1.21.2.24 Object 'Project Information'. ..... 2515
6.4.1.21.2.25 Object 'Recipe Manager' ..... 2519
6.4.1.21.2.26 Object 'Recipe Definition' ..... 2522
6.4.1.21.2.27 Object 'Symbol Configuration' ..... 2523
6.4.1.21.2.28 Object 'Text List'. ..... 2532
6.4.1.21.2.29 Object 'Task Configuration'. ..... 2533
6.4.1.21.2.30 Object 'Task'. ..... 2538
6.4.1.21.2.31 Object 'Trace'. ..... 2541
6.4.1.21.2.32 Object 'Trend Recording Manager'. ..... 2545
6.4.1.21.2.33 Object 'Trend Recording'. ..... 2545
6.4.1.21.2.34 Object 'Trend Recording Task'. ..... 2548
6.4.1.21.2.35 Object 'Unit Conversion'. ..... 2549

## General

Objects in CODESYS provide special functionalities to create applications. Examples: Application, program, function, Library Manager, devices, image pool. Objects are managed in tree structures in the views "Devices", "POUs" and "Modules".

You can add an object to the belonging "tree" by use of the command "Project $\rightarrow$ Add Object". The possible insert positions depends on the position within the tree.
Each object provides properties, which can be viewed and accessed with the command from the context menu of the object.
See also

- 参 Chapter 6.4.1.21.3.4.22 "Command 'Properties"' on page 2597


## Object 'Application'

## Symbol:

The object is displayed as a node in the device tree. It comprises the objects which are required for a controller program to run.
You can insert an application object below a "PLC Logic" node (below a programmable device) or as a child application below an existing application (parent application).
Below each application, there has to be a "Task Configuration" where you configure which program of the application will be called by which task and using which settings.
Furthermore, you insert the POUs of your controller program below an application, for example POUs, global variables lists, and the Library Manager. These POUs are available only for this application and its child applications.
In addition, the application can also use instances of project-global POUs. You manage projectglobal POUs in the "POUs" view. The use of these kinds of instances follows the thinking behind object-oriented programming.
Multiple applications can be inserted below a PLC device object. To do this, they have to have unique names.


## NOTICE!

An online change after a changing the parent application will remove the child application from the PLC.

When multiple applications are directly below a device object, for the I/O handling of the device you have to define the application whose variables CODESYS should use for communication with the target system. The settings are configured on the "PLC Settings" tab of the device editor.

The application that you want to work with in online mode has to be set as the "active application" (see "App2" in the figure above).
You can set special properties for an application on the "Application Build Options" tab of the "Properties" dialog of the application object. Example: Activation of dynamic memory allocation.
When downloading the application to the PLC, you can include information about the application contents. This is also a setting on the "Application Build Options" tab. Then later you can compare the application on the controller with the active application in CODESYS.

If you want to add individual information about the author, version, and an individual short description, you can modify the information in the general "Project Information" on the "Information" tab of the "Properties" dialog.
When you want to log in to the target system (PLC or simulation) with an application, it will first be checked which applications are currently on the PLC and whether or not the application parameters on the controller match those in the project configuration. Corresponding messages will notify you about mismatches and possible options for further action. In this step, you can also delete applications from the PLC.
On the "Application" tab of the device editor, you can see which applications currently exist on the device. There you can also delete applications from the target system. It is possible that you also see additional applications which are not represented by a separate object in the device tree, for example the <application>_symbols.app, which contains a symbol list created for the application (see "Symbol Configuration").

See also

- y Chapter 6.4.1.21.2.29 "Object 'Task Configuration'" on page 2533
- ${ }^{4}$ Chapter 6.4.1.21.4.11.10 "Dialog 'Properties - Application Build Options'" on page 2759
- ${ }^{4}$ Chapter 6.4.1.11 "Downloading an Application to the PLC" on page 1965
- ⓨ Chapter 6.4.1.21.2.8.10 "Tab 'PLC Settings'" on page 2439
- $\xi^{\star}$ Chapter 6.4.1.10.3 "Symbol Configuration" on page 1941
- ⓨ Chapter 6.4.1.14.2 "Executing the online change" on page 2025
- ⓢ Chapter 6.4.1.21.3.5.13 "Command 'Project information'" on page 2604


## Object 'POU Locations'

This object is available only for specific controllers. It is displayed automatically in the device tree. The object cannot be added or removed manually. The object can be used for mapping the executable code of an application in different code areas on the controller. Specifically small controllers often have limited internal code areas (flash memory). If one or more additional code areas (for example, external flash memory) are available on the controller, then the location of the code POUs of an application can be changed specifically.
If there are no specific requirements, then the code POUs are stored sequentially in the code areas ("default"). This means that the next code POUs are stored in the next areas only when the first code area is filled. In the "POU Locations" editor, you will see the current location of the POUs in the memory areas and you can change them specifically.

Editor 'POU Locations'

Double-clicking the "POU Locations" object in the device tree of the controller opens the editor. Then it receives the entry "<application>". After a code generation, all program blocks of the application are displayed with the respective object type, current location in the memory, and code size.

In the "Configured Location" column, you can set one of the memory areas other than the "Current Location" for each POU or library.
In order to move the POUs to the recently configured memory locations, you first have to "Clean" and then "Generate Code" again.

Make sure to pay attention to the messages in the category "POU Locations". This also shows when a code POU cannot be moved as expected.

| "Clean" | Deletion of the compile information for the application. Corresponds to the menu <br> command "Build $\rightarrow$ Clean". Requirement for moving the POUs to the configured <br> memory locations. |
| :--- | :--- |
| "Generate code" | Starting of the code generation for the application. Corresponds to the menu <br> command "Build $\rightarrow$ Generate Code". Requirement for moving the POUs to the <br> configured memory locations. |
| "Objects" | Objects of the application, including the objects from the referenced libraries |
| "Type" | Object type; examples: "Function block", "Method", "Library" |
| "Current location" | Current memory location of the POU: area_<n>. |
| "Configured location" | Configured memory location where the POU is moved at the next code genera- <br> tion. Possible values: <br> - "default": Automatically assigned area. <br> $\bullet \quad$ "area_<n>": Explicitly assigned memory area (n=number) |
| "Code size" | Code size of the POU (in bytes) |

- ${ }^{4}>$ Chapter 6.4.1.21.3.6.2 "Command 'Clean'" on page 2618
- «y Chapter 6.4.1.21.3.6.1 "Command 'Generate Code"' on page 2618


## Objects for Alarm Management

The help pages for alarm management are summarized in the help for CODESYS Visualization. So please see there for help on the following objects::

- Object "Alarm Configuration"
- Object "Alarm Class"
- Object "Alarm Group"
- Object "Alarm Storage"
- Object "Remote Alarms"


## Object 'Data Source Manager'

Symbol:
The object is used as a node for data sources below it. At least one data source has to exist. An application with the data source manager communicates with remote devices.
See also

- $\boldsymbol{y y}^{2}$ Chapter 6.4.1.10.6 "Data Link with Data Sources" on page 1947

Command 'Add Object' > 'Data Source'

Function: The command opens the "Add Data Source" dialog.

## Call

- Menu bar: "Project"
- Context menu in the "Devices" view of the CODESYSperspective
- Context menu in the "Data Sources" view of the "HMI" perspective

Requirement: The "Data Source Manager" object is selected that should have an additional data source.

Dialog 'Add
Data Source'

| "Name" | Example: Data_Source_A |
| :---: | :---: |
| "Select data source type" | Data source type that matches the controller configuration in order to establish communication. <br> - "CODESYS Symbolic": <br> Requirement: The local device is a CODESYS HMI. <br> The data is transmitted by means of symbolic monitoring. This requires that symbols are configured in the symbol configuration of the remote PLC application. <br> Note: As long as the symbol configuration is not impacted by an application change, you have the advantage that the application in the local device does not have to be updated. <br> Hint: Use this communication connection unless there are no resources available in the remote PLC for a symbol configuration. <br> - "CODESYS ApplicationV3": <br> The data is transmitted via the CODESYS address protocol. This requires that the address information between the remote PLC and the local device match. Otherwise a connection cannot be established. <br> Advantage: A symbol configuration is not required in the remote application. Note: For changes to the remote application, the local application has to be updated (for example, the HMI application). <br> Hint: Use this communication for embedded or mini PLCs when there are no available resources for the symbol configuration. <br> - "OPC UA Server": <br> Data is transferred from an OPC UA server to the local controller via a TCP connection. |
| "Add" | Opens the "Initialize Data Source - Provider settings" dialog. The contents of the dialog depend on the selected data source type. |

## Dialog 'Initialize Data Source Wizard - Provider settings' (for 'CODESYS ApplicationV3')



The settings of this dialog are described in the following chapter: Object 'Data Source' - Tab 'Communication".

The dialog is used to configure the connection initially when you have selected "CODESYS ApplicationV3" as the data source type. The communication is done by means of address monitoring.
See also

- $\Leftrightarrow$ Chapter 6.4.1.21.2.6.5 "Tab 'Communication' via CODESYS ApplicationV3" on page 2422

Dialog 'Initialize

Data Source
Wizard - Provider settings' (for 'OPC UA Server')

The settings of this dialog are described in the following chapter: Object 'Data Source' - Tab 'Communication".

The dialog is used to configure the connection initially when you have selected "OPC UA Server" as the data source type. The communication takes place over a TCP connection.

See also

- ${ }^{4}$ ) Chapter 6.4.1.21.2.6.6 "Tab 'Communication' via OPC UA Server" on page 2424


## Dialog 'Initialize Data Source <br> Wizard - Browse data items'

The settings of this dialog are described in the following chapter: Object 'Data Source' - Dialog 'Choose Variables'.

Function: You can select the variables for data transmission from the variables of the remote PLC. By clicking "Finish", the data source is initialized and the data types and variables (data interface) are declared below the folder "DataSources_Objects". You can modify the settings in the editor of the data source object.
Call: Automatic
See also

- ⓨ "Dialog 'Choose Variables'" on page 2416


## Object 'Data Source'

## General

Symbol:
In the editor (object type "Data Source"), the access to the data of a remote device is managed on the "Variables", "Type Mappings", "Communication", and "General and Diagnosis" tabs.

[^35]CODESYS ApplicationV3 (D: \Projects\Project_A): Project_A.App_A

See also

- « Chapter 6.4.1.10 "Working with Controller Networks" on page 1936


## Tab 'Variables'

The variables for the data originating from the remote source are declared in the global variable list <name of data source>. The global variable list acts as a data interface to the remote PLC. The object is located below the application and below the "DataSources_Objects" folder.

| "Update variables" | Establishes a connection to the remote device and opens the "Choose <br> Variables" dialog. |
| :--- | :--- |


| "Local variable" | Variable in the local application. Contains the remote data. |
| :---: | :---: |
| "Access rights" | Access rights of the variables. The respective remote variable has the same access rights. <br> - Write access. Every time the values changes, the variable is updated on the controller. <br> - $)^{0}$ : Read access. Every time the values changes on the controller, the variable is updated in the application. <br> - *: Read/Write access <br> Note: If you change the access rights, then a download is required for the change to go into effect. |
| "Update always" | $\square$ The controller data is updated automatically (via the data source). A variable is updated automatically if it is used in the visualization, trend, recipe, or as an alarm. <br> Note: This is the recommended setting type. <br> The variable is updated in each cycle. <br> Note: Select the option only when the variable is used exclusively in IEC code. If a variable is used in the visualization code, then it is updated automatically. <br> Note: When an instance of a function block or a data type is updated in this way, the instance is always transferred completely. |
| "Create or map" | Mapping type for how the remote variable is mapped to the local variable. <br> - Mapping to a specific created variable with the data type of the remote variable. The control data is mapped $1: 1$. That is the recommended mapping type. The variable is declared in the GVL <name of data source>. <br> - ": Mapping to an existing variable. This requires that the existing variable has the same data type. <br> - $\quad$ 各: Mapping to a specific created variable with type-conforming data type to the remote data type: remote and local data types are not the same, but compatible. For example, a type-conforming data type can be available in a library. The variable is declared in the GVL <name of data source>. |
| "Type mapping" | Data type of the remote variable. If the variable is not a scalar type, then the type is listed on the "Type Mappings" tab. |
| "Remote variable" | Variable in the remote PLC |

Function: The dialog lists the remote variables that are accessed by means of the configured connection.

Call: "Update variables" command on the "Variables" tab.
Requirement: The remote PLC is running. The control application is downloaded.

| "Variables" | The remote variables are listed in the tree view. The top node is identified by the <br> remote application name. Its variables are listed below that. Structured data is <br> listed with all of its subordinate elements. <br> Example: appControl_A <br> V: The variable is selected for transferring to the local device. When the variable <br> is structured, it is applied with all subelements. If the variables themselves are <br> subelements, then only this subelement is applied without accepting the struc- <br> ture completely. <br> Red font: When a variable is displayed in a red font, the variable is not available <br> (anymore) in the remote PLC. <br> Note: You can click "Uncheck unavailable variables" to remove the variable from <br> the list. <br> $\square:$ The variable is not selected for the transfer. |
| :--- | :--- |
| $\pm$ | The variable has expandable elements. By clicking the symbol, the variable is <br> extended by their elements. |
| "Insert the items structured" | $\square:$ The selected variables are transferred with this structure if they are struc- <br> tured. <br> $\square:$ The variable is transferred unstructured with a scalar data type. |
| "Uncheck unavailable | Requirement: The link is visible when previously are no longer available in the <br> variable available on the remote PLC. These variables are marked in red in the <br> window above. The symbol configuration or the application presumably changed <br> in the remote PLC. <br> By clicking the command, the red variables are removed from the list box. |

## Tab 'Type Mappings'

The tab lists the non-scalar data types as they are currently available in the
"DataSources_Objects" folder. You can edit or delete the data type declaration by selecting a data type and then the declared elements in the lower window. Moreover, you can modify the name, reset access rights, map another type, or select another remote variable.

Tab 'Type Map-
pings'

| "Local type" | Data type in the local application |
| :---: | :---: |
| "Create or map" | - $\quad$ : Mapping to a new created data type. Declared in the "DataSources_Objects" folder. <br> - ": Mapping to an existing data type <br> - \%: Mapping to a type-conforming data type. Declared in the "DataSources_Objects" folder. |
| "Mapping name" | Name of the data type |
| "Remote type" | Data type of the remote PLC |


| List with the subordinate elements of the selected data type. |  |
| :--- | :--- |
| "Local variable" | Local variable name of the element of the selected data type |
| "Access right" | Access rights to the element |
| "Type mapping" | Data type of the element |
| "Remote variable" | Remote variable name of the element of the selected data type |


| $[D e l]$ | Removes the selected element |
| :--- | :--- |

See also

- \# Chapter 6.4.1.10.6.3 "Editing data source variables" on page 1954


## Tab 'Communication' via CODESYS Symbolic

The tab includes the communication settings via CODESYS Symbolic for the remote data source.

When initially adding a data source, you have selected the "CODESYS Symbolic" data source type, and depending on that the communication settings to the data source were configured. Afterwards, the communication settings are outdated on this tab. You can only initially set the "Data source type" setting.
CODESYS Symbolic means that in the case of an active connection the communication is done via symbolic monitoring. This kind of symbolic access is possibly for CODESYS V2 and CODESYS V3 controller variants. In addition, the runtime system has to support the symbol configuration.

You can develop a local application offline based on the symbol information without a connection to the data source. To do this, you refer to a symbol file in the configuration settings in which all required variable information has been stored. Then no active connection is established.

Tab 'Communication' via CODESYS Symbolic

| "Variable information" | Source of the variable information <br> - "From connection settings" <br> A connection is established actively according to the communication settings <br> specified below ("Connection type" and "Connection Settings"). The variable <br> information is read from the remote controller application. <br> "<device names. <application name>.symbol configuration" <br> The variable information is read from the symbol configuration. The symbol <br> configuration is part of the active project and located in the device tree at the <br> object of the remote controller below the application. An active connection is <br> not established to the controller. <br> "From symbol file" <br> The variable information is read from a symbol configuration file that is <br> stored on the development system. In the "Choose symbol file" field, specify <br> this data. An active connection is not established to the controller. |
| :--- | :--- |
| The path of the symbol file for the "Variable information" selection is "From <br> symbol file". <br> The symbol file is stored on the development system and contains the |  |
| required variable information. By default, a symbol file path is created in the |  |
| project directory in the following structure: <project folder>\<project |  |
| name>. <device name>. <application name>. xml. |  |

## "Connection type"

Connection type between the remote PLC and the local device.
Depending on the selected connection type, the following settings below change.
Note: Whenever possible, avoid a direct connection without a gateway.

## - "CODESYS V2"

The devices exist in the same network. The V2 runtime on the remote PLC provides a communication interface.

- "CODESYS V2 (Via gateway)"

The devices do not exist in the same network. They are connected via a V2 gateway.

- "CODESYS V3"

The devices exist in the same network. The V3 runtime on the remote PLC provides a communication interface.

- "CODESYS V3 (Via gateway)"

The devices do not exist in the same network. They are connected via a V3 gateway.

## Connection set- <br> tings for con- <br> nection type <br> 'CODESYS V2'

| "PLC" |  |
| :---: | :---: |
| "Driver type" | - "Tcp/Ip (Level 2 Route)" <br> - "Tcp/lp (Level 2)" <br> - "Tcp/lp" |
| "Address" | Example: localhost (for the currently used system on your computer) |
| "Port" | Example: 1200 |
| "Block size" | Example: 128 <br> Requirement: The driver type is "Tcp/lp (Level 2)". |
| "Target ID" | Example: 0 <br> Requirement: The driver type is "Tcp/lp (Level 2 Route)". |
| "Motorola byte order" | Byte order on the PLC in big endian (Motorola format) <br> : Byte order in little endian (Intel format) |

Connection set-
tings for con-
nection type
'CODESYS V2
(Via gateway)'

| "Gateway" | The gateway settings are configured in addition to the PLC settings. <br> Note: For this connection, a "CoDeSys V2.3 Gateway Server" (V2 Gateway) <br> also has to be installed on the development computer where CODESYS V3 is <br> running. |
| :--- | :--- |
| "IP address" | Example: local host |
| "Port" | Example: 1217 |

## Connection set-

## tings for con-

nection type
'CODESYS V3'

| "PLC" |  |
| :--- | :--- |
| "Name or address of device" | The setting that you make here varies according to the selection in the "Type of <br> name or address" list box. For options that are derived automatically, you do not <br> have to specify the setting here. The setting can remain empty. <br> Example: Nothing specified for "...(automatically derived)" |
|  | Example: PLC_A for "Node name" <br> Example: $[$ ABCD] for "Node address" |
|  | Example: $192.168 .1 .5: 11741$ for "IP address" |
| Example: Pou. dssCommVar with data type |  |
| DatasourceSym. ConnectionSetup for "Dynamic from variable" |  |
| Hint: [..: Opens the input to select the program variables for |  |
| dynamic configuration. This variable has to be the data type |  |
| DatasourceSym. ConnectionSetup. |  |


| "Type of name or address" | - "Node name (automatically derived)" <br>  <br>  <br>  <br>  <br> - "Node address (automatically derived)" <br> - "IP address (automatically derived)" <br> - "Node name" <br> - "Node address" <br> - "IP address" <br> - "Dynamic from variable" |
| :--- | :--- |
|  | The device name or address is configured dynamically at runtime by means <br> of an IEC variable of data type DatasourceSym. ConnectionSetup. The <br> data type DatasourceSym. ConnectionSetup (STRUCT) is defined in the <br> Datasource Symbolic Access library. For the configuration, the structure <br> member xDataValid first has to be set to FALSE. If the address data has been <br> specified, then xDataValid has to be set back to TRUE. <br> Use case: The device name or address is not available when a project is being <br> created. <br> The dynamic configuration can also be used to change the settings at runtime <br> without restarting the HMI application. <br> Note: For this connection type, the connection is also not done dynamically via <br> gateway. |

## Connection set-

tings for con-
nection type
'CODESYS V3
(Via gateway)'

| "Gateway" | The gateway settings are configured in addition to the PLC settings. |
| :--- | :--- |
| "IP address" | Example: localhost |
| "Port" | Example: 1217 |

## Extending the communication settings for the PLCHandler interface



The connection to the controller is established via the CODESYS PLCHandler communication interface. In this case, the configuration is performed in the PLCHandler INI format and allows for advanced parameterization.

| "Advanced" |  |
| :--- | :--- |
| "Used as" | • "Don't use" <br> Recommended setting <br> The "INI content" property as well as any specified configuration settings <br> there are ignored. |
|  | "Extend the configuration by the following content" <br> As a rule, the configuration settings are used which are specified in the <br> "Connection Settings for CODESYS V3 (Via gateway)" property. Moreover, <br> the configuration settings are used in the "INI content" property. |
|  | "Configure completely with the following content" |
|  | The configuration settings of the "Configuration Settings for CODESYS V3 <br> (Via gateway)" property are ignored. Instead, only the configuration settings <br> are used in the "INI content" property. |


| "INI content" | Requirement: "Used as" is set to "Extend the configuration by the following content". <br> Example: <br> logfilter=16\#000000FF <br> Example: <br> parameter0=EncryptCommunication <br> value $0=1$ <br> Note: If the parameters are generic, then they can be specified as 0-based (parameter0 and value0). When extending, the numbering is automatically adjusted so that the extended parameters connect to the existing ones. The number of parameters (parameters $=\langle\mathrm{n}>$ ) is also set to the correct value. |
| :---: | :---: |
| "INI content" | Requirement: "Used as" is set to "Configure completely with the following content". <br> Example: ```[PLC:PLC_IdArti] interfacetype=ARTI active=1 logevents=1 motorola=0 nologin=0 timeout=10000 precheckidentity=0 tries=3 waittime=12 reconnecttime=10 buffersize=0 device=Tcp/Ip (Level 2 Route) instance=PLCWinNT_TCPIP_L2Route parameters=4 parameter0=Address value0=localhost parameter1=Port value1=1200 parameter2=TargetId value2=0 parameter3=Motorola byteorder value3=No``` |

## Communication settings for controllers with visualization user management

| "Login Configuration" | If a visualization user management is configured on the remote device, then <br> valid credentials are required at login. |
| :--- | :--- |
| "Type" | Defines how the visualization user management gets credentials <br> - "Login using the following credentials" |
|  | The credentials are hard-coded into the "User name" and "Password" set- <br> tings. They are used each time a connection is attempted. <br> - "Login using the credentials determined at runtime" <br> At runtime, a dialog opens and prompts the user to specify a user name and <br> password. Hard-coded credentials, which have nonetheless been specified <br> in "User name" and "Password", are ignored. |


| "User name" | Example: max.smith |
| :--- | :--- |
| "Password" | Example: $\bullet \bullet$ |

See also

- ${ }^{*}>$ Chapter 6.4.1.10.6.2 "Initially Adding a Data Source" on page 1949
- $\&$ Chapter 6.4.1.21.2.5 "Object 'Data Source Manager'" on page 2412


## Tab 'Communication' via CODESYS ApplicationV3

The tab includes the communication settings for a remote data source.
When initially adding a data source, you have selected the CODESYS ApplicationV3 data source type, and depending on that the communication settings to the data source were configured. Afterwards, the communication settings are outdated on this tab. You can only initially set the "Data source type" and "Select the project type" settings.

CODESYS ApplicationV3 means that in the case of an active connection the communication is done via address monitoring. In this case, the remote PLC is configured by directly specifying the device address or automatically via network scan.

Tab 'Communication' via CODESYS
ApplicationV3

| "Select the project type" | The project type indicates where the controller is configured: in the same project as the HMI application or in a separate project. <br> - "Current project" <br> The control application is part of the currently open project. The communication settings can be updated automatically or manually. <br> - "Other Project" <br> The control application is part of a separate project whose location is specified in "Choose file". The communication settings are done manually. <br> In the initial setting of the data source object, this option is fixed and influences which settings are available for "Target device". |
| :---: | :---: |
| "Choose file" | Name and path of the project that contains the control application (source project) <br> Example: D: \PLCs \PLC_A.project <br> Requirement: The "Select the project type" is "Other Project". |
| Window area for controllers of the project | Controllers and their subordinate applications, read from the selected project Example: $\begin{array}{r} =- \text { PLC_A } \\ \text { AppControl_A } \end{array}$ |

## Settings for 'Select the project type' == 'Current project'

| "Target device" | Note: The following settings are available when "Select the project type" is set to <br> "Current project". |
| :--- | :--- |
| "Automatic configuration" | O: The configuration is read automatically from the source project. This is the <br> recommended setting. <br> Example: "[DEVICE_A]" <br> Example: "[undetermined]": No configuration can be read. <br> Note: Make sure that the application is running on the controller and the network <br> path is active. The communication settings of the controller are applied only <br> then. These are the communication settings that were configured in the source <br> project in the device editor on the "Communication Settings" tab. |
| "Manual configuration" | O: More configuration settings are displayed. <br> See "Manual configuration" below. |

Settings for 'Se- The communication setting is done only manually.
lect the project type' == 'Other See "Manual configuration" below.
Project'
Manual configu-
ration

| "Dynamic from variable" | O: The communication parameters are configured at application runtime by means of an IEC variable of data type DatasourceAppV3. ConnectionSetup. <br> …: Opens the input for selecting the IEC variables for a dynamic configuration. <br> The data type DatasourceAppV3. ConnectionSetup (STRUCT) is defined in the Datasource ApplicationV3 Access library. For the configuration, the structure member xDataValid first has to be set to FALSE. If the address data has been specified, then xDataValid has to be set back to TRUE. <br> Use case: The communication parameters are not available yet when a project is being created. |
| :---: | :---: |
| "Use device address" | O: The communication is done via the address specified here. <br> Example: 0101 <br> Hint: Click "From device" for an automatic address setting. |
| "From device" | The data of the currently connected data source device is read automatically and specified in "Use device address". The address corresponds to the setting of the device in the device editor in "Communication Settings". |
| "Search for the target device using the network scan" | The data source manager starts the network scan for devices in the network. The scan is successfully when controllers are found whose communication settings match the selected search criteria. The result is displayed in the input fields. |
| "Node name" | Search for the specified node name Example: WST0 6 |
| "Target type" | Search for the specified target type Example. 4096 |
| "Target ID" | Search for the specified target ID Example: 00000001 |
| "Target version" | Search for the specified target version Example: 1.0.0.0 |


| "Network location" | - "Direct child of the data sources PLC": The scanned remote PLC has an address that is running with the address of the local controller (of the data source manager). <br> Example: Data sources PLC: 0000.0001 ; remote source PLC: 0000.0001 .0001 <br> - "Direct child of node with address": Specify the address of the parent node <br> - "Direct child of the data source PLC or of the node with address": Combination of both options above. |
| :---: | :---: |
| "Search type" | - "First found device": The first controller in the device tree is selected that fulfills the specified criteria. <br> - "Exactly found device": The controller is selected that fulfills the specified criteria exactly. <br> Note: The data source manager waits until the network scan is complete. This usually takes about 10 seconds. |

## Communication settings for controllers with visualization user management

| "Login Configuration" | If a visualization user management is configured on the remote device, then <br> valid credentials are required at login. |
| :--- | :--- |
| "Type" | Defines how the visualization user management gets credentials <br> - "Login using the following credentials" <br> The credentials are hard-coded into the "User name" and "Password" set- <br> tings. They are used each time a connection is attempted. <br> - "Login using the credentials determined at runtime" <br> At runtime, a dialog opens and prompts the user to specify a user name and <br> password. Hard-coded credentias, which have nonetheless been specified <br> in "User name" and "Password", are ignored. |
| "User name" | Example: max. smith |
| "Password" | Example: ••• |

## Specific set- <br> tings of the <br> communication <br> buffer

| "Advanced" | : The subsequent settings are changed. |
| :--- | :--- |
| "Default communication buffer <br> size" | Default setting: 50000 |

See also

- $y^{2}$ Chapter 6.4.1.10.6.2 "Initially Adding a Data Source" on page 1949
- ${ }^{\text {• }}$ Chapter 6.4.1.21.2.5 "Object 'Data Source Manager"" on page 2412


## Tab 'Communication' via OPC UA Server

## Where to find the server layout

"Browse Live Server:"
The client connects to the server and detects the existing variables and types.

| From Information Model | The client reads the data structure (layout) of the OPC UA Server from the information model set here and as a result receives the information about available variables and types. A connection to the server is not required. <br> The list box contains the information models installed in the OPC UA Information Model Repository. <br> ${ }^{4}$ Chapter 6.4.1.21.3.9.12 "Command 'OPC UA Information Model Repository"' on page 2665 |
| :---: | :---: |
| Read Connection Settings from IEC Variable | The connection settings used by the device are not read here from the dialog, but at runtime from the IEC variable specified here. <br> For this possibility, see: Using a Dynamic Connection to an OPC UA Server * Chapter 6.4.1.10.6.7 "Data Source OPC UA Client" on page 1960 |
| Connection Type |  |
| "Server URI" | URI of the OPC UA Server; editable |
| "Show All Endpoints" | The button opens the "Available Endpoints" dialog with the end points available on the OPC UA Server. An "endpoint" is a combination of security settings. <br> The lower part of the dialog displays the available user tokens: "Policyld, TokenType", and "SecurityPolice". <br> For the selected end point, clicking "Apply" will apply the security settings "SecurityMode" and "SecurityPolicy" on the "Communication" tab in the "Security" section. |
| Security |  |
| "Messages Security Mode" | Type of encryption <br> - "None": No encryption and no signing <br> Note: if you select this option, there can be no guarantee who receives the data. Therefore, "None" should be used exclusively in closed networks. <br> - "Sign and Encrypt": The transferred data will be signed and encrypted. Signing makes sure that the data is not manipulated and the receiver is correct. <br> - "Sign": The transferred data will be signed. <br> Signing and encryption work only for certificates. |
| "Security Policy" | List box for the encryption method to be used: <br> - Basic256sha256 <br> Requirement: Either "Sign and Encrypt" or "Sign" was selected for "Messages Security Mode". |
| "Client certificate" | Here you can specify a certificate for an encrypted signed access to the OPC UA Server for the purpose of browsing. <br> Note: The access which is safeguarded in this way is ONLY for browsing the server (see above, "Browse Live Server" option). For encrypted data exchange between the server and client in online mode, a separate certificate is required, which can be created in the security screen. For more information see: OPC UA Server, Creating a certificate of the CODESYS OPC UA Server. <br> In case a certificate is not available for selection, you can click the $\mathbb{R}_{\boxed{*}}$ button to open the "Create Self-Signed Certificate "dialog. Here you define a password for your private key and a file name for the certificate. When you click [OK], the certificate files.cer and. pfx are created in the project directory and the certificate is automatically registered in the certificate manager. |


| "Authentication" |  |
| :--- | :--- |
|  | Access without authentication. Permission for this can be granted for com- <br> patibility reasons in the "Change Communication Policy" dialog on the <br> "Communication Settings" tab of the device editor. |
| "User name" | Access to the OPC UA Server requires the entry of a user name and password. <br> You can programmatically set up how the transfer of this authentication data <br> should be done: <br> To help with the creation of a POU for this purpose in the project, you can <br> click the "Create Authentification Provider" button to create a program template <br> Datasource_OPCUA_Initializer and program it accordingly. |

- ③ Chapter 6.4.1.21.2.5 "Object 'Data Source Manager'" on page 2412


## Tab 'General and Diagnosis'

The "General and Diagnosis" tab provides information about the status of the data source communication.

| "Update Configuration" |  |
| :--- | :--- |
| "Update rate (ms)" | Example: 200 |


| "Connection Information" |  |
| :--- | :--- |
| "Connection status" | Example: online |
| "Error information" | Example: oK |

See also

- \& Chapter 6.4.1.10.6.5 " Updating data interfaces" on page 1957


## Object 'DeviceTrace'

Symbol:
A "DeviceTrace" object shows trace data in one or more diagrams, as does a "Trace" object.
The difference is that a "DeviceTrace" directly accesses traces that are running on the controller. The object is inserted below the device in the device tree. Therefore there is no immediate dependency on the applications in the CODESYS project.

You can use the DeviceTrace for visualizing the processor load of a multicore controller.

For more information about the editor and its operation, refer to the help page for the "Trace" object.
See also

- ${ }^{*}$ Chapter 6.4.1.21.2.31 "Object 'Trace"' on page 2541
- « "Runtime system component CmpTraceMgr, "Trace manager"" on page 2008
- \& Chapter 6.4.1.13.3.5 "Accessing All Traces on the Controller" on page 2014


## Object 'Device' and Generic Device Editor

## General

Symbol:
A device object represents a type of hardware; examples: control device, fieldbus node, bus node, drive, I/O module, monitor. The arrangement of the device objects in the device tree, that is the view "Devices" in CODESYS, maps the hardware structure. In the device object configuration editors inter alia you connect the controller I/Os with project variables.
Use command "Add Device" or "Insert Device" to insert a device object in the device tree. Depending on the insert position CODESYS always offers the currently matching devices.
A double-click on a device object in the device tree opens the associated device editor. The editor provides generic and device-specific tabs for the device configuration.

See also

- $\quad$ Chapter 6.4.1.21.2.8.2 "Generic device editor" on page 2427


## Generic device editor

The generic device editor contains tabs for the configuration of a PLC device in CODESYS. Additionally there are device-specific tabs, so that the configuration editor consists of many different dialogs, depending on the device.
The editor opens after a double-click the device object in the device tree ("Devices" view).
You can make general settings for a device editor in the CODESYS "Options" in the "Device Editor" category. For example, you can show and hide the tabs of the generic device editor.
A device editor is given the name of the device. The following tabs of the generic device editor can be included:

- "Communication": Configuration of the connection between the development system and a programmable device (PLC). Not available in the case of pure I/O devices.
- "Applications": List of the applications on the controller.
- "<device> Parameters": Display and configuration of device parameters.
- "Files": Configuration of the file transfers between a host file system and the device.
- "Log": Display of the PLC log file.
- "PLC Settings": Configuration of the handling of the I/Os: which application, behavior in the stop state, updating, bus cycle options, etc.
- "PLC Shell": Text-based control monitor for interrogating certain information from the controller.
- "Users and Groups": User management with regard to the device at runtime.
- "Access Rights": Rights for access to objects and files on the device.
- "Symbol Rights": Access rights of individual user groups to symbols (symbol sets) on the device.
- "Task List": Overview of all inputs and outputs, which are assigned to tasks - useful for troubleshooting.
- "Status": Device-specific status and diagnostic messages.
- "Information": General information about the device (name, vendor, version etc.)

See also

- ${ }^{\text {n }}$ Chapter 6.4.1.8 "Configuring I/O Links" on page 1835


## Tab 'Communication Settings'

On this tab of the generic device editor, you define the connection between CODESYS and the device on which your application(s) should run. This includes security settings such as encrypted communication and signing.

If you prefer the classic mode of display for the dialog, then select it in the CODESYS "Options" in the "Device editor" category).

You select a gateway and a target device from the list boxes. The possible selections depend on the entries in the "Manage Gateways" and "Manage Favorite Devices" dialogs (see the "Gateway" menu).
You can also specify the target directly with the IP address (example: "192.168.101.109"), device address (example: "[056D]"), or device name (example: "MyDevice"). After the device is entered, CODESYS searches for the device in the network of the gateway.

The option of searching by device name requires unique device names in the network.

The solid circle on the lower right corner of the gateway symbol provides information about the connection status:

- Red: CODESYS cannot establish the connection.
- Green: The connection is established.
- Black: The connection status is unknown.

Some communication protocols allow regular checking of the gateway so that the status cannot be displayed.

Clicking the solid circle of the target device starts a network scan for the device. This works only if the network is not already being scanned.

| "Scan Network" | This button opens the "Select Device" dialog. It lists all configured gateways with the associated devices. You can select one target device from this list. If the name of the selected device is unique, then the name will be used in the connection settings. Otherwise, the unique device address is applied. <br> For details about this dialog, see the description of the classic view below. <br> Option: Hide non-matching devices, filter by Target ID: The display is limited on the devices that have the same target ID as the current device configured in the project. $\square$ : All available devices in the network are displayed. Double-clicking the entry of a device, which has a different target system ID than the device configured in the project, opens a dialog box. This dialog box allows you to update the device description of the device in the project to the one matching the selected device. Note: This kind of update is possible only with devices which have different IDs, not for those which only have different versions. The update also requires that the device description is already installed in the repository. |
| :---: | :---: |
| "Gateway " | This menu includes the following commands: <br> - "Add New Gateway": Opens the "Gateway" dialog for defining a new gateway channel. <br> - "Manage Gateways": Opens the "Manage Gateways" dialog with an overview of all gateways. You can add or delete entries here or change their order. <br> - "Configure Local Gateway": Opens the "Gateway Configuration" dialog. You can configure the block drivers for the local gateway. |
| "Device" | This menu includes the following commands: <br> - Options: <br> - "Add Current Device to Favorites": Adds the currently set device to the list of favorite devices. <br> - "Manage Favorite Devices": Opens the favorites dialog with a list of all preferred devices. In this dialog, you can add or delete entries or change their order. The top device is the default. <br> - "Confirmed Online Mode": <br> For security reasons, CODESYS requires you to confirm the following when calling the following online commands: Force Values, Write Values, Multiple Loading, Release Force List, Single Cycle, Start, Stop. <br> - "Store Communication Settings in Project": <br> CODESYS saves the communication settings in the project for reuse on the same computer. <br> Note: If you use the project on another computer, then you have to reset the active path. <br> $\square$ : CODESYS saves the communication settings in the options of the local installation for reuse on the same computer. <br> Note: When using CODESYS SVN, the option should be cleared in order to prevent blocking the device object. <br> - "Rename Active Device": <br> Opens a dialog for changing the device name. <br> - "Wink Active Device": Devices that support this function react by blinking. <br> - "Send Echo Service": CODESYS sends five echo services to the controller. These are used to test the network connection, similar to the ping function. The services are sent first without a payload and then with a payload. The scope of the payload depends on the communication buffer of the PLC. A message view opens with information about the average echo service delay and the scope of the sent payload. <br> - "Encrypted Communication": <br> $\checkmark$ : The communication to this controller is encrypted. A certificate of the controller is required in order to log in to the controller. If the certificate is not available, then an error message opens prompting whether or not the certificate should be displayed and installed. |


|  | If the "Enforce encrypted communication" option is selected as "Security level" in the "Security Screen" view, then the "Encrypted Communication" command is disabled here. <br> - "Change Runtime Security Policy" <br> Opens the "Change Runtime Security Policy" dialog for changing the device setting for the encryption of communication. <br> - "Change Runtime Password Policy" <br> Opens the "Change Runtime Password Policy" dialog to change the settings for the password policy and the login lock <br> - "Security Settings": This command is available only if CODESYS Security Agent >= 1.3.0.0 is installed. It opens the "Device Security Settings" dialog. The current security settings on the connected device are displayed. You can change the settings in the "Value" column and click "OK" to write them to the device. <br> - "Filter Network Scans by Target ID": <br> . The display is limited on the devices that have the same target ID as the current device configured in the project. |
| :---: | :---: |

Table 402: Dialog "Change Runtime Security Policy"

| When you select a new communication policy in this dialog, the configuration in the runtime system is changed. |  |
| :---: | :---: |
| "Communication Settings" |  |
| "Current policy" | Shows the currently selected policy for the encryption of communication with the device |
| "New policy" | List box for the new policy for encryption <br> - "No encryption": The controller does not support encrypted communication. <br> - "Optional encryption": The controller supports encrypted and unencrypted communication. <br> - "Enforced encryption": The controller supports encrypted communication only. |
| "Code Signing" |  |
| "Current policy" | Display of the code signing policy which is currently set in the runtime system |
| "New policy" | - "All": All types of application code are accepted. <br> - "Enforced signing": Only signed application code is accepted (preventing loading an application from untrusted sources). |
| "Device User Management" |  |
| "Current policy" | Shows the currently selected policy for user management |
| "New policy" | - "Optional user management": It is the responsibility of the user to enable user management on the device or leave the device unprotected. <br> - "Enforced user management": The user management on the device is enabled and cannot be disabled by the user. |
| "Allow anonymous login" | Specific registered components (for example, OPC UA) can connect to the controller without the providing any credentials. Even if anonymous access to the OPC UA is permitted, the created device user management for the controller remains active. |

Table 403: Dialog "Change Runtime Password Policy"
The runtime system password policy configured here is already checked in advance in the programming system when the password for a new device user is entered or when the existing password of a device user is changed.

| "Password policy is active" | The "Password settings" can be changed and are used when a password for the device user management is created. <br> "Password settings" <br> - "Minimum length" <br> Default setting: 8 <br> - "Number of unique characters" <br> Default setting: 4 <br> - "Requires lowercase letter" <br> Default setting: enabled <br> - "Requires uppercase letter" <br> Default setting: enabled <br> - "Requires digit" <br> Default setting: enabled <br> - "Requires special character" <br> Default setting: enabled <br> - "Must not contain username" <br> Default setting: enabled <br> The following rules for the password always apply, even when the rules are partially or completely disabled: <br> - The password must not be blank. <br> - The password must not be the same as the user name. |
| :---: | :---: |
| "Login lock is active" | The "Login lock settings" are applied at login. <br> "Login lock settings" <br> - "Scope" <br> - "ADMINS": The settings apply to users who have administrator permissions. <br> - "NOADMINS": The settings apply to all users who do not have administrator permissions. <br> - "ALL": The settings apply to all users. <br> - "Maximal Retries": When the number of login attempts specified here is exceeded, the user will be locked out for the amount of time which is specified in the "Lock duration" field. The user cannot log in again until the lock is removed by an administrator or the lockout time has expired. <br> - "Lock duration [s]": Lock time (in seconds) <br> For information about unlocking a locked user, see: \# Chapter 6.4.1.11.3 <br> "Encrypting Communication, Changing Security Settings" on page 1967 |

Communication In the CODESYS options, you can activate the classic mode of the dialog ("Tools $\rightarrow$ Options", Settings "Device Editor" category).
Classic Mode

| "Select the network path to the <br> controller" | Gateway channel for the connection. <br> Select the channel from the lower part of the view. |
| :--- | :--- |

Table 404: "View displaying configured gateway channels and network devices"

| Left side of view | Tree structure of the configured gateway channels with the connected devices in the local network: <br> Note: CODESYS saves these entries on the local system, not in the project. <br> The device entries are preceded by a device symbol (1). Entries with a target ID that are different from those currently configured in the project are displayed in gray. <br> Click "Scan Network" to refresh the list. <br> Note: If you created the first project on the local system, then the local gateway is listed as an entry in the tree by default. CODESYS starts this gateway automatically on system boot. <br> The solid circle on the lower right corner of the gateway symbol provides information about the connection status: <br> - Red: CODESYS Development System cannot establish the connection. <br> - Green: The connection is established: $\square$ <br> - Black: The connection status is unknown. <br> Note: Some communication protocols allow regular checking of the gateway so that the status cannot be displayed. <br> Each of the device entries in the tree consists of a symbol followed by the "Device name"> ["Device address"]. On the right side of the view, you also see the "Target ID", "Target Name", "Target Type, Target Vendor", and "Target Version". |
| :---: | :---: |
| Right side of view | Information about the gateway channel of device selected on the left side of the view. <br> When a gateway channel is selected in the left view, the following information is displayed: "Device name", "IP address", "Port", "Driver" <br> When a device is selected in the left view, the following information is displayed (depending of the device): "Device name", "Device address", "Number of channels", "Block driver", "Serial number", "Encrypted communication", "Target vendor", "Target ID", "Target name", "Target type"," Target version". |

Table 405: "Filter and sorting functions on the right side of the dialog"

| "Filter" | You can reduce the displayed list of devices that have the same "Target ID" as <br> the current device configured in the project. |
| :--- | :--- |
| "Sorting order" | You can sort the list by "Name" or "Device Address" in alphabetical or ascending <br> order. |

Table 406: Command buttons on the right side of the dialog

| "Set Active Path" | The command sets the selected communications channel as active. Double- <br> clicking the entry in the channel tree achieves the same result. |
| :--- | :--- |
| "Add Gateway" | The command opens the "Gateway" dialog where you can define a gateway that <br> CODESYS should add to the current configuration. |
| "Add Device" | The command opens the "Add Device" dialog. Here you can manually define a <br> device that is to be inserted under the gateway entry currently selected in the <br> tree. Note the functionality of "Scan Network" as well. |
| "Scan Network" | The command starts a search for available devices in the local network. The <br> configuration tree of the gateway is refreshed accordingly. |

Table 407: "Commands in the context menu of the gateway tree and device tree in the dialog"

| "Scan for Device by Address" | The command searches the network for devices with a unique address as given <br> in the configuration tree. CODESYS displays the detected devices with the given <br> address below the gateway. The search always applies to the devices below the <br> selected gateway or below the selected entry. |
| :--- | :--- |
| "Scan for Device by Name" | The command searches the network for devices with the same name as given in <br> the configuration tree. Capitalization is ignored. CODESYS displays the detected <br> devices below the gateway with the given name together with its unique device <br> address. The search always applies to the devices below the selected gateway <br> or below the selected entry. |
| "Scan for Device by IP | The command searches the network for devices with a unique IP address as <br> given in the configuration tree. CODESYS displays the detected devices with <br> the given address below the gateway together with its name. The search always <br> applies to the devices below the selected gateway or below the selected entry. |
| "Send Echo Service" | CODESYS sends five echo services to the controller. These are used to test <br> the network connection, similar to the ping function. The services are sent first <br> without a payload and then with a payload. The scope of the payload depends <br> on the communication buffer of the PLC. A message view opens with information <br> about the average echo service delay and the scope of the sent payload. |
| "Delete Selected Device" | The command deletes the selected device from the channel tree. |
| "Edit Gateway" | The command opens the "Gateway" dialog for editing the settings for the <br> selected gateway. |
| "Configure the Local Gateway" | The command opens a dialog for configuring a local gateway. This provides an <br> alternative to manually editing the Gateway. cfg file. |

Table 408: Options in the lower part of the dialog

| "Don't store communication settings in project" | - $\square$ : CODESYS saves the communication settings in the options of the local installation for reuse on the same computer. <br> Note: When using CODESYS SVN, the option should be selected in order to prevent blocking the device object. <br> - $\square$ : CODESYS saves the communication settings in the project for reuse on the same computer. <br> Note: If you use the project on another computer, then you have to reset the active path. |
| :---: | :---: |
| "Confirmed Online Mode" | For security reasons, CODESYS requires you to confirm the following when calling the following online commands: "Force Values", "Write Values", "Multiple Loading", "Release Force List", "Single Cycle", "Start", "Stop". |

## Tab 'Parameters'



The device-specific parameters are displayed in a table on this tab of the generic device editor. The device description defines which parameters you can edit in this dialog.

You can sort the entries in alphabetically ascending or descending order or in the default order by clicking the column header.

| "Parameter" | Parameter name, not editable |
| :--- | :--- |
| "Type" | Data type of the parameter, not editable |


| "Value" | Initially displays the default value of the parameter, directly or the corresponding <br> symbol name. Non-editable parameters are displayed in light-gray. If the param- <br> eter is editable you can open an input field, a drop-down list or a file selection <br> dialog with a double-click the table field and use it to change the value. |
| :--- | :--- |
| "Default value" | Default value of the parameter defined by the device description, not editable |
| "Unit" | Unit of measure for the value (example: "ms" for milliseconds; not editable) |
| "Description" | Short description of the parameter specified by the device description, not edit- <br> able |

See also

- $\quad$ ² Chapter 6.4.1.21.2.8.2 "Generic device editor" on page 2427


## Tab 'Applications'

On this tab of the generic device editor you can see which applications exist on the device. Depending on the system you can delete the applications from the device or retrieve detailed information about the application.

| "Applications on the PLC" | List of the applications found via "Refresh list" during the last scan of the control <br> device. |
| :--- | :--- |
| "Delete" <br> "Delete All" | Deletes the application selected in the list or all listed applications on the con- <br> troller <br> Note: If a safety controller is inserted below a PLC, then this command can <br> permanently interrupt the communication links of the safety controller to other <br> safety controllers (via safety network variables), to field devices, and to the <br> development system. The safe field devices and the other safety controller can <br> enter the safe state as a reaction. The connection to the development system <br> is affected only in the case of a safety controller that is connected to the main <br> controller via a fieldbus. For more information, refer to the section "Subordinate <br> Safety Controller". |
| "Details" "Contents" | Opens the dialog box "Details". It displays information defined for the application <br> on the "Information" tab of the dialog box "Properties". |
|  | Requirement: The "Download the application info" option is activated in the <br> "Properties" of the application object on the "Application generation options" tab. <br> This causes information about the contents of the application to be additionally <br> loaded to the PLC. |
| The "Contents" button opens a dialog box with additional information about the <br> differences between the latest generated code and the application code that <br> exists on the controller. The different modules are displayed in a comparison <br> view. |  |
| The controller is scanned for applications and the list is refreshed accordingly |  | | "Refresh List" |
| :--- |

You can configure the commands "Remove Application from Device" and
"Remove Applications from Device" by means of the dialog box form "Tools
$\rightarrow$ Customize". These commands correspond to the "Delete" and "Delete All" buttons.

## See also

- \& Chapter 6.4.1.21.2.8 "Object 'Device' and Generic Device Editor" on page 2427
- ${ }^{*}$ Chapter 6.4.1.21.2.2 "Object 'Application"" on page 2410
- $\Downarrow_{\text {Cl }}$ Chapter 6.4.1.21.2.8.2 "Generic device editor" on page 2427
- ${ }^{4}$ Chapter 6.4.1.21.2.8.20 "Tab 'Information'" on page 2460
- ${ }^{4}$ Chapter 6.4.1.21.4.11.10 "Dialog 'Properties - Application Build Options"' on page 2759
- $\rightleftarrows^{*}$ Chapter 6.4.1.21.3.7.2 "Command 'Login'" on page 2624
- $\sum^{\star}$ Chapter 6.4.1.10.7 "Subordinate safety controller" on page 1964


## Tab 'Backup and Restore'

In this tabbed page of the generic device editor, you can backup and restore the applicationspecific file on the PLC by saving and reading a zip archive.
Requirement: The communication settings are correct for connection to the device. The application for backup is available on the PLC.

Table 409: Menu Bar

| "Backup" | This button opens a menu with the following commands: <br> - « "Read Backup Information from Device": This command searches for application-specific files from the \$PlcLogic\$ directory of the PLC and lists them in a table in the lower part of the tabbed page. <br> - "Create Backup File and Save to Disk": <br> Requirement: The "Read Backup Information from Device" command was used for determining the backup-related files. These files are located in the table in the lower part of the tabbed page. <br> This command compresses the files in the table set as "Active" and the meta. info information file into a backup zip file. The file extension is tbf (="Target Backup File"). <br> - "Save Backup File to Device": <br> Requirement: The backup file has been saved to the disk. This command saves the backup file to the TBF directory of the PLC. |
| :---: | :---: |
| "Restore" | This button opens a menu with the following commands: <br> - "Load Backup File from Disk": This command opens the "Open" dialog box for navigating the file system for a saved backup file. The included files are listed in a table in the lower part of the tabbed page. <br> - 著 "Load Backup File from Device": This command generates a list of all backup files found on the PLC. Select one of these files to view its contents in a table on the tabbed page. For the restore operation, you can deactivate optional components and edit the comments. <br> - * "Restore on Device": This command is available if at least one component of the backup file that is currently loaded in the tabbed page is set to active. It prompts for restoring the application status on the device. The user interface is blocked during restore. You can cancel the operation. |

Table 410: "Target Information"

| "ID" | ID of the PLC (example: 0000 0001) |
| :--- | :--- |
| "Type" | Device type (example: 4096) |
| "Version" | Device version (example: 3.5 .8 .0 ) |

Table 411: "Backup Information"

| "File name" | Storage path of the backup file. Clicking the symbol (莀) opens the file system <br> dialog box. Example: PlcLogic\$/Application/Application.crc |
| :--- | :--- |
| "Size of active files" | (in kilobytes) Total size of the files set as active in the table (example: 206 KB <br> $(210965$ bytes) ). |


| "Mode" | Defines the scope of the backup: "Application". The application-related files are <br> added to the archive. |
| :--- | :--- |
| "Comment" | Optional entry for comments to be saved in the meta.info file of the backup <br> and reading when the files are restored. |

Table 412: Table of files for backup

| "Active" | R: Optional files can be deactivated here for exclusion in the backup file. <br> Required components are shown here with a green check mark (no check box). |
| :--- | :--- |
| "Component" | Affected components (example: file system) |
| "File" | Name of the component file to back up (example: \$PlcLogic\$/ <br> Application/Application.app) |
| "Size" | File size in bytes (example: 43280) |
| "Requires STOP" | A $:$ For components, the application must be stopped before backup and restore. <br> A pill open to warn you of any backup or restore conflicts. |

See also

- $y^{*}$ Chapter 6.4.1.13.9 "Backup and restore" on page 2024


## Tab 'Synchronized Files'

The tab of the generic device editor lists the files that are downloaded to the PLC when the application is downloaded. For example, these are external files that were added to an application.
Implicit files, such as the source code archive file, are displayed here only if their time of download is configured for this and the "Show implicit files for application download on the editor of a PLC" option is activated in the CODESYS options ("Device Editor" category).

| "Refresh" | Refreshes the view |
| :--- | :--- |
| "Download 'on-demand' files" | For internal use only. |
| "File Name" | Name of the file below the application, or direct name of the implicitly transferred <br> file (example: archive.prj). <br> Double-click the file name to open the file. |
| "Host Path" | Location or original location of the file (example: D: $\backslash$ Proj $1 \backslash$ Files). <br> Double-click the path to open the directory in the file explorer. |
| "Timing" | Time interval of the file update on the PLC (example: "After application <br> download/online change"). |
| "Information" | Object-dependent additional information (example: "Object: External File"). |
| "Provider" | General origin type of the file (example: "External File Objects", "Source code <br> download provider"). |

See also

- $\Rightarrow$ Chapter 6.4.1.21.4.14.7 "Dialog 'Options' - 'Device Editor"' on page 2786
- $\boldsymbol{y y}^{\boldsymbol{y}}$ Chapter 6.4.1.21.2.10 "Object 'External File"" on page 2464


## Tab 'Files'

In this tab of the generic device editor, you can transfer files between CODESYS (host) and the PLC. If the communication settings are correct and the PLC is online, then CODESYS establishes the connection automatically to the PLC for the duration of the file transfer.

Table 413: "Host" / "Runtime"

|  | Access to the file system of the host with the functionalities of a standard file manager |
| :---: | :---: |
| "Location" | Current directory for the file transfer on the host side |
| $\square$ | Opens a dialog to create a new directory in the set path |
| -"" | Deletes the selected files or directories |
| 6 | Updates the list of files and directories there for the set location |
| «< $\square$ $\gg$ | Copies the selected files and directories to the respective other file system from the host and runtime system <br> If a file is not already available in the target directory, then it is created. If it is already available and not write-protected, then it is overwritten. Then a corresponding message is displayed. $\square$ corresponds to the "Write File to Controller" command. corresponds to the "Write File from Controller" command. |

By default, the "Write File to Controller" and "Write File from Controller" commands are not included in any menu. You can add it to a menu by means of the "Tools $\rightarrow$ Customize" dialog, in the "Online" command category.

See also

- ${ }^{4}$ Chapter 6.4.1.21.2.8.2 "Generic device editor" on page 2427
- « Chapter 6.4.1.15 "Copying files to/from PLC" on page 2027


## Tab 'Log'

You can view the PLC log on this tab of the generic device editor. It lists the events that were recorded on the target system. This concerns the following:

- Events during the startup and shutdown of the system (components loaded, with version)
- Application download and loading of the boot application
- Custom entries
- Log entries from I/O drivers
- Log entries from data sources


Table 414: Menu bar


Refreshes the list of log events for all runtime system component

| "Components" | Filters the display of log events by the runtime system components selected in <br> the list box <br> Example: <br> CmpApp displays all events which occur in these components, for example <br> "Application [ <name> ] loaded via [OnlineChange] ". <br> "<all components>": Displays the reported events of all components |
| :--- | :--- |
| Loads the next page with newer log messages |  |

Table 415: Display window with log file
Tabular display of the log messages
Ten thousand log messages are displayed per page.

| "Severity" | $\bullet$ | ©: Warning |
| :--- | :--- | :--- |
| $\bullet$ | ©: Error |  |
| $\bullet$ | E: Exception |  |
| $\bullet$ | ©: Information |  |
| $\bullet \bullet$ | D: Debug message |  |


| "Time Stamp" | Date and time of the development system or of the runtime system) <br> Example: $01 / 12 / 07 \quad 09: 48$ |
| :--- | :--- |
| "Description" | Description of the event <br> Example: PLC started |
| "Component" | Runtime component where the reported event occurred |

Table 416: Status bar

| "Auto-Scroll" | Displays whether auto-scrolling is enabled ("ON") or disabled ("OFF") <br> Hint: Click the 歪 button to enable "Auto-Scroll". <br> - "ON": The log list is refreshed automatically when changes occur. <br> - "OFF". When a new log event occurs, it is displayed next to "Off". Moreover, the 歪 button is decorated on the menu bar: ${ }^{\boldsymbol{\pi}}$. |
| :---: | :---: |

## Note for error checking

For exceptions with the description *SOURCEPOSITION*, the affected function opens in the editor by double-clicking it or by means of the "Display Source Code in Editor" command in the context menu. The cursor jumps to the line that is causing the error. You can also perform this diagnosis when you have the CODESYS project archive, including the download information files and the exported log file. When the affected function is protected, then the following message appears: "The source code is not available for <function name>".

If a VendorException is reported, then a manufacturer-specific exception error has occurred in the CODESYS runtime. Contact the PLC manufacturer for more information.

## See also

- $\quad 4$ Chapter 6.4.1.21.2.8.2 "Generic device editor" on page 2427
- ${ }^{*}$ Chapter 6.4.1.21.2.8 "Object 'Device' and Generic Device Editor" on page 2427
- $\xi^{*}$ Chapter 6.4.1.13.6 "Reading the PLC log" on page 2021
- ${ }^{\Perp}$ Chapter 6.4.1.21.4.15.2 "Dialog 'Customize' - 'Menu'" on page 2801


## Tab 'PLC Settings'

On this tab of the generic device editor you make the basic settings for the configuration of the PLC, for example the handling of inputs and outputs and the bus cycle task.
"Application for I/O handling" $\quad$ Application that is responsible for the I/O handling.

Table 417: "PLC Settings"

| "Refresh I/Os in stop" | V: The values of the input and output channels are also refreshed when the <br> PLC is in STOP mode. If the watchdog detects a malfunction, the outputs are set <br> to the predefined default values. <br> $\square$ |
| :--- | :--- |
| refreshed. The values of the input and output channels in STOP mode are not |  |
| "Behavior of the outputs at |  |
| stop" |  |$\quad$| Handling of the output channels when the controller enters the stop state: |
| :--- |
| - "Retain values": The current values are retained. |
| - "All outputs to default value": The default values resulting from the I/O map- |
| ping are assigned. |
| - "Execute program": You can control the handling of the output values via |
| a program contained in the project, which CODESYS executes at "STOP". |
| Enter the name of the program in the field on the right. |

Table 418: "Bus Cycle Options"

| "Bus cycle task" | Task that controls the bus cycle. By default the task defined by the device <br> description is entered. <br> By default the bus cycle setting of the superordinate bus device (use cycle set- <br> tings of the superordinate bus) applies, i.e. the device tree is scanned upwards <br> for the next valid bus cycle task definition. |
| :--- | :--- |

## NOTICE!

Before you select the "<unspecified>" setting for the bus cycle task, you should be aware that "<unspecified>" means that the default setting given in the device description goes into effects. You should therefore check this description. Use of the task with the shortest cycle time may be defined as the default there, but use of the task with the longest cycle time could equally well be defined!

## NOTICE!

For fieldbuses, a fixed cycle matrix is necessary to assure a determined behavior. Therefore, do not use the type 'freewheeling' for a bus cycle task.

Table 419: "Additional Settings"

| "Force variables for the I/O <br> mapping " | This setting is available only if it is supported by the device. <br> 目: When compiling the application CODESYS creates two global variables for <br> each I/O channel that is mapped to a variable in the dialog "I/O Mapping". You <br> can use these variables for the forcing of the input or output value on this <br> channel, for example via an HMI visualization. |
| :--- | :--- |
| "Activate diagnostics for |  |
| devices" | V: The CAA Device Diagnosis library is integrated in the project. An implicit <br> function block is generated for each device. If there is already a function block <br> for the device, then either an extended function block is generated (example: <br> EtherCAT) or another function block instance is added. This then contains a <br> general implementation of the device diagnostics. <br> By means of the function block instances you can determine the status of all <br> devices in the application and evaluate errors. In addition, the library contains <br> functions for the programmatic editing of the device tree. Example: Scanning of <br> all children of a bus system, jumping to the parent element. |
| "Create additional parameters" | This setting is available only if it is supported by the device. <br> Create additional parameters. |
| "Display I/O warnings as | Warnings concerning the I/O configuration are displayed as errors. |
| errors" |  |
| "Enable symbolic access for |  |
| I/Os" | V: Input and output variables (VAR_INPUT and VAR_OUTPUT) are automati- <br> cally created for the I/O channels of the device. For this purpose, an extended <br> function block is created for each slave. The basis is the existing function <br> block of the slave. This kind of automatically generated function block can be <br> accessed directly in the application code. <br> This symbolic access works in parallel to the I/O mapping which was set up <br> manually. <br> $\square: ~ A c c e s s ~ t o ~ I / O ~ c h a n n e l s ~ i s ~ n o t ~ s u p p o r t e d . ~ I t ~ i s ~ n e c e s s a r y ~ t o ~ m a n u a l l y ~ s e t ~ u p ~ a ~$ |
| mapping and specifically assign new or existing variables per I/O channel. |  |

## Tab 'PLC Shell'

This tab of the generic device editor includes a text-based control monitor for querying specific information from the controller. You can specify device-dependent commands for this and receive the response from the controller in a result window.

Table 420: ABB AG standard commands

| Command with Possible Parameters | Description |
| :--- | :--- |
| $?$ | List of available PLC shell commands with possible parameters and <br> short description |
| applist | Provides a list of all loaded applications <br> The order in the list defines the application index beginning with 0. |
| batt [INFO, ACK] | Get battery status; batt INFO: displays detailed information; <br> batt ACK: acknowledge fail state |
| cdump [exp,sdc,rst] | handle logging data in case of crash or malfunction; cdump ? for <br> help |


| Command with Possible Parameters | Description |
| :---: | :---: |
| cert-createcsr [<number for search result by "certgetapplist">] | Generates CSR files for all applications |
| cert-export <trust level> [<number for search result by "cert-getcertlist">] | Exports the specified certificate |
| cert-genselfsigned [<number for search result by "certgetapplist"> <expdays=>] | Generates self-signed certificates <br> The validity period of the certificate can be specified by means of expdays=. <br> Default value: 365 days |
| ```cert-gendhparams [length in bits]``` | Generates the parameters for the Diffie-Hellman key exchange Caution: This operation can take several minutes to complete. |
| cert-getapplist | Provides all registered and used certificates (ID of the component and usage). |
| cert-getcertlist [<trust level>] | Lists all certificates of the specified trust level If a trust level is not given, then all certificates are listed. <br> Possible trust levels <br> - untrusted: Untrustworthy certificates <br> - trusted: Trustworthy certificates <br> - own: Certificate of the controller <br> - quarantine: Certificates whose trust level (trusted, untrusted) cannot be determined by validation. Incoming connections were therefore denied. |
| cert-import <trust level> <file name.cer> | Imports the specified certificate |
| cert-remove <trust level> <number for search result by "cert-getcertlist" or "all"> | Removes the specified certificate |
| channelinfo | Provides information about the communication channel |
| clearsram <segment number retrieved by "getsramlayout" or "all"> | Init and mark SRAM segment or complete SRAM as free |
| coupler desc | Get version information of communication modules |
| com settings | Get information of serial COM ports |
| cpuload | Shows the processor load of the CPU (for multicore, each processor core) |
| date [yyyy-mm-dd] | Get/set system date; date: get system date; date yyyy-mm-dd: set system date |
| ethernet [netstat, desc] | Will show Ethernet information; ethernet ? for help |
| getcmdlist | List of names of available PLC shell commands |
| getmulticoreinfo | Shows whether or not multicore is supported and the number of available processor cores |
| ```getprgprop [not implemented, yet]``` | - |
| getprgstat [<application name>\| <application index>]* | Provides the program status of the given application, or the program status of all loaded applications if no application is given |
| getsramlayout | Display layout of complete SRAM |


| Command with Possible Parameters | Description |
| :---: | :---: |
| gettaskgroups | Provides a list of all task groups, their tasks, and the CPU core binding |
| gettaskgroups | Provides a list of defined task groups <br> The assigned tasks are shown for each task group. |
| gettasks | Get list of all tasks, its name and the optional bound taskgroup |
| heapinfo | Print memory usage and heap statistics |
| io-bus [stat, desc] | Will show I/O bus information; io-bus ? for help |
| ```license [export,import,required,availabl e,generate]``` | License handling commands; license ? for help |
| pid [<application name>\| <application index>]* | Provides the GUID (application index) of one or all loaded applications |
| pinf [<application name>\| <application index>]* | Provides the contents of the following fields from the project information: title, version, author, and description. <br> Requirement: The option "Create POU for properties access automatically" in the "Project Information" dialog is activated. |
| plcload | Shows the processor load of the controller (in percent) |
| proddata | Get production data and public keys information of CPU |
| reboot | Performs PLC reboot |
| reflect | Repeats the given command (for testing the connection) |
| reload[<application name>\| <br> <application index>] | Loads the boot application of the given application, or the boot projects of all loaded applications if no application is given |
| resetprg [<application name>\| <application index>]* | Resets the given application, or all loaded applications if no application is given |
| resetprgcold [<application name>\|<application index>]* | Executes a cold boot of the given application, or all loaded applications if no application is given |
| restoreretains <br> [<applicationname>] | Restore retains to files(s). [Optional only from specified application]. |
| restoresram <filename> | Restore complete SRAM content to file |
| rtc-get | Provides the universal time (UTC) via the DataTime string |
| rtc-set | Sets the universal time (UTC) via the DataTime string (see ISO 8601), required format: "rtc-set YYYY-MM-DDThh:mm:ss[,sss |
| rtsinfo | Provides information about the runtime system, for example the processor and version of the runtime system |
| safestopinfo <slot> | Get version errors from DPRAM from communication module <slot> |
| saveretains [<applicationname>] | Save retains to files(s). [Optional only from specified application]. |
| savesram <filename> | Save complete SRAM content to file |
| sdclone <slot> | Clone content of SM560-s at slot <slot> |
| sessinfo-getcnt | Provides the number of currently logged in clients/users |
| sessinfo-list | Provides a list of all currently logged in clients/users |
| showsecuritysettings | Show the list of all security settings and the current configuration |
| sram | Export or import data from SRAM |
| startprg [<application name>\| <application index>]* | Starts the given application, or all loaded applications if no application is given |


| Command with Possible Parameters | Description |
| :--- | :--- |
| stopprg [<application name>। <br> <application index>] * | Stops the given application, or all loaded applications if no applica- <br> tion is given |
| time [hh:mm:ss] | Get/set system time; time: get system time; time hh: mm: ss: set <br> system time |
| trenddatastore | Configure location of the trend DB, reboot of CPU and download is <br> required |
| versioninfo | Get version information on AC500 components |

## *Application name: Name of the application in the device tree

Application index: Results from the list of all applications on the controller that you can call with the "applist" command. Index 0 stands for the first application in the list, 1 for the second, and so on.

See also

- $\Rightarrow$ Chapter 6.4.1.13.7 "Using PLC shell for requesting information" on page 2022
- $\Leftrightarrow$ Chapter 6.4.1.21.2.8.2 "Generic device editor" on page 2427
- $\Rightarrow$ Chapter 6.4.1.21.3.5.13 "Command 'Project information'" on page 2604


## Tab '<device name> I/O Mapping'

Devices with I/O This tab is displayed in device editors for devices with I/O channels. It shows the available channels channels and allows for the mapping of input, output, and memory addresses of the controller to variables or entire function blocks of the application. In this way, you create the 'I/O Mapping'. The application that is to take care of the I/O handling is defined on the "PLC Settings" tab.


You can use the "Online Configuration Mode" if the device supports it. In this mode, you can access the I/Os of the hardware without having to download an actual application to the device beforehand.

You can also create the I/O mapping in the "Edit I/O Mapping" dialog. Here you get a mapping list with search and filter functions for an entire device tree.

## 0

## NOTICE!

## Mapping 'too large' data types

If a variable of a data type that is larger than a byte is mapped to a byte address, the value of the variable will be truncated to byte size there. For monitoring the variable value in the "//O Mapping" dialog, this means that, in the root element of the address, the value is displayed which the variable currently has in the project. The current individual bit values of the byte are displayed in succession in the bit elements below that, but this may not be sufficient for the entire variable value.

[^36]

The tab contains a table for editing the I/O mapping. The information displayed for the inputs and outputs originates from the device description.

| "Find" (1) | Input field for a search string for the mapping table. The search results are marked in yellow. |
| :---: | :---: |
| "Filter" (2) | List box with filters for the I/O mappings displayed in the mapping table: <br> - "Show all" <br> - "Show only outputs" <br> - "Show only inputs" <br> - "Show only unmapped variables" <br> - "Show only mapped variables" <br> - "Show only mapping to existing variables" <br> - "Show only mapping to new variables" |
| ↔"Add FB for IO channel" (11) | Depending on the device, available if the channel entry is selected in the mapping table. Opens the "Select Function Block" dialog for selecting the function block that should be linked directly to the channel. |
| $\rightarrow$ "Go to instance" (12) | Available if the entry is selected in the mapping table. Jumps to the corresponding entry on the "<device name> IEC Objects" tab. |


| "Variable" | Depending on the device, the inputs and outputs of the device are displayed as nodes and below them, indented, the associated channels or, depending on the device, only the implicitly created device instance. <br> The symbol indicates the type of channel: <br> Input <br> *: Output <br> Double-click the cell to open an input field. <br> - Option 1: The variable already exists; specify complete path: <application name>.<module name>.<variable name>; example: app1.plc_prg.ivar; input assistance via ... $\square$ <br> - Option 2: The variable does not exist yet; enter a simple name; automatically created internally as a global variable. <br> Depending on the device, inputs or outputs can be linked directly to a function block. In this case, the «"Add FB for IO channel" button can be clicked. See above. |
| :---: | :---: |
| "Mapping" (3) | Type of mapping: <br> - z: Existing variable <br> - New variable <br> - $\quad$ FE: Mapping to function block instance |
| " Channel" (4) | Symbolic name of the channel. |
| "Address" (5) | Address of the channel (example: \%IW0). <br> Address strikethrough: Indicates that you should not assign any more variables to this address. Reason: Although the variable specified here is managed - as an existing variable -at a different memory location, ambiguity could result when the values are written, particularly with outputs. <br> Indicates that this address has been edited and fixed. If the arrangement of the device objects in the device tree changes, then CODESYS does not adapt this address automatically. |
| "Type" (6) | Data type of the channel (example: BOOL). <br> Structures or bit fields defined in the device description are displayed only if they are part of the IEC standard and are identified as IEC data types in the device description. Otherwise the table cell remains empty. <br> When mapping structured variables, the editor prevents you from specifying both the structure variable (example: $\% \mathrm{QBO}$ ) and individual structure elements (example: \% QB 0.1 and QBO .2 ). Therefore, if there is a main entry with a subtree of bit channel entries in the mapping table, then the following applies: You can input a variable either into the line of the main entry, or into the lines of the subelements (bit channels), but not into both. |
| "Default value" | Default value of the parameter that applies to the channel: Appears only if the option "Set all outputs to default" is selected in the "PLC Settings" for the behavior of the outputs at stop. <br> Note: For compiler version V3.5 SP11 and higher, the initialization value of the variables is used automatically as the default value when mapping to an existing variable. You can edit the "Default value" field only if you map to a new created variable or if no mapping is specified. In older versions, users had to specify explicitly that the default value and initialization value were identical. |
| "Unit" (7) | Unit for the parameter value (example: ms for milliseconds). |
| "Description" (8) | Short description of the parameter. |
| "Current value" | Actual value of the parameter applied to the channel; displayed in online mode only. |

The change of the default value by an online change is allowed, however the value is applied only after a "Reset cold" or "Reset warm".

| "Reset Mapping" (9) | CODESYS resets the mapping settings to the default values as defined in the <br> device description file. |
| :--- | :--- |
| "Always update variables" (10) | Definition for the device object about updating I/O variables. The default value is <br> defined in the device description: <br> - "Use parent device setting": Update according to the setting of the superordi- <br> nate device. <br> - "Enabled 1 (use bus cycle task if not used in any task)": CODESYS updates <br> the I/O variables in the bus cycle task if they are not used in any other task. <br> - "Enabled 2 (always in bus cycle task)": CODESYS updates all variables in <br> each cycle of the bus cycle task, regardless of whether they are used and <br> whether they are mapped to an input or output channel. |

If a UNION is represented by I/O channels in the mapping dialog, it depends on the device whether mapping to the root element is also possible.

Devices with I/O For devices with I/O drivers, you can set the bus cycle task here in the "//O Mapping" tab if the drivers general settings should not be used ("PLC Settings" tab).

Table 421: Bus Cycle Options

| "Bus Cycle Task |
| :--- |
|  |
| General infor- <br> mation about <br> the bus cycle <br> task |

The list box provides all tasks which are defined in the task configuration of the active application (example: "MainTask". In case of "Use parent bus cycle setting", the settings of the parent node will be used.

## General inforthe bus cycle task

Generally, for each IEC task, the used input data is read at the start of each task (1) and the written output data is transferred to the I/O driver at the end of the task (3). The implementation in the I/O driver is decisive for additional transfer of the I/O data. It is responsible for the time frame and time point that the actual transfer to the corresponding bus system occurs.
The bus cycle task of the PLC can be defined globally for all fieldbuses in the PLC settings. For some fieldbuses, however, you can change this independent of the global setting. The task with the shortest cycle time is used as the bus cycle task (setting: "unspecified" in the PLC settings). The messages are normally sent on the bus in this task.

Other tasks copy only the I/O data from an internal buffer that is exchanged only with the physical hardware in the bus cycle task.

(1) Read inputs from input buffer
(3) Write outputs to output buffer
(5) Input buffer
(7) Copy data to/from bus
(9) Bus cycle task, priority 1, 1 ms
(10) Bus cycle task, priority 5
(11) Bus cycle task, priority 10, interrupted by task 5

Task usage
The "Task Deployment" tab provides an overview of used I/O channels, the set bus cycle task, and the usage of channels.

## WARNING!

If an output is written in various tasks, then the status is undefined, as this can be overwritten in each case.

If the same inputs are used in various tasks, then it is possible for the input to change during the processing of a task. This happens when the task is interrupted by a task with a higher priority and causes the process image to be read again. Solution: At the beginning of the IEC task, copy the input variables to variables and then work only with the local variables in the rest of the code.
Conclusion: Using the same inputs and outputs in several tasks does not make any sense and can lead to unexpected reactions in some cases.

## See also

- $\xi^{4}$ Chapter 6.4.1.8.2 "Configuring Devices and I/O Mapping" on page 1836
- $\boldsymbol{y y}^{\boldsymbol{y}}$ Chapter 6.4.1.21.2.8.2 "Generic device editor" on page 2427
-     * $\boldsymbol{y}^{*}$ Chapter 6.4.1.21.3.5.35 "Command 'Edit I/O Mapping'" on page 2615
- $\boldsymbol{y y}^{4}$ Chapter 6.4.1.21.2.8.13 "Tab '<device name> IEC Objects"' on page 2449
- (y) Chapter 6.4.1.21.4.4 "Dialog 'Select Function Block'" on page 2746
- ${ }^{\Perp}$ Chapter 6.4.1.21.3.5.39 "Command 'Online Config Mode"" on page 2616
- ${ }^{4}$ Chapter 6.4.1.21.2.8.10 "Tab 'PLC Settings'" on page 2439


## Tab '<device name> IEC Objects'

## NOTICE!

Please note that manually creating another instance of the device object can lead to malfunctions.

In this tab of the generic device editor, "objects" are listed that allow for access to the device from the IEC application. In online mode, this is used as the monitoring view.
For devices for which a function block instance is created implicitly, at least this instance is listed as an object here in the table. This instance can be used, for example, in order to restart a bus or to query information from the application. The device type determines whether this kind of device instance is available and which access options it has. Please refer to the help for the special device configuration.

Instances of function blocks that are linked with inputs or outputs of the device are also displayed here. The mapping of a function block to a channel is defined in the "<device name> I/O Mapping" tab. The $\rightarrow$ "Go to Instance" command takes you directly to the affected object from there.
In addition, you can create more objects in the table here that are not yet linked with a device channel.

In online mode, you can use the table of IEC objects as a monitoring view. It also shows the current value, the address, and the comment for the function block variable at the channel. Finally, it provides the capability of writing and forcing values.

| $\begin{aligned} & \ddagger \text { "Add" } \\ & \square \text { "Edit" } \end{aligned}$ | Opens the "Select Function Block" dialog for creating a new instance or for editing the instance selected in the table. |
| :---: | :---: |
| Х "Delete" | Deletes the selected entry. |
| $\rightarrow$ + "Go to Variable" | Jumps from the selected entry directly to the corresponding mapping in the "<device name> //O Mapping" tab. |
| "Variable" | The object name comprises the device name and the function block name. Example: EL2004_Relay. Changing the device name has an immediate effect. The part of the name after the device name is editable here. |
| "Mapping" | Mapping type, as in the "<device name> I/O Mapping" tab |
| "Type" | Data type: Here it is the name of the function block. |
| "Value" <br> "Prepared value" <br> "Address" <br> " Comment" | In online mode only: <br> Display of the current value, the address, and the comment for the variable at the channel. Moreover, the option of specifying a value for writing or forcing the variable. |

See also

- " $\boldsymbol{y}$ Chapter 6.4.1.21.2.8.12 "Tab '<device name> I/O Mapping'" on page 2444


## Tab 'Users and Groups'

## NOTICE! <br> Recommendations regarding data security

In order to minimize the risk of data security violations, we recommend the following organizational and technical actions for the system where your applications are running. Whenever possible, avoid exposing the PLC and control networks to open networks and the Internet. Use additional data link layers for protection, such as a VPN for teleaccess, and install firewall mechanisms. Restrict access to authorized persons only, and change any existing default passwords during the initial commissioning, and change them regularly.

On this tab of the generic device editor, you edit the device user management of the PLC.
Depending on how it is supported by the device, you can define user accounts and user groups. In combination with the configuration on the "Access Rights" tab, you thus control access to control objects and files at runtime.
Requirements: The controller has a user management and allows it to be edited. You have login data in order to be able to log in to the controller.

It is possible to apply user account definitions from the project user management into the device user management (see below: "Import" button).

Table 422: Toolbar of the tab

| G/Gynchronization | Switches on and off the synchronization between the editor and the user man- <br> agement on the device. |
| :--- | :--- |
| If the button is not pressed, then the editor is blank or it contains a configuration |  |
| that you loaded from the hard disk. |  |
| When the button is pressed, CODESYS synchronizes the display in the editor |  |
| continuously with the current user management on the connected device. |  |
| When you enable the synchronization while the editor contains a user configura- |  |
| tion that is not synchronized with the device yet, you are prompted what should |  |
| happen to the editor contents. Options: |  |
| - "Upload from the device and overwrite the editor content": The configuration |  |
| on the device is loaded into the editor, overwriting the current contents. |  |
| "Download the editor content to the device and overwrite the user |  |
| management there": The configuration in the editor is transferred to the |  |
| device and applied there. |  |

Table 423: "User"

| All currently defined users, and below them their memberships of user groups, are listed in a tree structure. |  |
| :--- | :--- |
| ॰ "Add" | Opens the "Add User" dialog for creating a new user account. |
| $\diamond$ "Import" | Opens the "Import User" dialog. It displays all the user accounts defined in the <br> project user management. <br> Select the desired entries and click "OK" in order to import them into the device <br> user management. CAUTION: The passwords are NOT applied. |


| "Edit" | Opens the "Edit User <user name>" dialog. It corresponds to the "Add User" <br> dialog and you can change the settings of the user account. |
| :--- | :--- |
| - "Delete" | Deletes the account of the currently selected user. |

Table 424: "Groups"

| All currently defined groups, and below them the users assigned to them, are listed in a tree structure. |  |
| :--- | :--- |
| - "Add" | Opens the "Add Group" dialog. <br> Define a new group name. From the list of defined users, select those that are to <br> belong to the group. Click "OK" to confirm the selection. The group is displayed <br> in the tree. |
| " "Import" | Opens the "Import User" dialog. It displays all the user groups defined in the <br> project user management. <br> Select the desired entries and click "OK" in order to import them into the device <br> user management. |
| "Edit" "Delete" | Opens the "Edit Group <group name>" dialog. It corresponds to the "Add Group" <br> dialog where you can change the group definition. |
|  | Deletes the currently selected group. |

Table 425: "Add Dialog 'Add User"'

| "Name" | Name of the new user |
| :--- | :--- |
| "Default group" | List box with all configured user groups. Every user has to belong to at least one <br> group. You define this here as a default group. |
| "Password" |  |
| "Confirm password" |  |
| "Password strength" | Password security in a range from "Very weak" to "Very good". |
| "Hide password" | 目: The password is shown only with asterisks "*" when it is typed in. |
| "Password can be changed by <br> the user" |  |
| "Password must be changed at <br> first login" |  |

See also

- « Chapter 6.4.1.21.2.8 "Object 'Device' and Generic Device Editor" on page 2427
- $\Leftrightarrow$ Chapter 6.4.1.21.2.8.2 "Generic device editor" on page 2427
- ${ }^{*}$ Chapter 6.4.1.11.4 "Handling of Device User Management" on page 1971
- $\Leftrightarrow$ Chapter 6.4.1.21.2.8.15 "Tab 'Access Rights"' on page 2453
- ${ }^{\circ}$ Chapter 6.4.1.21.3.7.16 "Command 'Add Device User'" on page 2637


## Tab 'Access Rights'

## NOTICE!

## Recommendations regarding data security

In order to minimize the risk of data security violations, we recommend the following organizational and technical actions for the system where your applications are running. Whenever possible, avoid exposing the PLC and control networks to open networks and the Internet. Use additional data link layers for protection, such as a VPN for teleaccess, and install firewall mechanisms. Restrict access to authorized persons only, and change any existing default passwords during the initial commissioning, and change them regularly.

## NOTICE!

Detailed information on the concept and use of device user management is provided in "Handling of Device User Management".

There you will also find the following instructions on how to use the editor:

- First-time login to the controller for editing and viewing its user management
- Setting up a new user in the user management of the controller
- Changing of access rights to controller objects in the user management of the controller
- Loading user management from a *.dum file, modifying it, and downloading it to the controller in offline mode

On this tab of the device editor, you define the device access rights of device users to objects on the controller. As in the project user management, users must be members of at least one user group and only user groups can be granted certain access rights.
Requirements for the "Access Rights" tab to be displayed:

- In the CODESYS options, in the "Device Editor" category, the "Show access rights page" option has to be selected.
Note that this CODESYS option can be overwritten by the device description.
Requirements for the access rights to be granted to user groups
- A component for the user management has to be available on the controller. That is the primary requirement.
- Users and user groups have to be configured on the "Users and Groups" tab.

Table 426: Toolbar of the tab

| Gy Synchronization | Switches on and off the synchronization between the editor and the user man- <br> agement on the device. <br> If the button is not pressed, then the editor is blank or it contains a configuration <br> that you loaded from the hard disk. <br> When the button is pressed, CODESYS synchronizes the display in the editor <br> continuously with the current user management on the connected device. <br> When you enable the synchronization while the editor contains a user configura- |
| :--- | :--- |
| tion that is not synchronized with the device yet, you are prompted what should |  |
| happen to the editor contents. Options: |  |

Table 427: "Objects"
In the tree structure, the objects are listed to which actions can be executed at runtime. The objects are each assigned by their object source and partially sorted in object groups. In the "Rights" view, you can configure the access options for a user group to a selected object.
Object source (root node)

- "File system objects $\rightarrow$ Device": In these objects, the rights can be granted to folders of the current execution directory of the controller.
- "Runtime objects $\rightarrow$ /": In these objects, all objects are managed that have online access in the controller and therefore have to control the access rights.
A description of the objects is located in the table. ", "Overview of the objects" on page 2457
Object groups and objects (indented)
Example: "Device" with child nodes "Logger", "PlcLogic", "Settings", "UserManagement".

Table 428: "Rights"
In general, the access rights are inherited from the root object (also "Device" or " $/$ " to the subobjects. This means that if a permission of a user group is denied or explicitly granted to a parent object, then this first affects all child objects.
The table applies for the object that is currently selected in the tree. For every user group, it shows the rights currently configured for the possible actions on this object.


Possible actions on the object:

- "Add/Remove"
- "Modify"
- "View"
- "Execute"

When an object is clicked, a table on the right side shows the access rights of the available user groups for the selected object.

This allows you to quickly see:

- Which access rights are evaluated by an object
- Which user group has which effective rights to which object

Meanings of the symbols

- $\ddagger$ : Access right granted explicitly
- $\quad=$ : Access right denied explicitly
- $\quad$ Access right granted through inheritance
- $\quad=$ : Access right denied through inheritance
- $\approx$ : The access right was not granted or denied explicitly and also not inherited by the parent object. Access is not possible.
- No symbol: Multiple objects are selected that have different access rights.

Change the permission by clicking the symbol.

| Example | The "Logger" object on the "Access Rights" tab was created by the "Logger" component and controls its access rights. It is located directly below the "Device" runtime object. <br> The possible access rights for this object can be granted only for the "View" action. |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | Objects: | Rights: |  |  |  |  |
|  | $\square$ Runtime objects A |  | Add/Remove | Modify | View | Execute |
|  | $=-\Rightarrow$ Device | Administrator |  |  | \& |  |
|  | - $\Rightarrow$ Logger | Developer |  |  | 4 |  |
|  | $\cdots$ Plclogic | Everyone |  |  | 号 |  |
|  | $\square \Rightarrow$-Backup8Restore | Service |  |  | \& |  |
|  | - $\Rightarrow$ PlcShell | Watch |  |  | \& |  |

Initially, each object has a read access. This means that every user can read the "Logger" of a controller. If this access right should be denied for a single user group ("Service" in the example), then the read access to the logger object has to be denied explicitly.

| Objects: |  | Rights: |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| -- Runtime objects | - |  | Add/Remove | Modify | View | Execute |
| - $\Rightarrow$ Device |  | Administrator |  |  | 4 |  |
| $\Rightarrow$ Logger |  | Developer |  |  | 4 |  |
| - $\Rightarrow$ Plclogic |  | Everyone |  |  | \& |  |
| $\Rightarrow$-Backup\&Restore |  | Service |  |  | - |  |
| PlcShell |  | Watch |  |  | ¢ |  |

Overview of the objects

| "Runtime objects $\rightarrow$ Device" |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| "Logger" | Online access to the logger is read only. Therefore, only the "View" access right can be granted or denied here. |  |  |  |  |  |
| "PlcLogic" | All IEC applications are inserted here automatically as child objects during download. When an application is deleted, it is removed automatically. This allows specific control of online access to the application. Access rights can be assigned centrally over all applications in the "PlcLogic" The "Administrator" and "Developer" user groups have full access to the IEC applications. The "Service" and "Watch" user groups only have read access (for example for read-only monitoring of values). |  |  |  |  |  |
|  | The following table shows which action is affected in particular when a specific access right is granted for an IEC application. <br> $\mathbf{x}$ : The right has to be set explicitly. <br> - : The right is not relevant. |  |  |  |  |  |
|  | "Application" | Operation | Access Rights |  |  |  |
|  |  |  | "Add/ Remove" | "Execute" | "Modify" | "View" |
|  |  | Login | - | - | - | x |
|  |  | Create | x | - | - | - |
|  |  | Create child object | x | - | - | - |
|  |  | Delete | x | - | - | - |
|  |  | Download / online change | x | - | - | - |
|  |  | Create Boot Application | x | - | - | - |



## See also

- $\quad$ Chapter 6.4.1.11.4 "Handling of Device User Management" on page 1971
- $\quad y^{4}$ Chapter 6.4.1.21.2.8.2 "Generic device editor" on page 2427
- ${ }^{4}$, Chapter 6.4.1.21.2.8.14 "Tab 'Users and Groups'" on page 2450


## Tab 'Symbol Rights'

In this tab of the generic device editor, you define the access rights of different user groups (clients) to the individual symbol sets available on the controller.

Requirement: User management must be set up on the PLC. An application was downloaded to the controller for which symbol sets were defined in the CODESYS project. They have access data for logging in to the controller.
In the "Symbol Sets" view, all symbol sets are listed below the "Application" node whose definition was downloaded with the application to the controller.
In the "Rights" view, the user groups defined in the user management of the controller are listed in a table. When a symbol set is selected, you see the access rights of the corresponding user group to the symbols of this set. 饣: Access granted; $=$ : Access not granted. You can change the access rights by double-clicking the symbol.

Click the button to save the current access configuration to an XML file The file type is "Device symbol management files (*.dsm)". Click the button to read a file like this from the hard drive.

See also

- "Creating symbol sets with different access rights for different control clients" on page 1943
- ${ }^{4}$ ) Chapter 6.4.1.21.2.8.2 "Generic device editor" on page 2427
- ${ }^{\wedge}$ Chapter 6.4.1.21.2.8.15 "Tab 'Access Rights"' on page 2453
- $\Leftrightarrow^{*}$ Chapter 6.4.1.21.2.8.14 "Tab 'Users and Groups'" on page 2450


## Tab 'Licensed Software Metrics'

The tab of the device editor displays the code sizes of the applications of the open project in a tree structure. The display is refreshed when you click "Build $\rightarrow$ Generate Code" or "Online $\rightarrow$ Login" for the active application. When the compile information is deleted, the displayed code size of the corresponding application is reset.

| "Metric" | Applications of the open project |
| :--- | :--- |
| "Size" | - "Size of User Code": Sum of the displayed code sizes of the applications <br> listed below <br> - Code size of the respective application |
| "Unit" | Unit in which the "Size" is displayed |
| "Max. Allowed" | Not implemented yet |

## Tab 'Task deployment'

This sub-dialog box of the device editor displays a table of inputs and outputs as well as their assignment to the defined tasks.

The information only becomes visible after code has been generated for the application. It is used for troubleshooting, because it shows where inputs or outputs are used in several tasks with different priorities. Multiple use can lead to undefined values through overwriting.


Table 429: "//O Deployment for Tasks" (1)

| "//O Channels" (2) | All inputs and outputs of the linked devices. The display corresponds to that in <br> the dialog box "I/O Mapping" of the device editor. <br> By double-clicking on an input or output you can open the associated I/O map- <br> ping editor. |
| :--- | :--- |
| "<task name>" (3) | A column appears for every task defined in the task configuration. The title <br> contains the task name and priority. <br> The priority of the tasks decreases from first to the last column. A red cross <br> appears in the box for inputs and outputs that are written or read by a task: <br> In addition, the task defined as a "Bus cycle task" in the "PLC Settings" of the <br> device editor is marked at these points with a blue double arrow symbol $\sigma$. <br> Following a mouse-click on the title cell, only the I/Os assigned to this task are <br> displayed. <br> Following a mouse-click on the "//O Channels" cell, all channels are shown <br> again. |

See also

- \& Chapter 6.4.1.21.2.8.2 "Generic device editor" on page 2427
- \& Chapter 6.4.1.21.2.8.12 "Tab '<device name> I/O Mapping'" on page 2444
- \& Chapter 6.4.1.21.2.8.10 "Tab 'PLC Settings'" on page 2439
- $\boldsymbol{\xi}^{\boldsymbol{y}}$ Chapter 6.4.1.9.17.2 "Creating a task configuration" on page 1914


## Tab 'Status'

This tab of the generic device editor displays status information, for example 'Running' or 'Stopped', and specific diagnostic messages from the respective device, also information about the card used and the internal bus system.

See also

- ${ }^{2}$ Chapter 6.4.1.21.2.8.2 "Generic device editor" on page 2427


## Tab 'Information'

This tab of the generic device editor displays general information that originates from the device description file: name, vendor, categories, version, order number, description, if necessary an illustration.

See also

- $\Leftrightarrow$ Chapter 6.4.1.21.2.8.2 "Generic device editor" on page 2427


## Object 'DUT'

Symbol:

- for a DUT without text list support
- for an enumeration data type with text list support

A DUT (Data Unit Type) declares a user-specific data type.
You can add this kind of object below the application or in the "POUs" view. When the object is created, the "Add DUT" dialog opens. There you select among the "Structure", "Enumeration", "Alias", or "Union" data types.
Moreover, enumerations can have a text list stored to localize the enumeration values. Then the object also has a localization view.

```
Syntax TYPE <identifier> : <data type declaration with optional
initialization>
END_TYPE
```

How the data type declaration has to be done syntactically depends in detail on the selected data type.
Examples
Declaration of a structure
TYPE S POLYGONLINE :
TYPE S POLYGONLINE :
STRUCT
STRUCT
aiStart : ARRAY[1..2] OF INT := [-99, -99];
aiStart : ARRAY[1..2] OF INT := [-99, -99];
aiPoint1 : ARRAY[1..2] OF INT;
aiPoint1 : ARRAY[1..2] OF INT;
aiPoint2 : ARRAY[1..2] OF INT;
aiPoint2 : ARRAY[1..2] OF INT;
aiPoint3 : ARRAY[1..2] OF INT;
aiPoint3 : ARRAY[1..2] OF INT;
aiPoint4 : ARRAY[1..2] OF INT;
aiPoint4 : ARRAY[1..2] OF INT;
aiEnd : ARRAY[1..2] OF INT := [99, 99];
aiEnd : ARRAY[1..2] OF INT := [99, 99];
END_STRUCT
END_STRUCT
END_TYPE
END_TYPE
Extension of a structure
TYPE S_PENTAGON EXTENDS S_POLYGONLINE :
TYPE S_PENTAGON EXTENDS S_POLYGONLINE :
STRUCT
STRUCT
aiPoint5 : ARRAY[1..2] OF INT;
aiPoint5 : ARRAY[1..2] OF INT;
END_STRUCT
END_STRUCT
END_TYPE
END_TYPE
Declaration of
\{attribute 'qualified_only'\}
\{attribute 'strict'\}
TYPE E_TRAFFICSIGNAL :
(
eRed,
eYellow,
eGreen := 10
);
END_TYPE
Enumeration with text list support in the localization view

| ETL_TRAFFICSIGNAL |  |  |  |  |  |  |  |  |  |
| :--- | :--- | :--- | :--- | :--- | :--- | :---: | :---: | :---: | :---: |
| Member | Value | de | en |  |  |  |  |  |  |
| eRed |  | rot | red |  |  |  |  |  |  |
| eYellow |  | gelb | yellow |  |  |  |  |  |  |
| eGreen | 10 | grün | green |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

The 空 "Textual View" and $\square$ "Localization View" buttons are located on the right edge of the editor. Click the buttons to toggle between the views.
Declaration of an alias
Declaration of a union of components with different data types
TYPE A MESSAGE : STRING[50];
TYPE A MESSAGE : STRING[50];
END_TY\overline{PE}
END_TY\overline{PE}
TYPE U DATA :
TYPE U DATA :
UNION
UNION
lrA : LREAL;
lrA : LREAL;
liA : LINT;
liA : LINT;
dwA : DWORD;
dwA : DWORD;
END_UNION
END_UNION
END_TYPE
END_TYPE

Dialog 'Add Function: The dialog is used to configure a new DUT (Data Unit Type). DUT'

Call: Menu bar: "Project $\rightarrow$ Add Object $\rightarrow$ DUT"; context menu of the application object.

| "Name" | Name of the new DUT data type <br> Example: S_POLYGONLINE |
| :--- | :--- |

Table 430: "Type"

| "Structure" | Creates an object which declares a structure that combines multiple variables with different data types into a logical unit. The variables declared within the structure are called members. <br> Example: S POLYGONLINE |
| :---: | :---: |
| "Extends" | Extends an existing structure by more members. In the input field, specify an existing structure. The members of the existing structure are automatically available in the new structure. <br> Example: <br> PENTAGON |
| "Enumeration" | Creates an object which declares an enumeration that combines multiple integer constants into a logical unit. The constants declared within an enumeration are also called enumeration values. <br> Example: E_TRAFFICSIGNAL |
| "Add Text List Support" | $\square$ : Enumeration that does not have any text list support Enumeration with additionally stored text list for the enumeration values. The text list allows you to localize the names of the enumeration values. <br> Example: ETL_TRAFFICSIGNAL <br> Note: In the case of an existing enumeration type, text list support can be added or removed at any time. As a result, the "Add Text List Support" and "Remove Text List Support" commands are provided in the context menu of the object. <br> Hint: The localized texts can be displayed, for example, in a visualization. In this case, the text output of a visualization element displays the symbolic enumeration values in the current language instead of the numeric enumeration values. When an enumeration with text list support is specified in the "Text variable" property of a visualization element, it gets the additional property enumeration name\gg. <br> Example: In a visualization, you use the variable PLC_PRG.eTrafficLight of type ETL_TRAFFICSIGNAL. ETL_TRAFFICSIGNAL is an enumeration with text list support. Then the entry in the properties editor of the visualization element looks like this: PLC_PRG.eTrafficLight <ETL_TRAFFICSIGNAL>. <br> Hint: When you edit the enumeration type in the application, a prompt opens when you close the application and asks whether the affected visualizations should be updated automatically. <br> See also: Help for "Enumerations" with information about the declaration syntax |
| "Alias" | Creates an object which declares an alias with which an alternative name is declared for a base type, data type, or function block |
| "Union" | Creates an object which declares a union that combines multiple members with mostly different data types into a logical unit. <br> All members have the same offset so that they are occupy the same memory. The memory requirement of a union is determined by the memory requirement of its "largest" component. |
| "Add" | Closes the dialog and creates the new object <br> The object is displayed with the ${ }^{\circ}$ symbol in the device tree or in the "POUs" view. When a text list is also stored for the object, the symbol is displayed. |

See also

- \# Chapter 6.4.1.20.5.19 "Alias" on page 2267
- © Chapter 6.4.1.20.5.18 "Enumerations" on page 2263
- \& Chapter 6.4.1.20.5.17 "Structure" on page 2261
-     * $\Rightarrow$ Chapter 6.4.1.20.5.20 "Data type 'UNION"' on page 2268
- \& Chapter 6.4.1.21.3.21.12 "Command 'Add Text List Support"' on page 2732
- *s Chapter 6.4.1.21.3.21.13 "Command 'Remove Text List Support"' on page 2732
- Help for CODESYS Visualization: Using Texts


## Object 'External File'

An "External File" is any file that you add to the project in the "POUs" view or "Devices" view. Click "Project $\rightarrow$ Add Object" to open the "Add External File" dialog and define how the file belongs to the project.
An external file which was inserted in the "POUs" view is never downloaded to the controller.
An external file which was added in the "Devices" view is always downloaded to the controller when an online change or a download is performed due to an IEC code change.
When an external file is downloaded to the controller, it is not updated in the project.

## Dialog 'Add External File'

| "File path" | The ... button opens a dialog for selecting a file in the local file system. |
| :--- | :--- |
| "Name" | Object name for the file in CODESYS. If you do not type anything, the file will <br> have its previous name. |

Table 431: "File Handling"

| "Remember the link" | The file is available in the project only as long as it exists in the defined file path. |
| :--- | :--- |
| "Remember the link and <br> embed into project" | CODESYS saves an internal copy of the file in the project, as well as the link to <br> the defined file path. The update option selected below applies as long as the <br> external file exists there. Otherwise CODESYS uses the version saved in the <br> project. |
| "Embed into project" | CODESYS saves only one copy of the file in the project. There is no longer a link <br> to the external file. |

Table 432: "Change Tracking"

| "Reload the file automatically" | If the external file changes, then CODESYS updates the file in the project. |
| :--- | :--- |
| "Prompt whether to reload the <br> file" | If the external file changes, then a dialog prompt opens whether CODESYS <br> should also update the file in the project. |
| "Do nothing" | The file remains unchanged in the project, even if the external file changes. |


| "Display File Properties" | Clicking this button opens the default "Properties of <file name>" dialog, which <br> you can also open in the Windows file system by right-clicking the file. |
| :--- | :--- |
| "Open" | The file object is inserted into the device tree ("Devices" or "POUs" view) and <br> opened in the editor for the matching file format. |

See also

- ${ }^{\bullet}$ Chapter 6.4.1.21.3.5.1 "Command 'Add Object"' on page 2598
- $\Leftrightarrow$ Chapter 6.4.1.21.4.11.8 "Dialog 'Properties' - 'External file"' on page 2757


## Object 'GlobalTextList'

## Symbol:

This object is for the management and translation of texts that are written as static text in visualizations in the project. It contains a table with these texts. If you write a text in a visualization in an element under the property "Texts", CODESYS automatically adds a line in the table. You cannot write any new text here, you can only edit an existing text.

In addition CODESYS makes the following commands available, in order to consolidate the "GlobalTextList":

- "Check Visualization Text IDs"
- "Update Visualization Text IDs"
- "Remove Unused Text List Entries"

The object is located in the POUs view and exists once at the most

| "ID" | Unambiguous identifier of the text |
| :--- | :--- |
| "Default" | Source text as a character string with one formatting specification at the most, <br> for example Information A: \%i possibilities. If no translation is written <br> under a language column , CODESYS uses this text. <br> Double-click in the field in order to edit the text. |
| The table contains as many language columns as you have added. A language column is named with a language <br> code that you entered when creating the column with the command "Insert Language". |  |
| "<Language code>" | Name of the language as a language code, for example en-US. This column <br> contains the translation of the text that is written under "Standard". <br> If the language code is selected as a language in the visualization manager, <br> a visualization displays the translation during operation. A running visualization <br> can switch over during operation to another language at the request of a user. <br> Double-click in the field in order to edit the text. |

See also

- " $y^{2}$ Chapter 6.4.1.21.3.21.1 "Command 'Add Language'" on page 2728
- $\Leftrightarrow$ Chapter 6.4.1.21.3.21.2 "Command 'Create Global Text List"' on page 2728
- $\Leftrightarrow$ Chapter 6.4.1.21.3.21.6 "Command 'Import/Export Text Lists'" on page 2729
- $\Leftrightarrow$ Chapter 6.4.1.21.3.21.7 "Command 'Remove Language'" on page 2730
- $\Leftrightarrow$ Chapter 6.4.1.21.3.21.9 "Command 'Remove Unused Text List Entries'" on page 2731
- $\quad \stackrel{y}{c}$ Chapter 6.4.1.21.3.21.10 "Command 'Check Visualization Text IDs'" on page 2731
- $\Leftrightarrow$ Chapter 6.4.1.21.3.21.11 "Command 'Update Visualization Text IDs'" on page 2731
- $\quad{ }^{\circ} \gg$ Chapter 6.4.1.21.2.28 "Object 'Text List"' on page 2532
- $\Leftrightarrow$ Chapter 6.4.1.9.10 "Managing text in text lists" on page 1891


## Object 'GVL' - Global Variable List

Symbol:
A global variable list is used for the declaration, editing and display of global variables.
A GVL is added to the application or the project with the command "Project $\rightarrow$ Add Object
$\rightarrow$ Global Variable List".

If you insert a GVL under an application in the Device tree, the variables are valid within this application. If you add a GVL in the POUs view, the variables are valid for the entire project.

You can apply settings for the editor of the object in the dialog "Tools $\rightarrow$ Options" in the categories "Declaration Editor" and "Text Editor".
If the target system supports network functionality, you can convert the variables of a GVL into network variables and thus use them for data exchange with other devices in the network. To do this you must define corresponding properties for the GVL in the "Network Variables" tab of the "Properties" dialog.
See also

- ② Chapter 6.4.1.9 "Programming of Applications" on page 1844
- ${ }^{\Perp}$ Chapter 6.4.1.21.4.11.12 "Dialog 'Properties' - 'Network Variables'" on page 2760
- $\quad$ ${ }^{\circ}$ Chapter 6.4.1.21.4.14 "Dialog 'Options'" on page 2781


## Object 'GVL' - Global Variable List (task-local)

## Symbol: 徐

A global variable list (task-local) is used for the declaration, editing and display of global variables. For this special global variable list, the declared variables in the list can be written by one task only. All other tasks have only read-only access. This makes sure that the values of these variables are always consistent, even for multicore projects.

The object is available for compiler version 3.5.13.0 with the corresponding device description.
"Task with write access" $\quad$ Task that has exclusive write access to the variables.

See also

- ${ }^{4}$ Chapter 6.4.1.9.4.6 "Using Task-Local Variables" on page 1855
- ${ }^{*}$ Chapter 6.4.1.9.4.5 "Declaring global variables" on page 1854
- $y^{4}$ Chapter 6.4.1.21.2.12 "Object 'GVL' - Global Variable List" on page 2465
- $\Leftrightarrow$ Chapter 6.4.1.21.2.29.6 "Tab 'Task Groups'" on page 2537

Object: IEC Symbol Publishing

The" IEC Symbol Publishing" object contains an editor for configuring variables and IEC data types from the project for publishing via an OPC UA Server. A group of symbols can be compiled and configured for each created object.
The editor is a newer alternative to the already known editor for the Symbol Configuration.
Use the Add Object dialog to attach one or more IEC Symbol Publishing objects below an OPC UA Server object below the Communication Manager. You can change the object name, for example to webvisu, if the symbols are to be used for exchange with a web visualization.

Editor: IEC Function: Configuration of a symbol set for exchange via OPC UA Symbol Publishing

Call: Double click or [Edit Object] command on the desired "<name of IEC symbol publishing>" object in the device tree.
The editor consists of two tabs: On the" IEC Symbol Editor" tab, you configure the symbol set to be published from individual variables from the project. On the "Symbol Type Editor" tab, you work with whole data types, such as GVL and FB.

Note that the editor works with variables and types from the precompile state of the project. If changes are made to variables or types in the project, then it can create inconsistencies with the current configuration in the symbol publishing editor. They are indicated at the relevant places in the editor by icons of the $\$$ error, $\$$ warning, or information. A tooltip on the icon provides more information about the inconsistency and a possible fix. The tooltip contains at least the message text, which also appears in the message window in the "IEC Symbol Publishing" category.

| $\$$ "Refresh" | Refreshes the display in the editor according to the current <br> situation in the project |
| :--- | :--- |
| "Use access rights defined in <br> the user management" | 目: The access rights to the symbol set are taken from the <br> current settings in the user management of the controller. |

Editor: IEC On this tab of the editor, you configure individual variables from the application as symbols to be Symbol Publishing, Tab: IEC Symbol Editor published. You do this by dragging the desired variables from the "Precompile Sets" from the left part of the tab to the right part, where the symbols can then be configured.

| "Precompile Sets" | The tree structure shows all variables which are already available for publication in the project before compilation. To do this, the project scans the following: <br> - "<device>.<application>": Project POUs below the device application <br> - "Libraries": Integrated libraries <br> - "Pool": Project POUs in the "POUs" view <br> A list of excluded types can be found in the tooltip after the Warum erscheint meine Variable nicht icon. |
| :---: | :---: |
| D Sync | This button is operable as long as there are inconsistencies between the current symbol configuration and the project. In this case you will see icons of the type $\$, \oplus$, or 9 information. After correcting the inconsistencies, synchronize the editor with the project again. |
| 7 | The display of information about the configured symbols can be filtered. In the list after $\gamma$, you select which table columns are to be displayed. When you enter a string in the search field, the search hits are highlighted in yellow in the table. |
| Table of configured symbols | Information about the individual symbols, depending on the set filter: <br> - Symbol: Variable name (example: iVar) <br> - Path: Variable path (example: PLC_PRG.iVar, PLC_PRG.fb1_inst) <br> - IEC type: IEC data type of the variable (example: INT, FB1) <br> - Symbol Type (Exported Type Name): Name of the symbol under which it will appear after export. In the case of structured types, the instance can be selected here. <br> - Access Rights: Permissions to the symbol. Example: Read/Write <br> - Comment: Comment stored in the application code |

Editor: IEC Symbol Publishing, Tab: Symbol Type Editor

On this tab of the editor, you configure entire "symbol types" for publishing. To do this, you drag the desired "IEC Data Types" from the left part of the window to the right side into the configuration table. Then there you can configure for each symbol type in detail which of its components should be exported with which access rights and with which name.

You can also filter the display in this table via $\gamma$ and search for strings.

| Symbol Type (Exported Type <br> Name) | The name of the symbol type dragged from IEC data types <br> can be edited. It is used after export as defined here. |
| :--- | :--- |
| Members | The $\ldots$ button opens the Data type members of <symbol <br> type> dialog (see above). |
| IEC Type | IEC data type as defined in the project (example: $\because$ Color) |

Dialog: Data type members of <symbol type>.
Call: ... button on the" IEC Symbol Type Editor" tab in the "Members"
In this dialog, you select which members of the symbol type in question should be exported as symbols. You can define a name for each member under which it will be exported, including the access rights.

| Export | $\checkmark:$ The member is exported as a symbol. |
| :--- | :--- |
| Exported Name | Default: Member name (IEC Name) from the project; can be <br> edited |
| IEC Name | Member name from the project |
| IEC Type | IEC data type of the member (example: INT) |
| Symbol Type (Exported Type <br> Name) | Name of the data type to which the member belongs <br> (example: |
| Access Rights | Access right to the symbol (Read, Write, Read/Write) |

Above the table, there are filter and search options, just like on the editor tabs described above.

## Object 'Image Pool'

The "Image Pool" object contains a table with image ID assignments.

| "ID" | ID of the image; you reference this ID, for example in the visualization of the <br> image. |
| :--- | :--- |
| "File name" | File path of the image; if you click for more settings (...), the "Select Image" <br> dialog box opens. |
| "Image" | Show a thumbnail of the image. |
| "Link type" | Opens the "Select Image" dialog box, where you define the link type. |

Dialog box 'Se-
lect Image'

| "Image file" | Name and directory of the image file (example: "C:IPro- <br> grammelimagesllogo.bmp") CODESYS supports the following image formats: <br>  <br> BMP, EMF, GIF, ICO, JPG, PNG, SVG, and TIFF. Please note that a controller <br> may not support all formats. <br>  <br>  <br>  <br>  <br> Whether or not you can use images formatted as scalable vector graphics (*.svg) <br> depends on the operating system. Any necessary information is located in the <br> device description of the hardware vendor. |
| :--- | :--- |

Table 433: "File Handling"

| "Remember the link" | CODESYS saves only the link. CODESYS automatically updates any changes to <br> an image file in the image pool. You must ensure that the path of the image file <br> does not change. <br> When saving the project as an archive, CODESYS embeds the image file in the <br> project archive. |
| :--- | :--- |
| "Remember the link and |  |
| embed into project" | CODESYS copies the image to the image pool and the link information is <br> retained. In this way, CODESYS recognizes any changes to the image file and <br> then update the image pool can as needed. This behavior is controlled with the <br> options in the next table. <br> Embedded image files increase the memory requirement of the project. |
| "Embed into project" | CODESYS copies the image to the image pool. If the image file is changed <br> again afterwards, then it is not updated in the project. For libraries, you must <br> embed the image in the project. <br> Embedded image files increase the memory requirement of the project. |

Table 434: "Change Tracking"
These options are available only if you have selected the "Remember the link and embed into project" check box as described above.

| "Reload the file automatically" | CODESYS automatically updates the image file in the project without prompting. |
| :--- | :--- |
| "Prompt whether to reload the <br> file" | If the image file has changed, you may be prompted whether or not the image <br> file should be updated. |
| "Do nothing" | CODESYS does not update the image file in the image pool. |

See also

- ${ }^{4}$ Chapter 6.4.1.9.11 "Using image pools" on page 1899


## Object 'Library Manager'

## Symbol:

The Library Manager lists all libraries that were integrated in the project for creating applications. It provides information about the type of library, its properties, and its contents.
You can expand or collapse the list of integrated libraries, as well as edit library properties for non-dependent libraries.
The Library Manager consists of three views:

- Upper view: List of integrated libraries
- Lower left view: Tree structure with all modules of the library selected in the upper view
- Lower right view: Documentation for the module selected in the tree

See also

- *a Chapter 6.4.1.17 "Using Libraries" on page 2034

List of inte- List of all libraries integrated in the project. If a library depends on other libraries, then these grated libraries

| Displayed in gray fonts | The library was added to the project automatically by means of a plug-in. |
| :--- | :--- |
| Displayed in black fonts | The library was added to the project automatically by means of the "Add Library" <br> command. |
| "Name" | Display of the integrated library in the following syntax: <br> "<placeholder name> = <library name>, <version> (<company>)": <br> "<placeholder name>": If it is a placeholder library for a library, then the place- <br> holder name is before a " ". <br> "<library name>": Name of the library that is used for management in the library <br> repository. <br> "<version>": Version that was referenced at the first time it was integrated. <br> " (<company>)": Vendor (optional) |
| "Namespace" | Namespace for unique access to the contents of the library. <br> It is prepended to a module identifier for this purpose: <br> <namespace>.<library module identifier> |
| "ne namespace usually coincides with the library name. |  |
| The |  |
| Note: If the library has the property LanguageModelAttribute |  |
| "qualified-access-only", then you must access the library module in |  |
| the application code by means of the namespace. Qualified (unique) access |  |
| is enforced. |  |
| You can modify the standard namespace for local use (within the project) in the |  |
| "Properties" dialog. |  |


| Q | Library that is signed with a trusted certificate (compatible with CODESYS >= V3 <br> SP15) |
| :--- | :--- |
|  | Library that is signed with a trusted certificate, but references at least one <br> unsigned library |
|  | Library that is signed with a private key and token (compatible with CODESYS < <br> V3 SP15) |
|  | Library that is not signed, or signed with an untrusted or expired certificate. In <br> the case of an untrusted certificate, the "Trust Certificate" command is provided <br> in the context menu. |
|  | Library that is defined as optional and not currently available |
|  | Library whose status is being determined |
|  | Licensed library for which no valid license is currently available |
|  | Library symbol for a library that cannot be loaded because its signature (encryp- <br> tion) could not be verified |

Table 435: Commands in the Library Manager

| $\pm$ "Add Library" | Opens the dialog for selecting a library. All libraries installed in the library repository are offered. |
| :---: | :---: |
| $\times$ "Delete Library" | Removes the presently selected library from the project |
| - "Properties" | Opens the dialog for the display and editing of the properties of the presently selected library |
| Tix "Details" | Opens a dialog with details for the presently selected library (general information, contents, properties, license information) |
| -0 "Try to Reload Library" | If you select a library marked as not found, you can attempt to load it into the project again using this command. |
| -"Download Missing Libraries" | CODESYS scans for the missing libraries in the download servers specified in the project options. <br> After that you can download and install the library. |
| 或 "Placeholders" | The "Placeholders" dialog opens. The current resolution is displayed there and you can edit it. |
| 何 "Library Repository" | Opens the "Library Repository" dialog for installing and uninstalling libraries and for defining library locations |
| (1) "Icon legend" | Opens the "Information" dialog with a legend of the icons that display the current status of a library in the list of integrated libraries (see above) |
| ® "Summary" | Opens the "Library Summary" dialog. All libraries referenced in the project are displayed in a tree structure in the dialog, and those libraries which reference these libraries. <br> - Command "Display all occurrences in library hierarchy and close dialog": In the editor of the Library Manager, the libraries in the open tree structure are marked which reference or use this library. Requirement: A library is selected. The "Information" dialog is then closed. <br> This command is also executed when you double-click a library. <br> Display of the libraries <br> - "Managed Library": Name and version of the library <br> - "Number of Occurrences": Number of locations where this library is referenced by other libraries. <br> When you click " + " for a library, the libraries, which reference this library, are displayed in the next level down. |


| "4 "Trust Certificate" | Only in the context menu of a library selected in the Library Manager, in which <br> the library has been signed with an untrusted certificate. The command turns the <br> untrusted certificate into a trusted certificate and the prepended icon changes <br> from L to |
| :--- | :--- |
| "Export Library" | Only in the context menu of a library selected in the Library Manager: Opens the <br> default dialog for saving the library file in the file system |

See also

- ${ }^{\text {® }}$, Chapter 6.4.1.21.3.15.1 "Command 'Add Library'" on page 2712
- ${ }^{*}>$ Chapter 6.4.1.21.3.15.5 "Command 'Export Library'" on page 2716
- $y^{\circ}$ Chapter 6.4.1.21.3.15.3 "Command 'Properties'" on page 2714
- ${ }^{4}$ Chapter 6.4.1.21.3.9.5 "Command 'Library Repository'" on page 2657
- $\Leftrightarrow$ Chapter 6.4.1.21.3.15.4 "Command 'Placeholders"' on page 2716
- $\quad$ \& Chapter 6.4.1.21.4.14.16 "Dialog 'Options' - 'Library Download"' on page 2790
- $\Leftrightarrow$ Chapter 6.4.1.21.2.24 "Object 'Project Information"' on page 2515

Tree structure of All library modules that were integrated with the library are listed in the tree structure. all modules of a selected library

Requirement: A library is selected in the upper view.

The usual sorting and search functions are available in the menu bar.

## Documentation

 for the library module selected in the lower left view| Tab "Inputs/Outputs" | Interface (inputs/outputs) of the library module |
| :--- | :--- |
| Tab "Graphical" | Graphical display of the module |
| Tab "Documentation" | Documentation for the library module. <br> Note: As a library developer, you have to follow the rules for documentation <br> inclusion in 'Guidelines for library development'. |
| Tab "Parameter List" | Requirement: The library project contains a parameter list. <br> You can change the values of these parameters in the column "Value (editable)". |
| See also |  |

- ${ }^{4}$ ) Chapter 6.4.1.17.3 "Information for Library Developers" on page 2035


## Object 'OPC UA Information Model'

Symbol:
The "OPC UA Information Model" object is added to the "Communication Manager" in the application. When added, an OPC UA publishing object $\square$ and below that an information model object $\xlongequal{\circ}$ as a child object are also added.
In the "Add OPC UA Information Model" dialog, specify a name for the information model and select the OPC UA information model. The selection includes the OPC UA information models which are installed in the "OPC UA Information Model Repository".

## OPC UA information model editor

Symbol:
The editor is used to select the object types and data types of the OPC UA information model which you want to use in the open CODESYS project. The selected OPC UA types are converted to IEC types in the editor.
$\left.\begin{array}{|l|l|}\hline \text { "Browse Information Model" } & \begin{array}{l}\text { List box } \\ \text { The currently used information model and the information models which are } \\ \text { referenced by the current model are displayed. The dependencies depend on } \\ \text { the respective information model. The OPC UA base model is always displayed. }\end{array} \\ \hline \text { "Generate IEC declarations" } & \begin{array}{l}\text { Generates an IEC declaration for all OPC UA types converted into an IEC type. } \\ \text { The generated IEC types are saved in a folder (example: "OPC Objects") in the } \\ \text { "Devices" view and can be used in the implementation of the IEC code. When } \\ \text { implementing the CODESYS project, you can select them in the "Input Assistant" } \\ \text { dialog. } \\ \text { When the IEC declarations are generated, the appropriate attrib- } \\ \text { utes are automatically added to the generated POUs (example: } \\ \text { ' opcua. mapping. type ', opcua.mapping. member. accesslevel '). } \\ \text { Note: The attributes added by the system should not be changed by the user. } \\ \text { When the IEC type cannot be created, the entry UNKNOWN_TYPE is displayed } \\ \text { in the declaration instead of the data type. The user should delete this variable } \\ \text { because in this case it is almost always an OPC UA feature which is not sup- } \\ \text { ported yet. OPC UA features which are not supported yet are grayed out in the } \\ \text { left area. }\end{array} \\ \hline \text { Right area: Object types and data types of the OPC UA information model which are mapped to IEC types }\end{array}\right\}$

| "Name" | Name of the IEC POU or data type in the project <br> By default, the name of the type is displayed in the OPC UA information <br> model. OPC UA also supports names which are invalid in IEC. In these cases, <br> CODESYS automatically generates a valid IEC name. <br> You can change the name. |
| :--- | :--- |
| "IEC Type" | IEC type to which the OPC UA type was mapped (example: BOOL, "Method"). |
| "OPC UA Type" | Corresponds to the "Element Type" displayed in the left area |

## OPC UA pub-

 lishing editorSymbol:
In the editor, the instances (OPC UA objects) of the OPC UA types are configured which should be available to the OPC UA Clients via the controller

| "Search for Mapped Instances" | Searches in GVLs and PRGs below the current application for instances of the mapped OPC UA types which have already been declared. The search result is displayed in the list. <br> Note: Instances in the "POUs" view and in libraries are not taken into consideration. |
| :---: | :---: |
| "Create New Instance" | Opens the "Create New Instance" dialog to select the IEC type for which a new instance should be generated. <br> Instances can be generated for the POUs which have been created in the OPC UA information model editor from OPC UA types. These instances can be used in POUs in the application. <br> Requirement: In the OPC UA information model editor, the "Generate IEC declarations" command has been executed after mapping the OPC UA types to the IEC types. |
| "Root Node" | Selection of directories or the object instance of the server which is displayed on the OPC UA Client for publishing the instances. The list box depends on the applied OPC UA companion specification. |
| Tabular list of generated instances: |  |
| "OPC UA Variable" | Variable which has been generated as an instance of an OPC UA type. This variable can be published in an OPC UA Client. <br> You can edit the displayed name. |
| "OPC UA Type" | OPC UA type of the "OPC UA Variable" |
| "Map or Generate" | - ": The "OPC UA Variable" has been mapped to an existing variable. <br> - : The "OPC UA Variable" has been generated as a new instance. |
| "IEC Variable" | Full variable name |
| "IEC Type" | IEC type of the IEC variable |


| "Access Rights" | Note that an OPC UA Client may have read/write access the OPC UA variable. <br> In the function blocks, the access rights to the variables can be changed by <br> attributes which can also be read from the XML file if necessary. <br> Reading and writing |
| :--- | :--- |
| "Maximal" | Maximum possible permissions for the OPC UA variable |

See also - « Chapter 6.4.1.21.3.9.12 "Command 'OPC UA Information Model Repository"' on page 2665

## Object 'Network Variable List (Sender)'

Symbol:
A network variable list (sender) is used for declaring and listing global variables that should be sent to network variable lists (receiver) of other devices or network projects.
You add the object to the device tree by clicking "Add Object $\rightarrow$ Network Variable List (Sender)" of an application.
You can configure the protocol and transfer parameters in the "Add Network Variable List
(Sender)" dialog box or "Properties" dialog box of the object in the "Network Variables" tab.

Dialog Box 'Add Function: This dialog box defines the network properties for the sender NVL. When you close Network Vari- the dialog box, CODESYS adds the sender NVL of the application to the device tree.

Call: Main menu "Project $\rightarrow$ Add Object $\rightarrow$ Network Variable List (Sender)" while the application is selected in the device tree.

This dialog box corresponds to the "Network Variables" tab in the "Properties" of the network variable list object.
See also

- $\Rightarrow$ Chapter 6.4.1.21.4.11.12 "Dialog 'Properties' - 'Network Variables'" on page 2760
- $\quad$ • Chapter 6.4.1.21.2.19 "Object 'Network Variable List (Receiver)"' on page 2475
- 


## Object 'Network Variable List (Receiver)'

Symbol:
The object is used for listing the received network variables and displaying the information: network and transmit information and sender.
You add the object to an application by clicking "Add Object
$\rightarrow$ Network Variable List (Receiver)".
The network variable list (receiver) shows the received network variables, which were declared in network variable list (sender) of another device or project. You cannot change the network variables in the object editor.

The object editor consists of two parts:

- Information about the sender and transfer log of the list
- List of declarations of network variables

Dialog Box 'Add Function: This dialog box defines the receiver NVL to a sender NVL and adds the receiver NVL Network Variable List
(Receiver)' to the application object in the device tree.

Call: Main menu "Project $\rightarrow$ Add Object $\rightarrow$ Network Variable List (Receiver)" (when the application object is selected).

| "Task" | Task of the current application that controls the variables to be <br> received. |
| :--- | :--- |
| "Sender" | Drop-down list <br> - Available sender NVLs of another device in the project <br> - <br> "Import from file": Required if the necessary sender NVL is <br> defined in another project. For this, the necessary sender NVL <br> must have been generated in another project as "GVL export file <br> ".gvl" in the properties dialog of the NVL in the "Link To File" tab. |
| "Import from file" | File name in "GVL export file *.gvl" format if you have selected <br> "Import from file" for "Sender ". |

See also

## -

- ${ }^{4}$, Chapter 6.4.1.21.2.18 "Object 'Network Variable List (Sender)"' on page 2475


## Object 'Persistent variable list'

## Symbol: T

The object contains the declaration of global persistent variables in the declaration section VAR GLOBAL PERSISTENT RETAIN . . END VAR. The variables are stored in special nonvolatile memory.

The persistence editor shows the variables as a list in the usual way. The displayed list does not influence the persistence behavior of the variables, but only the list stored internally in the process image. The list there contains all variables ever declared in chronological order. Variables that you have removed are marked with a placeholder and continue to exist as a gap.
The declaration section can also contain instance paths, which refer to locally declared persistent variables and were created with the command "Declarations $\rightarrow$ Add All Instance Paths".

## NOTICE!

Before you decide how to set up persistence for an application, it would be helpful for you to be familiar with the use cases described in the "Data Persistence" section. Moreover, it is helpful if you can differentiate between the mechanisms of persistent variables, retain variables, variables of the Persistence Manager, and recipe variables.

Commands The following commands are provided in the persistence editor:

- Command "Declarations $\rightarrow$ Add All Instance Paths"
- Command "Declarations $\rightarrow$ Reorder List and Clear Gaps"


## See also

- TODO
- $\quad$ M Chapter 6.4.1.21.3.18.1 "Command 'Reorder List and Clean Gaps'" on page 2719
- $\Leftrightarrow$ Chapter 6.4.1.9.19.2 "Preserving data with persistent variables" on page 1923
- « Chapter 6.4.1.9.19.3 "Preserving data with retain variables" on page 1925
- $\quad$ © Chapter 6.4.1.21.3.18.4 "Command 'Add all instance paths'" on page 2720


## Object 'POU'

## General

Symbol: | \# |
| :---: |

An object of the type "POU" is a Program Organization Unit in a CODESYS project. You write source code for your controller program in POUs.
There are the following types of POUs:

- Program
- Function
- Function block

A "POU" object is inserted by using the command "Project $\rightarrow$ Add Object" in the Device tree or in the "POUs" view. When adding a POU you define the POU type and the implementation language.
You can also add other programming objects (method, action, etc.) to these objects.

## Calling POUs

Certain POUs can call other POUs. Recursions are not permitted.
When calling POUs via the namespace, CODESYS browses the project for the POU to be called in accordance with the following order:

1. Current application
2. "Library Manager" of the current application
3. "POUs" view
4. "Library Manager" in the "POUs" view

If you want to call a POU that exists with the same name in a library used in the application and as an object in the "POUs" view, note the following: There is no syntax that allows you to call the POU in the "POUs" view only by its name. In this case you must shift the library from the application's library manager to the project's library manager (in the "POUs" view). After that you can call the POU object in the "POUs" view purely by its name. If you add the namespace to the library, you can call the POU of the library.

The term "POU" is also used in CODESYS for the "POUs" view in which CODESYS manages the global objects in the project.

See also

- $\quad$ Chapter 6.4.1.21.2.21 "Object 'POU"' on page 2477
- $\quad$ © Chapter 6.4.1.21.4.11 "Dialog 'Properties'" on page 2753

Dialog 'Add Function: The dialog is used to configure a new POU according to the IEC 61131-3 standard. POU' This means that a POU can be a program, a function, or a function block.
Call: "Project $\rightarrow$ Add Object" menu; context menu in the "Devices" view when an application is selected; context menu in the "POUs" view

| "Name" | Name of POU |
| :--- | :--- |

Table 436: "Type"

| "Program" |  |
| :---: | :---: |
| "Function Block" | "Extends": Specification or selection of a base function module in the sense of object-oriented programming. Specified with the EXTENDS keyword in the function block declaration <br> "Implements": Specification or selection of an interface in the sense of object-oriented programming. Specified with the IMPLEMENTS keyword in the function block declaration. When the POU is created, all methods are created which are defined via the interface. <br> "Final": Derived access is not allowed. This means that you cannot extend the function block with another function block. This allows for optimized code generation. <br> "Abstract": Identifies that the function block has a missing or incomplete implementation and cannot be instantiated. Abstract FBs are used exclusively as base function blocks and the implementation typically occurs in a derived FB. If a non-abstract function block is created, which in turn extends an abstract function block, then all abstract methods of the abstract basic function block are added to the new function block as (non-abstract) methods. <br> "Access specifier" <br> - "PUBLIC": Corresponds to the specification of no access specifier. <br> - "INTERNAL": Access to the function block is restricted to the namespace (library). <br> - "Method implementation language": When you select the "Implements" option, you can select an implementation language here for all method objects that CODESYS generates by means of the implementation of the interface. <br> The "Method implementation language" does not depend on the implementation language of the function block. |
| "Function" | Note: Not available when "Sequential Function Chart (SFC)" is selected as the "Implementation language". <br> "Return type:": Data type of the return value |
| "Implementation language" | Implementation language of the POU |

See also

- ${ }^{\mu}$ Chapter 6.4.1.21.2.21.3 "Object 'Function Block'" on page 2479
- ${ }^{\bullet}>$ Chapter 6.4.1.21.2.21.2 "Object 'Program'" on page 2478
- ${ }^{*}>$ Chapter 6.4.1.21.2.21.4 "Object 'Function'" on page 2482
- $\Leftrightarrow$ Chapter 6.4.1.9.22.2 "Extension of function blocks" on page 1929
- $\boldsymbol{y y}^{\text {• }}$ Chapter 6.4.1.9.22.3 "Implementing interfaces" on page 1931


## Object 'Program'

A program is a POU that supplies one or more values during execution. After execution of the program, all values are retained until the next execution. The order of calling the programs within an application is defined in task objects.
A program is added to the application or the project using the command "Project $\rightarrow$ Add Object $\rightarrow P O U$ ". In the Device tree and in the "POUs" view the program POUs have the suffix " $(P R G)$ ".

The editor of a program consists of the declaration part and the implementation part.
The uppermost line of the declaration part contains the following declaration:
PROGRAM <program>


See also

- $\quad 4$ Chapter 6.4.1.21.2.21 "Object 'POU"' on page 2477
- $\boldsymbol{y y}^{2}$ Chapter 6.4.1.9.17 "Task Configuration" on page 1914


## Object 'Function Block'

A function block is a POU that yields one or more values when executed.
The object is added to the application or the project by clicking "Project $\rightarrow$ Add Object $\rightarrow$ POU". In the device tree or in the "POUs" view, function block POUs have the "(FB)" suffix.
It always calls a function block by means of an instance that is a copy of the function block.

The editor of a function block consists of the declaration part and the implementation part.
The values of the output variables and the internal variables remain unchanged after execution until the next execution. This means that the function block does not necessarily return the same output values for multiple calls with the same input variables.
In addition to the functionality described in IEC 61131-3, you can also use function blocks in CODESYS for the following functionalities of object-oriented programming:

- Extension of a function block
- Implementation of interfaces
- Methods
- Properties

The top line of the declaration part contains the following declaration:

```
FUNCTION_BLOCK <access specifier> <function block> | EXTENDS <function
block> | IMPLEMENTS <comma-separated list of interfaces>
```

| Calling a func- <br> tion block | The call is always made by means of an instance of the function block. When a function block is <br> called, only the values of the respective instance change. |
| :--- | :--- |
|  | Declaration of the instance: |
|  | <instance> : <function block>; |

<instance> : <function block>;

You access a variable of the function block in the implementation part as follows:
<instance> . <variable>

## NOTICE!

## Note the following:

- You can access only input and output variables of a function block from outside the function block instance, not the internal variables.
- Access to a function block instance is restricted to the POU in which the instance is declared, unless you have declared the instance globally.
- You can assign the desired values to the function block variables when you call the instance.



## Assigning variable values when calling:

In the textual languages IL and ST, you can assign values directly to input and/or output variables when you call the function block.

A value is assigned to an input variable with := .
A value is assigned to an output variable with $=>$.
Example $\quad$ The instance CMD_TMR of the timer function block is called with assignments for the input
variables IN and PT. Then the output variable $Q$ of the timer is assigned to the variable A.

```
PROGRAM PLC_PRG
VAR
    CMD_TMR : TOF;
END_VAR
CMD_TMR(IN := %IX5.1, PT := T#100MS);
A := CMD_TMR.Q;
```

When you insert a function block instance by means of the "Input Assistant" and select the "Insert with arguments" option in the "Input Assistant" dialog, CODESYS inserts the call with all input and output variables. Then you only have to insert the desired value assignment. In the example above, CODESYS inserts the call as follows: CMD_TMR (IN:=, PT:=, $Q=>$ ).

You can use the attribute 'is_connected' and a local variable to determine at the time of the call in the function block instance whether or not a specific input receives an external assignment.

## See also

- \# Chapter 6.4.1.21.2.21 "Object 'POU"' on page 2477
- \&Chapter 6.4.1.9.22.2 "Extension of function blocks" on page 1929
- \& Chapter 6.4.1.9.22.3 "Implementing interfaces" on page 1931
- $\&$ Chapter 6.4.1.21.2.21.6 "Object 'Method'" on page 2485
- $\Rightarrow$ Chapter 6.4.1.21.2.21.9 "Object 'Property"' on page 2493
- ${ }^{\omega}$ Chapter 6.4.1.20.6.3.24 "Attribute 'is_connected"' on page 2293


## Object 'Function'

A function is a POU that supplies precisely one data element when executed and whose call in textual languages can occur as an operator in expressions. The data element can also be an array or a structure.
The object is added to the application or the project by clicking "Project $\rightarrow$ Add Object $\rightarrow$ POU". In the device tree or in the "POUs" view, function POUs have the "(FUN)" suffix.

## NOTICE!

Functions have no internal status information, which means that functions do not save the values of their variables until the next call. Calls of a function with the same input variable values always supply the same output value. Therefore, functions must not use global variables and addresses!

The editor of a function consists of the declaration part and the implementation part.
The top line of the declaration part contains the following declaration:
FUNCTION <function> : <data type>
Below that, you declare the input and function variables.
The output variable of a function is the function name.

## NOTICE!

If you declare a local variable in a function as RETAIN, this has no effect. In this case, CODESYS issues a compiler error.

## NOTICE!

You cannot mix explicit and implicit parameter assignments in function calls in CODESYS V3. This means that you have to use either only explicit or only implicit parameter assignments in function calls. The order of the parameter assignments when calling a function is arbitrary.

[^37]

Functions with additional outputs

According to the IEC 61131-3 standard, functions can have additional outputs. You declare the additional outputs in the function between the keywords VAR_OUTPUT and END_VAR. The function is called according to the following syntax:

```
<function> (<function output variable1> => <output variable 1>,
<function output variable n> => <output variable n>)
```

Example The fun function is defined with two input variables in1 and in2. The output variable of the fun function is written to the locally declared output variables loc1 and loc2.

```
    fun(in1 := 1, in2 := 2, out1 => loc1, out2 => loc2);
```

See also

- $\quad \Leftrightarrow$ Chapter 6.4.1.21.2.21 "Object 'POU"' on page 2477


## Object 'Interface'

Symbol: $\simeq$
Keyword: InTERFACE
An interface is a means of object-oriented programming. The object ITF describes a set of method and property prototypes. In this context, prototype means that the methods and properties contain only declarations and no implementation.
This allows different function blocks having common properties to be used in the same way. An object "ITF" is added to the application or the project with the command "Project $\rightarrow$ Add Object $\rightarrow$ Interface".

Table 437: "Adding an interface"

| "Inheritance" |  |
| :--- | :--- |
| "Name" | Interface name |
| "Extends" | $\checkmark:$ Extends the interface that you enter in the input field or via the input assistant <br> $\ldots$. <br> are also available in the new interface. |

You can add the objects "Interface property" and "Interface Method" to the object "ITF". Interface methods may contain only the declarations of input, output and input/output variables, but no implementation.
So that you can also use an interface in the program, there must be a function block that implements this interface.

This means:

- the function block contains the interface in its IMPLEMENTS list in its declaration part
- the function block contains an implementation for all methods and property prototypes of the interface

A function block can implement one or more interfaces. You can use the same method with identical parameters, but different implementation code in different function blocks.

Please note the following:

- You may not define variables within an interface. An interface has no implementation part and no actions. Only a collection of methods is defined, in which you may define only input, output and input/output variables.
- CODESYS always treats variables declared with the type of an interface as references.
- A function block that implements an interface must contain implementation code for the methods of the interface. You have named the methods exactly as in the interface and the methods contain the same input, output and input/output variables as in the interface.


## NOTICE!

## Interface references and online change

The following can happen with a compiler version < 3.4.1.0: if a function block changes its data because variables are added or deleted, or because the type of variables changes, then CODESYS copies all instances of the function block to a new memory location. In this case, however, an interface reference refers not to the new memory location, but still to the old one.

In case of compiler versions >= 3.4.1.0, CODESYS automatically re-addresses the interface references so that CODESYS also references the correct interface in case of an online change. CODESYS requires additional code and more time for this, so that jitter problems can occur depending on the number of objects concerned. Therefore, CODESYS displays the number of variables and interface references concerned before the execution of the online change and you can then decide whether the online change should be executed or aborted.


- ${ }^{*}$ Chapter 6.4.1.9.22.3 "Implementing interfaces" on page 1931
- \& Chapter 6.4.1.9.22.4 "Extending interfaces" on page 1933


## Object 'Method'

Symbol: IM $_{\text {m }}$
Keyword: METHOD

Methods are an extension of the IEC 61131-3 standard and a tool for object-oriented programming which is used for data encapsulation. A method contains a declaration and an implementation. However, unlike a function, a method is not an independent POU, and it is subordinated to a function block or program. A method can access all valid variables of the superordinate POU.

You can use interfaces for the organization of methods.
You can add a method below a program or a function block. Click "Project $\rightarrow$ Add Object $\rightarrow$ Method" to open the "Add Method" dialog.

Declaration - The variables of a method contain temporary data that are valid only during the execution of the method (stack variables). All variables that are declared and implemented in a method are reinitialized each time the method is called.

- Like functions, methods can have additional outputs. You have to assign these additional outputs in the method call.
- Depending on the declared access specifier, a method can be called only within its own namespace (INTERNAL), only within its own POU and its derivatives (PROTECTED), or only within its own POU (PRIVATE). For PUBLIC, the method can be called from anywhere.
Interface methods can have declared input, output, and VAR_IN_OUT variables, but do not contain an implementation.
See also
- \& Chapter 6.4.1.21.2.21.7 "Object 'Interface Method"" on page 2490

Implementation - Access to function block instances or program variables is allowed in the implementation of the method.

- The THIS pointer allows for access to its own function block instance. Therefore, the pointer is allowed only in methods that are assigned to a function block.
- A method cannot access VAR_TEMP variables of the function block.
- A method can call itself recursively.


## NOTICE!

When you copy a method below a POU and add it below an interface, or move the method there, the contained implementation is removed automatically.

## Calling a method

Syntax for calls:

```
<return value variable> := <POU name> . <method name> ( <method input
name> := <variable name> (, <further method input name> := <variable
name> )* );
```

For the method call, you assign transfer parameters to the input variables of the method.
Respect the declaration when doing this. It is enough to specify the names of the input variables without paying attention to their order in the declaration.

## Example

Declaration

```
METHOD PUBLIC DoIt : BOOL
VAR_INPUT
    iInput_1 : DWORD;
    iInput-}2 : DWORD
    sInput_3 : STRING(12);
END_VAR
```

Call
bFinishedMethod := fbInstance.DoIt(sInput_3 := 'Hello World ', iInput_2 := 16\#FFFF, iInput_1 := 16);
When the method is called, the return value of the method is assigned, for example, to variables declared locally. When you omit the names of the input variables, you have to pay attention to the declaration order.

## Example

## Declaration

```
METHOD PUBLIC DoIt : BOOL
VAR_INPUT
        iInput_1 : DWORD;
        iInput_2 : DWORD;
        sInput_3 : STRING(12);
END_VAR
```

Call

```
bFinishedMethod := fbInstance.DoIt( 16, 16#FFFF,'Hello World ');
```


## Recursive method call

Within the implementation, a method can call itself, either directly by means of the THIS pointer, or by means of a local variable for the assigned function block.

- THIS^. <method name> ( <parameter transfer of all input and output variables> )
Direct call of the relevant function block instance with the THIS pointer
- VAR fb_Temp : <function block name>; END_VAR

Call by means of a local variable of the method that temporarily instantiates the relevant function block
A compiler warning is issued for a recursive call. If the method is provided with the pragma \{attribute 'estimated-stack-usage' :=
'<sstimated_stack_size_in_bytes>'\}, then the compiler warning is suppressed. For an implementation example, see the "Attribute 'estimated-stack-usage'" chapter.
To call methods recursively, it is not enough to specify only the method name. If only the method name is specified, then a compiler error is issued: "Program name, function or function block instance expected instead of"
See also

- $\Leftrightarrow$ Chapter 6.4.1.9.22.5 "Calling methods" on page 1933
- $\Leftrightarrow$ Chapter 6.4.1.20.6.3.14 "Attribute 'estimated-stack-usage"" on page 2281
- $\Leftrightarrow$ Chapter 6.4.1.20.2.16 "THIS" on page 2126

[^38]| FB_Init | Declarations automatically implicit, but explicit declaration also possible Contains initialization code for the function block, as is defined in the declaration part of the function block |
| :---: | :---: |
| FB_Reinit | Explicit declaration is necessary. <br> Call after the instance of the functio change). It reinitializes the new in |
| FB_Exit | Explicit declaration is necessary. <br> Call for each instance of the function block before a new download or a reset or during an online change for all shifted or deleted instances. |
| Properties | Provides Set and/or Get accessor methods. |
| See also <br> - $\Leftrightarrow$ Chapter 6.4.1.20.10 "Methods 'FB_Init', 'FB_Reinit', and 'FB_Exit'" on page 2336 <br> - $\xi^{*}$ Chapter 6.4.1.21.2.21.9 "Object 'Property'" on page 2493 <br> - ${ }^{\Perp}$ Chapter 6.4.1.21.2.21.8 "Object 'Interface Property'" on page 2490 <br> Dialog 'Add Method' <br> Function: Defines a method below the selected POU when the dialog is closed. <br> Call: Menu bar: "Project $\rightarrow$ Add Object $\rightarrow$ Method"; context menu <br> Requirement: A program (PRG) or a function block (FUNCTION_BLOCK) is selected in the "POUs" view or the "Devices" view. <br> The interface of a method inserted below a basic function block is copied when a method with the same name is inserted below a derived function block. |  |
|  |  |
|  |  |


| "Name" | Example: meth_DoIt. <br> The standard methods FB_Init and FB_Exit are offered in a list box if they <br> are not already inserted below the POU. If it is a derived function block, then the <br> list box also offers all of the methods of the basic function block. |
| :--- | :--- |
| "Return type" | Default data type or structured data type of return value <br> Example: BOOL |
| "Implementation language" | Example: "Structured Text (ST)" |
| "Access specifier" | Controls access to data. <br> - "PUBLIC" or not specified: Access is not restricted. <br> - "PRIVATE": Access is restricted to the program, function block, or GVL. <br> The object is marked as (private) in the POU or device view. The decla- <br> ration contains the keyword PRIVATE. <br> "PROTECTED": Access is restricted to the program, function block, or GVL <br> with its derivations. The declaration contains the keyword PROTECTED. <br> The object is marked as (protected) in the POU or device view. |
| "INTERNAL": Access to the method is restricted to the namespace (library). |  |
| "Abstract" | The object is marked as (internal) in the POU or device view. The <br> declaration contains the keyword INTERNAL. |
| "Add" | V: Identifies that the method does not have an implementation and the imple- <br> mentation is provided by the derived FB |
| Adds a new method below the selected object. |  |

Input support when generating inheriting POUs

When you doing object-oriented programming and using the inheritance (keyword EXTENDS) of POUs, you can get support as follows:
When you insert an action, a property, a method, or a transition below a POU derived from a base POU, the "Add ..." dialog opens. Then the input field for the name extends to a list box. The list box contains a valid selection from the actions, properties, methods, or transitions available in the base POU. Now you can, for example, easily accept a method of the base POU and then adapt it to the derived function of the POU.

Methods and properties with the access modifier PRIVATE are not listed here because they are also not inherited. Methods and properties with the access modifier PUBLIC automatically get a blank access modifier field when accepting into the derived POU, which means the same thing functionally.
Example

## See also

- ${ }^{4}$ Chapter 6.4.1.9.22 "Object-Oriented Programming" on page 1929
- $\sum^{*}$ Chapter 6.4.1.21.2.21.10 "Object 'Action'" on page 2497
- $y^{*}$ Chapter 6.4.1.21.2.21.9 "Object 'Property'" on page 2493
- ${ }^{4}$ Chapter 6.4.1.21.2.21.6 "Object 'Method'" on page 2485
- ② Chapter 6.4.1.21.2.21.11 "Object 'Transition'" on page 2499


## See also

- $\boldsymbol{y y}^{4}$ Chapter 6.4.1.20.1.4.2 "ST editor in online mode" on page 2050
- ${ }^{\bullet}$, Chapter 6.4.1.13.2.3 "Using watch lists" on page 2002
- ๕ Chapter 6.4.1.20.2.10 "Instance variables - VAR_INST" on page 2120


## Object 'Interface Method'

Symbol: $\Phi_{\mathrm{m}}$
This object is used for object-oriented programming.
The object "Interface Method" is added to an interface via the command "Project $\rightarrow$ Add Object".
If a method is inserted underneath an interface, you can add and instance only variable declarations (input, output and input/output variables) in this method.
You can only add program code to the method if a function block 'implements' the interface to which the method belongs. CODESYS then inserts the method underneath the function block.

See also

- ${ }^{*}$, Chapter 6.4.1.21.2.21.5 "Object 'Interface'" on page 2484
- ${ }^{4}$ Chapter 6.4.1.21.2.21.6 "Object 'Method"' on page 2485
- ${ }^{4}$ Chapter 6.4.1.9.22.3 "Implementing interfaces" on page 1931


## Object 'Interface Property'

Symbol: 野
Interface properties are an extension of the IEC 61131-3 standard and a tool for object-oriented programming. An interface property declares the accessor methods Get and Set (no implementation code). Therefore, a function block that implements an interface also inherits their interface properties.
You can add an interface property to the device tree for an interface. Then an interface is extended with the accessor methods Get and Set. The Get accessor is for read access. The Set accessor is for write access. You can delete an unneeded accessor. Click "Project $\rightarrow$ Add Object $\rightarrow$ Interface Property" to add an accessor. The "Add Interface Property" dialog opens.

See also

- ${ }^{2} \gg$ Chapter 6.4.1.21.2.21.5 "Object 'Interface"' on page 2484
- ⓨ Chapter 6.4.1.21.2.21.9 "Object 'Property'" on page 2493


## Example

| Declaration and implementation of the interface property Literal_A |  |
| :---: | :---: |
|  | This interface itf_A has the property Literal_A with the accessor methods Get and Set. <br> The function blocks $f b \_A 1$ and $f b \_A 2$ implement the interface it $_{f} \_A$ and therefore inherit its interface property. Each FB has its own implementation. |
| Interface itf_A | ```INTERFACE itf_A VAR END VAR PROPEERTY Literal_A : STRING``` |
| FB fb_A1 | ```FUNCTION_BLOCK fb_A1 IMPLEMENTS itf_A VAR str_1 : STRING; str_2 : STRING; iCn̄ : INT; END_VAR iCnt := iCnt + 1; str_1 := 'Function block A1';``` |

```
Accessor
```



```
Accessor
fb_A1.Litera VAR
1_\overline{A}.Set END_VAR
    str_2 := Literal_A;
FB fb_A2
    FUNCTION_BLOCK fb_A2 IMPLEMENTS itf_A
    VAR
        str 1 : STRING;
        str_2 : STRING;
            iCnt : INT;
        END_VAR
    iCnt := iCnt + 1;
    str_1 := 'Function block A2';
```


## Accessor

```
fb A2.Litera VAR
1_\overline{A}.Get END_VAR
    Litēral_A := str_1;
Accessor
fb A2.Litera VAR
l_A.Set END_VAR
    str_2 := Literal_A;
Program
PLC_PRG PROGRAM PLC_PRG
    VAR
        iCnt : INT;
        my_1 : fb_A1;
        my_2 : fb_A2;
        str
        strName_2: STRING;
        END_VAR
        iCnt := iCnt + 1;
        my_1();
        my_2();
        strName_1:= my_1.Literal_A;
        strName_2:= my_2.Literal_A;
        my_1.Literal_A':= 'Hello-1';
        my_2.Literal_A := 'World 2';
```

This leads to the following monitoring of PLC_PRG when the application is in runtime mode:


## Object 'Property'

Symbol: 㩆

## Keyword: PROPERTY

Properties are an extension of the IEC 61131-3 standard and a tool for object-oriented programming.
Properties are used for data encapsulation because they allow for external access to data and act as filters at the same time. For this purpose, a property provides the accessor methods Get and Set which allows for read and write access to the data of the instance below the property.

You can add a property with accessor methods below a program, a function block, or a global variable list. Click "Project $\rightarrow$ Add Object $\rightarrow$ Property" to open the "Add Property" dialog.

## You can add an interface property below an interface.

When you copy a property that is inserted below a POU and add it below an interface, or if you move the property there, the included implementations are removed automatically.

See also

- « Chapter 6.4.1.21.2.21.8 "Object 'Interface Property"' on page 2490

Dialog 'Add Property'

Function: Creates a new property below the selected POU when the dialog is closed.
Call: Menu bar: "Project $\rightarrow$ Add Object $\rightarrow$ Property"; context menu
Requirement: A program (PRG), a function block (FUNCTION_BLOCK), or a global variable list (GVL) is selected in the "POUs" view or the "Devices" view.

| "Name" | Name (identifier) of the property <br> Example: prop_iA |
| :--- | :--- |
| "Return type" | Default type or structured type of return value <br> Example: INT |
| "Implementation language" | Example: "Structured Text (ST)" |


|  |  |
| :--- | :--- |
| "Access specifier" | Controls access to data |
| "PUBLIC" or unspecified | Access is not restricted. |
| "PRIVATE" | Access is restricted to the program, function block, or GVL. <br> The object is marked as (private) in the POU or device view. The declaration <br> contains the keyword PRIVATE. |
| "PROTECTED" | Access is restricted to the program, function block, or GVL with its derivations. <br> The object is marked as (protected) in the POU or device view. The declara- <br> tion contains the keyword PROTECTED. |
| "INTERNAL" | Access is restricted to the namespace (library). <br> The object is marked as (internal) in the POU or device view. The declara- <br> tion contains the keyword INTERNAL. |
| "Abstract" | V: Identifies that the property does not have an implementation and the imple- <br> mentation is provided by the derived FB |
| "Add" | Adds a new property be low the selected object and below that the accessor <br> methods Get and Set <br> Note: When you select a property, you can also add a previously removed <br> accessor explicitly by clicking "Add Object". |

Editor 'Property' You can program the data access in the editor. The code can contain additional local variables. However, it must not contain any additional input variables or (as opposed to a function or method) output variables.

```
Example
Function block
FB_A
    FUNCTION_BLOCK FB_A
    VAR_INPUT
    END_VAR
    VAR_OUTPUT
    END - VAR
    VAR
            iA : INT;
    END_VAR
    iA := iA + 1;
Property
prop_iA
Accessor
method
FB_A.prop_iA.
Get
Accessor iA := prop_iA;
method
FB_A.prop_iA.
Set
PROGRAM PLC_PRG
VAR
        fbA : FB A;
        iVar: INT;
END_VAR
fbA();
IF fbA.prop_iA > 500 THEN
        fbA.prop_iA := 0;
END_IF
iVar := fbA.prop_iA;
```

Get and Set accessors

The call of the Set accessor is written to the property. Then it is used in the same way as an input parameter. When the Get accessor is called, the property is read. It is used in the same way as an output parameter. Access is restricted in each case by means of access modifiers (qualifiers). As a result, the objects are identified accordingly.
When a property is accessed as read only or write only, you can delete the unneeded accessors.
You can add accessors explicitly by selecting a property and clicking "Add Object". A dialog opens, either "Add Get accessor" or "Add Set accessor". There you can set the implementation language and the access.

Table 438: Dialog "Add Get (Set) Accessor"

| "Implementation language" | Example: "Structured Text (ST)" |
| :--- | :--- |
|  |  |


| "Access specifier" | Qualifier for the declaration part |
| :--- | :--- |
| PUBLIC or unspecified | Access is not restricted. |
| PRIVATE | Access is restricted to the program, function block, or GVL. <br> The object is marked as (private) in the POU or device view. The declaration <br> contains the keyword. |
| PROTECTED | Access to the property is restricted to the program, function block, or GVL and its <br> derivations. The declaration contains the keyword. <br> The object is marked as (protected) in the POU or device view. |
| INTERNAL | Access to the method is restricted to the namespace (the library). <br> The object is marked as (internal) in the POU or device view. The declara- <br> tion contains the keyword. |
|  | "Add" |

Monitoring of properties in online mode

Input support when generating inheriting POUs

The following pragmas are provided for the monitoring of properties in online mode. You insert them at the top position of the property definition:

- \{attribute 'monitoring' := 'variable'\}

Each time the property is accessed, CODESYS saves the actual value to a variable and displays the value of this variable. This value can become outdated if no more access to the property takes place in the code.

- \{attribute 'monitoring' := 'call'\}

Each time the value is displayed, CODESYS calls the code of the Get accessor. If this code contains a side effect, then the monitoring executes the side effect.

You can monitor a property with the help of the following functions.

- Inline monitoring

Requirement: The "Enable inline monitoring" option is selected in the "Text Editor" category of the "Options" dialog.

- Watch List

See also

- ${ }^{\wedge}$ ) Chapter 6.4.1.20.6.3.26 "Attribute 'monitoring'" on page 2295

When you doing object-oriented programming and using the inheritance (keyword EXTENDS) of POUs, you can get support as follows:

When you insert an action, a property, a method, or a transition below a POU derived from a base POU, the "Add ..." dialog opens. Then the input field for the name extends to a list box. The list box contains a valid selection from the actions, properties, methods, or transitions available in the base POU. Now you can, for example, easily accept a method of the base POU and then adapt it to the derived function of the POU.

Methods and properties with the access modifier PRIVATE are not listed here because they are also not inherited. Methods and properties with the access modifier PUBLIC automatically get a blank access modifier field when accepting into the derived POU, which means the same thing functionally.


See also

- \& Chapter 6.4.1.9.22 "Object-Oriented Programming" on page 1929
- \& Chapter 6.4.1.21.2.21.10 "Object 'Action"' on page 2497
- \# Chapter 6.4.1.21.2.21.9 "Object 'Property"' on page 2493
- \& Chapter 6.4.1.21.2.21.6 "Object 'Method'" on page 2485
- $\&$ Chapter 6.4.1.21.2.21.11 "Object 'Transition"' on page 2499


## Object 'Action'

Symbol: 略
Implement more program code in an action. You can implement this program code as the base implementation in another language. The base implementation is a function block or a program where you inserted the action.
An action does not have its own declaration and it works with the data from the base implementation. This means that the action uses the input and output variables and the local variables from its base implementation.
Add an "Action" to a function block or program by clicking "Project $\boldsymbol{\rightarrow}$ Add Object $\boldsymbol{\rightarrow}$ Action".

Table 439: "Add Action"

| "Name" | Name of the action |
| :--- | :--- |
| "Implementation language" | List box of implementation language |

Input support When you doing object-oriented programming and using the inheritance (keyword EXTENDS) of when generating inheriting POUs POUs, you can get support as follows:
When you insert an action, a property, a method, or a transition below a POU derived from a base POU, the "Add ..." dialog opens. Then the input field for the name extends to a list box. The list box contains a valid selection from the actions, properties, methods, or transitions available in the base POU. Now you can, for example, easily accept a method of the base POU and then adapt it to the derived function of the POU.
Methods and properties with the access modifier PRIVATE are not listed here because they are also not inherited. Methods and properties with the access modifier PUBLIC automatically get a blank access modifier field when accepting into the derived POU, which means the same thing functionally.


## See also

- ② Chapter 6.4.1.9.22 "Object-Oriented Programming" on page 1929
- $\quad$ 出 Chapter 6.4.1.21.2.21.10 "Object 'Action'" on page 2497
- ${ }^{\bullet}$, Chapter 6.4.1.21.2.21.9 "Object 'Property'" on page 2493
- ${ }^{\circledR}$ Chapter 6.4.1.21.2.21.6 "Object 'Method'" on page 2485
- $\Leftrightarrow$ Chapter 6.4.1.21.2.21.11 "Object 'Transition'" on page 2499

Calling an | Syntax: |
| :--- |
| action |
|  | <program>. <action> or <FB instance>. <action>

To call an action from only within the base implementation, you only have to provide the action name.
Examples Calling a "Reset" action from another POU The call is not executed from the base implementa-
tion.
Declaration:
PROGRAM PLC_PRG
PROGRAM PLC_PRG
VAR
VAR
Inst : Counter;
Inst : Counter;
END VAR
END VAR
Calling a "Reset" action from an IL POU
CAL Inst.Reset(In := FALSE)
LD Inst.Out
ST ERG
Calling a "Reset" action from an ST POU
Inst.Reset(In := FALSE);
Inst.Reset(In := FALSE);
Erg := Inst.out;
Calling a "Reset" action from an FBD POU


Actions are used frequently in the SFC implementation language.


See also

- ${ }^{4}$ Chapter 6.4.1.20.1.5.8.2 "SFC Element 'Action"" on page 2075


## Object 'Transition'

Symbol: 鱀
The object can be used as a transition element in a program block implemented in SFC.
See also

- $\Leftrightarrow$ Chapter 6.4.1.20.1.5.8.1 "SFC elements 'Step' and 'Transition"' on page 2073

Input support when generating inheriting POUs

When you doing object-oriented programming and using the inheritance (keyword EXTENDS) of POUs, you can get support as follows:
When you insert an action, a property, a method, or a transition below a POU derived from a base POU, the "Add ..." dialog opens. Then the input field for the name extends to a list box. The list box contains a valid selection from the actions, properties, methods, or transitions available in the base POU. Now you can, for example, easily accept a method of the base POU and then adapt it to the derived function of the POU.

Methods and properties with the access modifier PRIVATE are not listed here because they are also not inherited. Methods and properties with the access modifier PUBLIC automatically get a blank access modifier field when accepting into the derived POU, which means the same thing functionally.


See also

- \& Chapter 6.4.1.9.22 "Object-Oriented Programming" on page 1929
- $\triangleq$ Chapter 6.4.1.21.2.21.10 "Object 'Action'" on page 2497
-     * Chapter 6.4.1.21.2.21.9 "Object 'Property"' on page 2493
- ${ }^{\circ}$ Chapter 6.4.1.21.2.21.6 "Object 'Method"' on page 2485
- \&Chapter 6.4.1.21.2.21.11 "Object 'Transition"' on page 2499


## Object 'POUs for Implicit Checks'

## General

You can add these special POUs to an application to equip them with implicit monitoring functions. At runtime, these functions check the limits of arrays or subrange types, the validity of pointer addresses, and division by zero. Please note: This option can be disabled for devices that are already equipped with these kinds of monitoring blocks by a special implicit library.
The command "Add Object $\rightarrow$ POU for Implicit Checks" is used for adding to the application. The command opens the "Add POU for Implicit Checks" dialog where you can select a monitoring function type (see table below). Depending on the monitoring function, you have to adapt the implementation code or create it yourself from scratch.

To prevent multiple inclusions, monitoring functions that have already been inserted are disabled in the "Add POU for Implicit Checks" dialog.

## NOTICE!

To get the feature for monitoring functions, do not edit their declaration part.
However, you are permitted to add local variables.
After removing an implicit monitoring function (example: Check Bounds) from the project, only a download is possible, not an online change. A corresponding message is issued.

By default, CODESYS does not run implicit checks for function blocks from libraries used in the application. However, you can extend the check to the libraries by opening the "Properties" dialog of the application and specifying the compiler definition checks_in_libs in the "Compiler-Defines" field in the "Build" tab. This definition affects implementation libraries (*. library) only, not protected libraries (*. compiled-library).

You can use the "no_check" attribute to deactivate the check for special POUs in the project.

Table 440: "Available Functions"

| Monitoring function | Type |
| :---: | :---: |
| "Check Bounds" | "Bound Checks" <br> Appropriate handling of bound violations; such handling includes setting flags or changing field indices. |
| "CheckDivDInt" | "Division checks": <br> Monitors the divisor value to avoid division by zero. |
| "CheckDivLInt" |  |
| "CheckDivReal" |  |
| "CheckDivLReal" |  |
| "CheckRangeSigned" | "Range checks": <br> Monitors the range limit of a subrange type in runtime mode. Valid for data types DINT/UDINT. |
| "CheckRangeUnsigned" |  |
| "CheckLRangeSigned" | "L-range checks": <br> Monitors the range limit of a subrange type in runtime mode. Valid for data types LINT/ULINT. |
| "CheckLRangeUnsigned" |  |
| "CheckPointer" | "Pointer checks" |
|  | You are responsible for filling in this function completely with implementation code. Refer to the help page for "POU 'CheckPointer'". The function should monitor whether the passed pointer reference a valid memory address, and whether the orientation of the referenced memory area matches the variable type to which the pointer refers. If both conditions are fulfilled, then the pointer is returned. If not, then CheckPointer should complete an appropriate error handling. CheckPointer monitors the same way as variables of type REFERENCE TO. |

See also

- « Chapter 6.4.1.9.21 "Using POUs for implicit checks" on page 1928
- $\quad>$ Chapter 6.4.1.21.2.22.2 "POU 'CheckBounds'" on page 2502
- $\begin{gathered} \\ \text { • }\end{gathered}$ Chapter 6.4.1.21.2.22.3 "POU 'CheckDivInt'" on page 2505
- ${ }^{*}$ Chapter 6.4.1.21.2.22.4 "POU 'CheckDivLInt"' on page 2505
- ${ }^{2} \gg$ Chapter 6.4.1.21.2.22.6 "POU 'CheckDivLReal'" on page 2507
- $\Leftrightarrow$ Chapter 6.4.1.21.2.22.5 "POU 'CheckDivReal"' on page 2506
- « Chapter 6.4.1.21.2.22.11 "POU 'CheckPointer'" on page 2513
- $y^{*}>$ Chapter 6.4.1.21.2.22.7 "POU 'CheckRangeSigned'" on page 2508
- « Chapter 6.4.1.21.2.22.9 "POU 'CheckRangeUnsigned"" on page 2511
- $\Leftrightarrow$ Chapter 6.4.1.21.2.22.8 "POU 'CheckLRangeSigned'" on page 2510
- ${ }^{\bullet}$ ) Chapter 6.4.1.21.2.22.10 "POU 'CheckLRangeUnsigned'" on page 2512
- $\Leftrightarrow$ Chapter 6.4.1.21.4.11.5 "Dialog 'Properties' - 'Build"' on page 2755
- $\quad$ ² Chapter 6.4.1.20.6.3.29 "Attribute 'no_check'" on page 2298


## POU 'CheckBounds'

Functions for The task of this monitoring function is to handle bound violations appropriately. Examples of Bound Checks: reactions to violations include setting error flags and changing the value of the array index. The CheckBounds check is performed only for one variable array index. An incorrect constant array index causes a compiler error. CODESYS calls the function implicitly when values are assigned to an ARRAY variable.

After inserting the function, you receive automatically generated code in the declaration and implementation parts. See below.

## CAUTION!

To obtain the feature for monitoring functions, do not edit the declaration part. However, you are permitted to add local variables.

Declaration part // Automatically generated code: DO NOT EDIT FUNCTION CheckBounds : DINT VAR_INPUT
index, lower, upper: DINT;
END_VAR
Implementation // This automatically generated code is a suggested implementation.
IF index < lower THEN CheckBounds := lower;
ELSIF index > upper THEN
CheckBounds := upper;
ELSE
CheckBounds := index;
END_IF
(* It is also possible to set a breakpoint, log messages or e.g. to halt on an exception:
Add CmpApp.library, SysExcept.library and SysTypes2_Itf as newest. Declaration:
VAR
_pApp : POINTER TO CmpApp.APPLICATION; _result : SysTypes.RTS_IEC_RESULT;
END_V̄AR
Implementation:
_pApp := AppGetCurrent(pResult:=_result);
$\overline{\mathrm{I}} \mathrm{F}$ index < lower THEN
CheckBounds := lower;
IF _pApp <> 0 THEN
AppGenerateException(pApp:=_pApp,
ulException:=RtsExceptions.RTSEXCPT_ARRAYBOUNDS);
END_IF
ELSIF index $>$ upper THEN
CheckBounds := upper;
IF _pApp <> 0 THEN
AppGenerateException(pApp:=_pApp,

```
ulException:=RtsExceptions.RTSEXCPT_ARRAYBOUNDS);
    END_IF
ELSE
    CheckBounds := index;
END_IF
*)
```

When the "CheckBounds" function is called, it receives the following input parameters:

- index: Index of the array element
- lower: Lower limit of the array range
- upper: Upper limit of the array range

The return value is the index of the array element, as long as it is within a valid range. If not, then the CODESYS returns either the upper or lower limit, depending on which threshold was violated.

```
Example: Cor- In the sample program below, the index falls short of the defined lower limit of the a array.
rection of the
access to an
array outside
the defined
array bounds
```

```
PROGRAM PLC_PRG
```

PROGRAM PLC_PRG
VAR
VAR
a: ARRAY[0..7] OF BOOL;
a: ARRAY[0..7] OF BOOL;
b: INT:=10;
b: INT:=10;
END_VAR
END_VAR
a [b]:=TRUE;

```
    a [b]:=TRUE;
```

In this example, the CheckBounds function causes a to change the upper limit of the array range index to 10 . The value TRUE is assigned then to the element a [7]. In this way, the function corrects array access outside of the valid array range.

```
Example:
exception
when array
limits are vio-
lated.
Declaration
part
Implementa-
tion part
Add the following libraries in the library manager of the application:
- CmpApp.library and SysExcept.library as placeholder libraries
- SysTypes2_Itfs.library with "Newest version always"
Add a "CheckBounds" object below the application and modify the specified code as shown below.
```

```
FUNCTION CheckBounds : DINT
```

FUNCTION CheckBounds : DINT
VAR_INPUT
VAR_INPUT
index, lower, upper: DINT;
index, lower, upper: DINT;
END_VAR
END_VAR
VAR
VAR
pApp : POINTER TO CmpApp.APPLICATION;
pApp : POINTER TO CmpApp.APPLICATION;
Result : ISystypes2.RTS_IEC_RESULT;
Result : ISystypes2.RTS_IEC_RESULT;
END_\overline{V}AR
END_\overline{V}AR
// This automatically generated code is a suggested implementation.
// This automatically generated code is a suggested implementation.
_pApp := AppGetCurrent(pResult := _Result);
_pApp := AppGetCurrent(pResult := _Result);
IF index < lower THEN
IF index < lower THEN
CheckBounds := lower;
CheckBounds := lower;
IF _pApp <> 0 THEN
IF _pApp <> 0 THEN
AppGenerateException(pApp := _pApp, ulException :=
AppGenerateException(pApp := _pApp, ulException :=
RtsExceptions.RTSEXCPT_ARRAYBOUNDS);
RtsExceptions.RTSEXCPT_ARRAYBOUNDS);
END_IF
END_IF
ELSIF index > upper THEN
ELSIF index > upper THEN
CheckBounds := upper;
CheckBounds := upper;
IF _pApp <> 0 THEN
IF _pApp <> 0 THEN
AppGenerateException(pApp:=_pApp,
AppGenerateException(pApp:=_pApp,
ulException:=RtsExceptions.RTSEXCPT_ARRAYBOUNDS);
ulException:=RtsExceptions.RTSEXCPT_ARRAYBOUNDS);
END_IF
END_IF
ELSE
ELSE
CheckBounds := index;
CheckBounds := index;
END_IF
END_IF
Program a "MAIN_PRG" object below the application with the contents shown below.

```
```

PROGRAM MAIN_PRG

```
PROGRAM MAIN_PRG
VAR
VAR
    xInit : BOOL;
    xInit : BOOL;
    arData : ARRAY[0..7] OF BYTE;
    arData : ARRAY[0..7] OF BYTE;
    i : INT;
    i : INT;
    dwAdr : DWORD;
    dwAdr : DWORD;
END_VAR
END_VAR
IF NOT xInit THEN
IF NOT xInit THEN
    // Required for CheckBounds
    // Required for CheckBounds
    xInit := TRUE;
    xInit := TRUE;
END_IF
END_IF
// Set i to a value > 7 or < 0
// Set i to a value > 7 or < 0
// Generates an exception in CheckBounds, user-defined
// Generates an exception in CheckBounds, user-defined
arData[i] := 11;
```

arData[i] := 11;

```

When you load and start this application, an exception will be thrown when array bounds are violated. Processing stops in "CheckBounds" so that the type of error can be detected.

\section*{See also}
- \({ }^{4}\) Chapter 6.4.1.9.21 "Using POUs for implicit checks" on page 1928
- \({ }^{\bullet}\) Chapter 6.4.1.21.2.8.9 "Tab 'Log'" on page 2437

\section*{POU 'CheckDivInt'}

Functions for To prevent division by zero, you can use the functions CheckDivInt, CheckDivLint, preventing division by zero:
CheckDivInt,
CheckDivLint,
CheckDivReal,
and
CheckDivLReal CheckDivReal, and CheckDivLReal. If you include these functions in the application, then they are called before each division operation in the code.


\section*{CAUTION!}

To obtain the feature for monitoring functions, do not edit the declaration section. However, you are permitted to add local variables.

\section*{The default implementation of CheckDivReal:} Declaration section:
```

// This is automatically generated code: DO NOT EDIT
FUNCTION CheckDivReal : REAL
VAR_INPUT
divisor:REAL;
END_VAR

```

Implementation section:
```

// This automatically generated code is a suggested implementation.
IF divisor = O THEN
CheckDivReal:=1;
ELSE
CheckDivReal:=divisor;
END_IF;

```

The DIV operator uses the output of the CheckDivReal function as a divisor. In the sample program below, CheckDivReal prevents division by 0 by changing the implicit value of the divisor a from " 0 " to 1 before the division operation is executed. Therefore, the division result is 799.
```

PROGRAM PLC_PRG
VAR
erg:REAL;
v1:REAL:=799;
d:REAL:=0;
END VAR
erg:= v1 / d;

```

See also
- \#ch Chapter 6.4.1.9.21 "Using POUs for implicit checks" on page 1928

\section*{POU 'CheckDivLInt'}

Functions for To prevent division by zero, you can use the functions CheckDivInt, CheckDivLint, preventing division by zero: CheckDivInt, CheckDivLint, CheckDivReal, and
CheckDivLReal

\section*{CAUTION!}

To obtain the feature for monitoring functions, do not edit the declaration section. However, you are permitted to add local variables.
```

The default
Declaration section:
implementation
of CheckDiv-
Real:

```
```

// This is automatically generated code: DO NOT EDIT
FUNCTION CheckDivReal : REAL
VAR_INPUT
divisor:REAL;
END_VAR
Implementation section:

```
```

// This automatically generated code is a suggested implementation.

```
// This automatically generated code is a suggested implementation.
IF divisor = 0 THEN
IF divisor = 0 THEN
    CheckDivReal:=1;
    CheckDivReal:=1;
ELSE
ELSE
    CheckDivReal:=divisor;
    CheckDivReal:=divisor;
END_IF;
```

END_IF;

```

The DIV operator uses the output of the CheckDivReal function as a divisor. In the sample program below, CheckDivReal prevents division by 0 by changing the implicit value of the divisor d from "0" to 1 before the division operation is executed. Therefore, the division result is 799.
```

PROGRAM PLC_PRG
VAR
erg:REAL;
v1:REAL:=799;
d:REAL:=0;
END_VAR
erg:= v1 / d;

```

See also
- \# Chapter 6.4.1.9.21 "Using POUs for implicit checks" on page 1928

\section*{POU 'CheckDivReal'}

Functions for To prevent division by zero, you can use the functions CheckDivInt, CheckDivLint, preventing divi- CheckDivReal, and CheckDivLReal. If you include these functions in the application, then sion by zero: CheckDivInt, CheckDivLint, CheckDivReal, and
CheckDivLReal
they are called before each division operation in the code.


\section*{CAUTION!}

To obtain the feature for monitoring functions, do not edit the declaration section. However, you are permitted to add local variables.
```

The default Declaration section:
implementation
of CheckDiv-
Real:
// This is automatically generated code: DO NOT EDIT
FUNCTION CheckDivReal : REAL
VAR_INPUT
divisor:REAL;
END_VAR
Implementation section:

```
```

// This automatically generated code is a suggested implementation.

```
// This automatically generated code is a suggested implementation.
IF divisor = 0 THEN
IF divisor = 0 THEN
        CheckDivReal:=1;
        CheckDivReal:=1;
ELSE
ELSE
        CheckDivReal:=divisor;
        CheckDivReal:=divisor;
END_IF;
END_IF;
The DIV operator uses the output of the CheckDivReal function as a divisor. In the sample program below, CheckDivReal prevents division by 0 by changing the implicit value of the divisor d from " 0 " to 1 before the division operation is executed. Therefore, the division result is 799.
```

```
PROGRAM PLC_PRG
```

PROGRAM PLC_PRG
VAR
VAR
erg:REAL;
erg:REAL;
v1:REAL:=799;
v1:REAL:=799;
d:REAL:=0;
d:REAL:=0;
END_VAR
END_VAR
erg== v1 / d;

```
erg== v1 / d;
```


## See also

- ※ Chapter 6.4.1.9.21 "Using POUs for implicit checks" on page 1928


## POU 'CheckDivLReal'

Functions for To prevent division by zero, you can use the functions CheckDivInt, CheckDivLint, preventing divi- CheckDivReal, and CheckDivLReal. If you include these functions in the application, then sion by zero:
CheckDivInt, CheckDivLint, CheckDivReal, and
CheckDivLReal
they are called before each division operation in the code.


## CAUTION!

To obtain the feature for monitoring functions, do not edit the declaration section. However, you are permitted to add local variables.

```
The default Declaration section:
implementation
of CheckDiv-
Real:
// This is automatically generated code: DO NOT EDIT
FUNCTION CheckDivReal : REAL
VAR_INPUT
        divisor:REAL;
END_VAR
Implementation section:
```

```
// This automatically generated code is a suggested implementation.
```

// This automatically generated code is a suggested implementation.
IF divisor = 0 THEN
IF divisor = 0 THEN
CheckDivReal:=1;
CheckDivReal:=1;
ELSE
ELSE
CheckDivReal:=divisor;
CheckDivReal:=divisor;
END_IF;
END_IF;
The DIV operator uses the output of the CheckDivReal function as a divisor. In the sample program below, CheckDivReal prevents division by 0 by changing the implicit value of the divisor d from " 0 " to 1 before the division operation is executed. Therefore, the division result is 799.

```
```

PROGRAM PLC_PRG

```
PROGRAM PLC_PRG
VAR
VAR
        erg:REAL;
        erg:REAL;
        v1:REAL:=799;
        v1:REAL:=799;
        d:REAL:=0;
        d:REAL:=0;
END_VAR
END_VAR
erg:= v1 / d;
```

erg:= v1 / d;

```

\section*{See also}
- « Chapter 6.4.1.9.21 "Using POUs for implicit checks" on page 1928

\section*{POU 'CheckRangeSigned'}

Function for monitoring the range limits of a subrange type of type DINT.

This monitoring function is responsible for the appropriate handling violations to range limits. Examples of reactions to violations include setting error flags and changing values. The functions are called implicitly when a value is assigned to a subrange type variable.

\section*{CAUTION!}

To obtain the feature for monitoring functions, do not edit the declaration section. However, you are permitted to add local variables.

When the function is called, it receives the following input parameters:
- value: Value that should be assigned to the subrange type variables
- lower: Lower range limit
- upper: Upper range limit

The return value is the assignment value as long as it is within the valid range. If not, then either the upper or lower limit is returned, depending on which threshold was violated.
For example, the assignment \(\mathrm{i}:=10{ }^{\mathrm{y}} \mathrm{y}\) is replaced implicitly by \(\mathrm{i}:=\) CheckRangeSigned (10*y, -4095, 4095);

If \(y\) is " 1000 ", then " \(10 * 1000=10000\) " is not assigned to \(i\) like in the original code. Instead, the upper range limit of "4095" is assigned.

The same is true for CheckRangeUnsigned function.

\section*{NOTICE!}

If functions are not available, then the subrange is not checked for the respective variables at runtime. In this case, you can assign any value between -2147483648 and +2147483648 (or between 0 and 4294967295 ) to a variable of subrange type DINT/UDINT. You can assign any value between -9223372036854775808 and +9223372036854775807 (or between 0 and 18446744073709551615) to a variable of a subrange type LINT/ULINT.

\section*{CAUTION!}

Linking area monitoring functions can lead to endless loops. For example, an endless loop can occur if the counter variable of a FOR loop is a subrange type and the counting range for the loop exits the defined subrange.

\section*{Example of an endless loop:}
```

VAR
ui : UINT (0..10000);
END_VAR
FOR ui:=0 TO 10000 DO
END_FOR

```

The program never exits the FOR loop because the CheckRangeSigned monitoring function prevents ui from being set to a value greater than 10000.
```

Example for The assignment of a value to a DINT variable of a signed subrange type is a condition for
CheckRangeSi automatically calling the CheckRangeSigned. This function restricts the assignment value
gned to the subrange as defined in the variables declaration. The default implementation of the function in ST is as follows:
Declaration section:

```
Implementation:
```

```
// This is automatically generated code: DO NOT EDIT
```

// This is automatically generated code: DO NOT EDIT
FUNCTION CheckRangeSigned : DINT
FUNCTION CheckRangeSigned : DINT
VAR_INPUT
VAR_INPUT
value, lower, upper: DINT;
value, lower, upper: DINT;
END_VAR

```
END_VAR
```

```
// This automatically generated code is a suggested implementation.
```

// This automatically generated code is a suggested implementation.
IF (value < lower) THEN
IF (value < lower) THEN
CheckRangeSigned := lower;
CheckRangeSigned := lower;
ELSEIF(value > upper) THEN
ELSEIF(value > upper) THEN
CheckRangeSigned := upper;
CheckRangeSigned := upper;
ELSE
ELSE
CheckRangeSigned := value;
CheckRangeSigned := value;
END_VAR

```
END_VAR
```

See also

- « Chapter 6.4.1.9.21 "Using POUs for implicit checks" on page 1928


## POU 'CheckLRangeSigned'

Function for monitoring the range limits of a subrange type of type LINT.
For an implementation example of range monitoring, refer to the help page for the CheckRangeSigned function.

This monitoring function is responsible for the appropriate handling violations to range limits. Examples of reactions to violations include setting error flags and changing values. The functions are called implicitly when a value is assigned to a subrange type variable.

## CAUTION!

To obtain the feature for monitoring functions, do not edit the declaration section. However, you are permitted to add local variables.

When the function is called, it receives the following input parameters:

- value: Value that should be assigned to the subrange type variables
- lower: Lower range limit
- upper: Upper range limit

The return value is the assignment value as long as it is within the valid range. If not, then either the upper or lower limit is returned, depending on which threshold was violated.
For example, the assignment $i \quad:=10 * y$ is replaced implicitly by $i \quad:=$ CheckRangeSigned (10*y, -4095, 4095);
If $y$ is " 1000 ", then " $10 * 1000=10000$ " is not assigned to $i$ like in the original code. Instead, the upper range limit of "4095" is assigned.
The same is true for CheckRangeUnsigned function.

## NOTICE!

If functions are not available, then the subrange is not checked for the respective variables at runtime. In this case, you can assign any value between -2147483648 and +2147483648 (or between 0 and 4294967295 ) to a variable of subrange type DINT/UDINT. You can assign any value between -9223372036854775808 and +9223372036854775807 (or between 0 and 18446744073709551615) to a variable of a subrange type LINT/ULINT.

## CAUTION!

Linking area monitoring functions can lead to endless loops. For example, an endless loop can occur if the counter variable of a FOR loop is a subrange type and the counting range for the loop exits the defined subrange.

```
Example of an
endless loop:
```

```
VAR
```

VAR
ui : UINT (0..10000);
ui : UINT (0..10000);
END_VAR
END_VAR
FOR ui:=0 TO 10000 DO
FOR ui:=0 TO 10000 DO
END_FOR

```
        END_FOR
```

The program never exits the FOR loop because the CheckRangeSigned monitoring function prevents ui from being set to a value greater than 10000.

## See also

- y Chapter 6.4.1.9.21 "Using POUs for implicit checks" on page 1928
- *y Chapter 6.4.1.21.2.22.7 "POU 'CheckRangeSigned'" on page 2508


## POU 'CheckRangeUnsigned'

Function for monitoring the range limits of a subrange type of type UDINT.
For an implementation example of range monitoring, refer to the help page for the CheckRangeSigned function.

This monitoring function is responsible for the appropriate handling violations to range limits. Examples of reactions to violations include setting error flags and changing values. The functions are called implicitly when a value is assigned to a subrange type variable.

## CAUTION!

To obtain the feature for monitoring functions, do not edit the declaration section. However, you are permitted to add local variables.

When the function is called, it receives the following input parameters:

- value: Value that should be assigned to the subrange type variables
- lower: Lower range limit
- upper: Upper range limit

The return value is the assignment value as long as it is within the valid range. If not, then either the upper or lower limit is returned, depending on which threshold was violated.

For example, the assignment i $:=10 * y$ is replaced implicitly by i $:=$
CheckRangeSigned(10*y, -4095, 4095);
If $y$ is " 1000 ", then " $10 * 1000=10000$ " is not assigned to $i$ like in the original code. Instead, the upper range limit of "4095" is assigned.

The same is true for CheckRangeUnsigned function.

## NOTICE!

If functions are not available, then the subrange is not checked for the respective variables at runtime. In this case, you can assign any value between -2147483648 and +2147483648 (or between 0 and 4294967295 ) to a variable of subrange type DINT/UDINT. You can assign any value between -9223372036854775808 and +9223372036854775807 (or between 0 and 18446744073709551615) to a variable of a subrange type LINT/ULINT.

```
Example of an
endless loop:
```

```
VAR
```

VAR
ui : UINT (0..10000);
ui : UINT (0..10000);
END_VAR
END_VAR
FOR ui:=0 TO 10000 DO
FOR ui:=0 TO 10000 DO
END_FOR

```
        END_FOR
```

The program never exits the FOR loop because the CheckRangeSigned monitoring function prevents ui from being set to a value greater than 10000.

See also

- y Chapter 6.4.1.9.21 "Using POUs for implicit checks" on page 1928
- ${ }^{*}>$ Chapter 6.4.1.21.2.22.7 "POU 'CheckRangeSigned'" on page 2508


## POU 'CheckLRangeUnsigned'

Function for monitoring the range limits of a subrange type of type ULINT.
For an implementation example of range monitoring, refer to the help page for the CheckRangeSigned function.

This monitoring function is responsible for the appropriate handling violations to range limits. Examples of reactions to violations include setting error flags and changing values. The functions are called implicitly when a value is assigned to a subrange type variable.

## CAUTION!

To obtain the feature for monitoring functions, do not edit the declaration section. However, you are permitted to add local variables.

When the function is called, it receives the following input parameters:

- value: Value that should be assigned to the subrange type variables
- lower: Lower range limit
- upper: Upper range limit

The return value is the assignment value as long as it is within the valid range. If not, then either the upper or lower limit is returned, depending on which threshold was violated.
For example, the assignment i $:=10 * y$ is replaced implicitly by i $:=$
CheckRangeSigned(10*y, -4095, 4095);
If $y$ is " 1000 ", then " $10 * 1000=10000$ " is not assigned to i like in the original code. Instead, the upper range limit of "4095" is assigned.

The same is true for CheckRangeUnsigned function.

## NOTICE!

If functions are not available, then the subrange is not checked for the respective variables at runtime. In this case, you can assign any value between -2147483648 and +2147483648 (or between 0 and 4294967295 ) to a variable of subrange type DINT/UDINT. You can assign any value between -9223372036854775808 and +9223372036854775807 (or between 0 and 18446744073709551615) to a variable of a subrange type LINT/ULINT.

## CAUTION!

Linking area monitoring functions can lead to endless loops. For example, an endless loop can occur if the counter variable of a FOR loop is a subrange type and the counting range for the loop exits the defined subrange.

```
Example of an
endless loop:
```

```
VAR
```

VAR
ui : UINT (0..10000);
ui : UINT (0..10000);
END_VAR
END_VAR
FOR ui:=0 TO 10000 DO
FOR ui:=0 TO 10000 DO
END FOR

```
    END FOR
```

The program never exits the FOR loop because the CheckRangeSigned monitoring function prevents ui from being set to a value greater than 10000 .

See also

- « Chapter 6.4.1.9.21 "Using POUs for implicit checks" on page 1928
- ${ }^{*}>$ Chapter 6.4.1.21.2.22.7 "POU 'CheckRangeSigned'" on page 2508


## POU 'CheckPointer'

## Monitoring function for pointers (Checkpointer)

Use this function to monitor the memory access of pointers in runtime mode. As opposed to other monitoring functions, a standard suggestion does not exist for the implementation of CheckPointer. You must define an implementation according to your own requirements.

The CheckPointer function should check whether the returned pointer references a valid memory address; monitors whether the orientation of the referenced memory range matches the variable type that the pointer refers to. If both conditions are fulfilled, then the pointer is returned. If not, then the function should complete an appropriate error handling.

## CAUTION!

To obtain the feature for monitoring functions, do not edit the declaration section. However, you are permitted to add local variables.

[^39]
## NOTICE!

For complier version 3.5.7.40 and later, the implicit check function "Checkpointer" also acts on REFERENCE variables in the same way as on pointer variables.

## Model

Declaration:

```
// Automatically generated code: DO NOT EDIT
FUNCTION CheckPointer : POINTER TO BYTE
VAR_INPUT
        ptToTest : POINTER TO BYTE;
        iSize : DINT;
        iGran : DINT;
        bWrite: BOOL;
END_VAR
// Not a standard implementation. Please add your own code here.
CheckPointer := ptToTest;
```

Implementa-
tion: (incom-
plete)

When the function is called, CODESYS provides the following input parameters:

- ptToTest: Target address of the pointer
- iSize: Size of the referenced variable; the data type of iSize must be compatible with INT and cover the dimensional scope of the variables.
- iGran: Granularity of the referenced size; this is the largest non-structured data type contained in the referenced variables; the data type of iGran must be compatible with INT
- bWrite: Access type (TRUE = write access, FALSE = read access); the data type of bWrite must be BOOL

When the result of the check is positive, the unchanged pointer is returned (ptToTest).

## See also

- « Chapter 6.4.1.9.21 "Using POUs for implicit checks" on page 1928


## Object 'Project Settings'

Symbol:
Function: This object contains the configuration of the project.

## Call:

- "Project $\rightarrow$ Project Settings"
- Double-click on the object in the device tree

CODESYS saves the project settings directly in the project. If, for example, you transfer a project to another system, the "Project Settings" object is also transferred with it without a project archive being required.
The project settings are valid project-wide and offer setting possibilities for various categories such as "AS" or "Users and Groups". The available categories vary, depending on which software packages you have installed via the package manager.
See also

- ${ }^{*}>$ Chapter 6.4.1.3.4.3 "Making project settings" on page 1814
- ${ }^{\text {• }}$ - Chapter 6.4.1.21.3.9.4 "Command 'Package Manager'" on page 2655
- « ${ }^{*}$ Chapter 6.4.1.21.4.12.2 "Dialog 'Project Settings' - 'SFC"" on page 2767
- $\sum^{\circledR}$ Chapter 6.4.1.21.4.12.3 "Dialog 'Project Settings' - 'Users and Groups'" on page 2768
- $y^{\Perp}$ Chapter 6.4.1.21.4.12.4 "Dialog Box 'Project Settings' - 'Compileoptions'" on page 2769
- $\boldsymbol{y}^{\prime}$ Chapter 6.4.1.21.4.12.5 "Dialog Box 'Project Settings' - 'Compiler Warnings'" on page 2770
- $\Leftrightarrow$ Chapter 6.4.1.21.4.12.6 "Dialog 'Project Settings' - 'Source Download"' on page 2770
- $\underbrace{*}$ Chapter 6.4.1.21.4.12.7 "Dialog 'Project Settings' - 'Page Setup'" on page 2772
- $\Leftrightarrow$ Chapter 6.4.1.21.4.12.8 "Dialog 'Project Settings' - 'Security"' on page 2772
- $\Rightarrow$ Chapter 6.4.1.21.4.12.9 "Dialog 'Project Settings' - 'Static Analysis Light"' on page 2773
- « Chapter 6.4.1.21.4.12.10 "Dialog 'Project Settings' - 'Visualization'" on page 2776
- $\quad$ Chapter 6.4.1.21.4.12.11 "Dialog 'Project Settings' - 'Visualization Profile"" on page 2777


## Object 'Project Information'

Symbol: i
Function: The object contains the properties, meta-information, and project information. With this, you can check the authorship and integrity of the project.

## Call

- Double-click the object in the device tree
- Menu bar: "Project $\rightarrow$ Project Information"

Requirement: CODESYS creates the object when you click "Project $\rightarrow$ Project Information", and the dialog opens.

CODESYS saves the project information directly in the project. For example, if you transfer a project to another system, then the "Project Information" object is also transferred. You do not need a project archive.

Tab 'File' The tab displays the properties of the project file and their attributes. You cannot edit these attributes. They correspond to the file properties of Windows Explorer.

Tab 'Summary' The tab contains general information and meta-information of the project file. CODESYS uses this information to create keys on the "Properties" tab. For example, if the name Company_A is specified in "Company", then the Company key with the value Company_A is provided on the "Properties" tab.

## NOTICE!

If you save your project as a library project, then you should pay attention to the guidelines for library developers (Library Development Summary).

| For a library project, a "Company", a "Title", and a "Version" must be specified to install the library. |  |
| :--- | :--- |
| "Company" | Name of the company (example: Company_A). |
| "Title" | Title of the project (example Automation_A). |
| "Version" | Version of the project (example: 0.0 .0 .1 ). |
| "Released" | V: Activates protection from modification. <br> Result: If you edit the project now, then a dialog prompt opens to confirm <br> whether you really want to change the project. If you reply to this prompt <br> one time by clicking "Yes", then no additional prompts appear for more editing <br> actions. |


| "Categories" | Categories of the library project, according to which you can sort in the "Library <br> Repository" dialog. If no category is specified, then the category "Other" is <br> assigned to the library. <br> The categories originate from one or more external description files in XML <br> format. However, they can also originate from a library project that has already <br> been created. <br> Requirement: The project is a library project. <br> (...: The "Library Categories" dialog opens where you can add library categories. |
| :--- | :--- |
| "Default namespace" | If you do not define a standard namespace here, then the name of the library file <br> is applied automatically as the namespace. |
| "Author" | Author of the project (example: Arnold Best). |
| "Description" | Example: For internal use only |

See also

- ${ }^{\text {r }}$ Chapter 6.4.1.21.3.2.7 "Command 'Save Project as Compiled Library'" on page 2556
- $\Leftrightarrow$ Chapter 6.4.1.17.3 "Information for Library Developers" on page 2035

Table 441: Dialog "Library Categories"

| List of categories | List of the categories that are assigned to the library project. They can originate <br> from several sources. After you specify all desired categories, click "OK" to <br> confirm. |
| :--- | :--- |
| Button "Add" | The "From Description File" and "From Other Library" commands appear. |
| Button "Remove" | CODESYS removes the selected category. |
| Command "From Description <br> File" | The "Select Description File" dialog opens for you select a description file <br> (*. libcat. xml). The file contains command categories. When you click <br> "Open", CODESYS accepts the categories. |
| Command "From Other <br> Library" | The "Select Library" dialog opens, where you select a library with command <br> categories to be accepted. When you click "Open", CODESYS accepts the cate- <br> gories. |
| Button "OK" | CODESYS provides the categories as project information and displays it in the <br> "Library Categories" field. |

See also

- $乡$ Chapter 6.4.4.2 "Guidelines for creating libraries" on page 2852
- $\triangleq$ Chapter 6.4.1.21.3.9.5 "Command 'Library Repository"' on page 2657

Tab 'Properties' On this tab, you can define keys that you can control externally from user-specific programs.


| "Key" | Name of the key. Specify any string of text for the new key, or select an existing <br> key from the "Properties" table. |
| :--- | :--- |
| "Type" | Data type of the key. Possible types: "Text", "Date", "Number", "Boolean", <br> "Version". |


| "Value" | Value of the key in permitted format <br> - "Text": Any character string <br> - "Date": Example: Friday, January 1, 2016 00:00:00. Minimum entry for the date: 1.1 <br> - "Number": Integer in Integer32 format with or without a sign (example: -32500). <br> - "Boolean": True or False, capitalization irrelevant. <br> - "Version": Examples: 1.1,1.0.1.0, maximum four figures. |
| :---: | :---: |
| "Add" | Adds the new defined key to the "Properties" table. |
| "Modify" | Saves the change made for the key selected in the "Properties" table. |
| "Remove" | Removes the key selected in the "Properties" table. |


| "Properties" | List of the properties that are defined as keys. CODESYS creates keys automati- <br> cally for the information in the "Summary" tab. <br> Click a key to edit it in the input fields above the list. |
| :--- | :--- |

See also

- ⓨ Chapter 6.4.4.2 "Guidelines for creating libraries" on page 2852
- Using the Symbol Library in the Visualization
- $\quad{ }^{n} \boldsymbol{y}$ Chapter 6.4.1.21.4.11.18 "Dialog 'Properties' - 'Image Pool"' on page 2764

Tab 'Statistics' The dialog provides statistical information about the number of objects of the individual type or use in the project.

Tab 'Licensing' The dialog is for the license protection of libraries.

## CAUTION!

You can protect only compiled libraries in this way.

Table 442: "Variables"

| "Activate dongle licensing" | $\boxed{ }$ : The library requires a dongle with a license to use it. |
| :--- | :--- |
| "Firm code" | License information that must be supplied from the dongle for using the library |
| "Product code" |  |
| "Acter. |  |

See also

-     * Chapter 6.4.4.2 "Guidelines for creating libraries" on page 2852

Tab 'Signing' This tab is displayed only for existing libraries whose signing has been created with this tab. This tab is no longer visible for newly generated libraries.
When a certificate-signed library is created (possible as of CODESYS V3 SP15) and library compatibility with CODESYS < V3 SP15 is not set, the settings on this tab are disabled. In this case, the signing is done by means of a certificate that has to be assigned to the user profile in the security screen.

One method, which is not recommended but may be necessary in some case for compatibility with versions < V3 SP15, is the less secure signing of a library by means of a vendor-specific, one-time key in this dialog. Requirement: This key is available as a private key file (*. libpk) with an associated token. The user of the library also has to obtain this key in order to be able to check whether the last signing was actually performed by the library vendor.

| "Activate signing" | 局: CODESYS signs the library project with a single-use, manufacturer-specific <br> key. |
| :--- | :--- |
| "Private key file" | Location of the private key file *. libpk (example: D: $\backslash$ for lib developers <br> only $\backslash$ mycomp_libkey. libpk). |
| "Public key token" | Example: 427A5701DA3CF3CF <br> Requirement: A private key file is specified, and CODESYS has read and <br> entered the token. |
| "Create Private Key File" | CODESYS creates a new private key file. |

## See also

- ${ }^{\mu}$ Chapter 6.4.1.21.3.4.18 "Command 'Security Screen'" on page 2592

Options for creating blocks for accessing project information

| "Automatically generate | Note: The functions that are created with this option can be used only if the <br> runtime supports the WSTRING data type. If this is not the case, then you can <br> use the functions that were created automatically for the with the individual items <br> of the project information, at least in the application for accessing properties. <br> These functions are not registered in the runtime. <br> V: CODESYS creates POUs of the FUNCTION type in the "POUs" view, allowing |
| :--- | :--- |
|  | programmatic access to the project properties in the application. The function <br> blocks GetCompany, GetTitle and GetVers ion are created for the proper- <br> ties "Company", "Title" and "Version". <br> The following function blocks are available for user-defined properties: <br> - GetBooleanProperty: BOOL (TRUE/FALSE) <br> $\bullet \quad$ GetNumberProperty: DINT (numeric value) <br> - GetTextProperty: WSTRING (character string) |
| - GetTextProperty2: POINTER TO WSTRING (unlimited length) |  |
| $\bullet \quad$ GetVersionProperty: VERSION (version number as character string) |  |
| Note: Do not activate this option for standard libraries, because this can cause |  |
| problems on smaller systems due to the additional memory requirements. |  |
| Note: If a library also contains this project information POU, then you should use |  |
| the operator __POOL to make sure that this POU is accessed. |  |

See also

- $\boldsymbol{y}^{*}$ Chapter 6.4.1.3.4.2 "Retrieving and Editing Project Information" on page 1813
- $\quad$ - Chapter 6.4.1.3.4.3 "Making project settings" on page 1814
- $\quad$ \& Chapter 6.4.1.20.3.74 "Operator '__POOL"' on page 2217


## Object 'Recipe Manager'

Tab 'Storage' The recipe manager provides functions for maintaining user-defined variable lists, known as recipe definitions. The recipe definitions can be stored in recipe files on the PLC.

| "Storage Type" | "Textual": CODESYS saves the recipe in a readable Format with the configured columns and delimiters. <br> "Binary": CODESYS saves the recipe in a non-readable binary format. This format requires less storage space. <br> Note: You can read binary recipes again only if you have not changed the variable lists. |
| :---: | :---: |
| "File path" | <directory name>\} <br> Example: AllRecipes $\backslash$ <br> Path on the runtime system <br> Notes: <br> - The path has to end with a backslash ("l"). <br> - The path is usually a relative path on the target system in the directory of the runtime files (PlcLogic). <br> - Access to paths outside of the directory PlcLogic is not permitted on every controller. An absolute path for Windows systems can be selected by pressing the... button. <br> Example of the file path in the runtime system: PlcLogic/AllRecipes <br> CODESYS saves a file in this directory for each recipe when downloading to the PLC. The requirement is that you select the "Recipe management in the PLC" option. <br> The files are loaded to the recipe manager each time the application is restarted. |
| "File extension" | File extension for the recipe file in the format .<file extension> <br> The resulting default name for recipe files is in the form <recipe>.<recipe definition>.<file extension>. |
| "Separator" | Delimiters between the individual values in the saved file |
| "Available Columns" <br> "Selected Columns" | Defines the information that is saved and in which order in the recipe file |
| "Save as Default" | CODESYS uses the settings on the tab throughout the entire project for all other recipe managers. |

Tab 'General'

```
"Recipe management in the
PLC"
```

$\checkmark$ : Has to be selected for the user program or visualization elements to load recipes at runtime. If you transfer recipes to the PLC exclusively via the CODESYS program interface, then you can clear this option.

Table 443: "Save Recipe"
"Save recipe changes to recipe files automatically"

When "Recipe management in the PLC" is selected, there is the following option for saving the recipe:
$\checkmark$ : We recommend this option because it helps the recipe manager operate "normally". The recipe files on the PLC are updated automatically at runtime whenever a recipe is changed.

Table 444: "Load Recipe"
When "Recipe management in the PLC" is selected, there are the following two options for downloading from the PLC:
"Download only for exact match of the variable list"

Download variables with matching names"

The recipe is only downloaded if the file on the PLC contains all variables from the variable list of the recipe definition of the application and these are sorted in the same order. Additional entries at the end are ignored. If the required match does not exist, then the error status ERR_RECIPE_MISMATCH is set (RecipeManCommands.GetLastError).
: The recipe values are downloaded only for those variables that have the same name in the recipe definition of the application as in the recipe file on the PLC. If the variable lists differ in composition and sorting, then no error status is set.

In this way, recipe files can also be downloaded if variables in the file or in the recipe definition have been deleted.
"Overwrite existing recipes on download"
$\checkmark$ : If recipe files with the same name exist on the controller, then they are overwritten with the configured values from the project when the application is started. If the values from the existing recipe files should be loaded instead, then this option has to be disabled.
Requirement: The "Storage Type" is "Textual" and the "Save recipe changes to recipe files automatically" option is selected.

Table 445: "Write Recipe"
The following options are available for writing recipe values to the variables on the PLC:
"Limit the variable to $\mathrm{min} / \mathrm{max}$ when recipe value is out of the range"
"Do not write to a variable when the recipe value is out of the min/max range"

O: If the recipe contains a value that is outside of the range of values specified in the definition, then the defined minimum or maximum value is written to the PLC variable instead of this value.
: If the recipe contains a value that is outside of the range of values specified in the definition, then no value is written to the PLC variable. It retains its current value.

Table 446: "Read Recipe"
The following option is available for reading recipe values from the PLC into the recipe in the project:
"Check recipe for changes"

Always use the function block RecipeManCommands from RecipeManagement. library to read recipes. Never call the method cyclically. This is because each call can be written to the file system, which is time-intensive and burdens the controller. For example, a Raspberry protocol interpreter has a limited number of write cycles.
$\checkmark$ : With each method call, the current PLC variable values are first read into the recipe. Then the system checks whether the values have changed. Only if the values have changed is the recipe saved. This means that the recipe file is overwritten with the current recipes.
The option can be used in order to update the recipe file in the local file system only if recipe values have changed on the PLC. However, it affects performance because it generates additional code for checking.
. With each method call, the current PLC variable values are first read into the recipe. Then the recipe is written to the recipe file in the local file system.
Note: As the file system is written to each call, the controller can be very burdened.

Recipes during online mode

Table 447: Option "Save recipe changes to recipe files automatically" is selected.

| Menu commands | Behavior of the recipes <br> defined in the project | Behavior of the defined rec- <br> ipes at runtime |
| :--- | :--- | :--- |
| "Online $\rightarrow$ Reset Warm" | The recipes of all recipe def- <br> initions are downloaded with <br> "Online $\rightarrow$ Reset Cold" <br> "Onejects. | Dynamically generated rec- <br> ipes remain unchanged. |
| "Online $\rightarrow$ Download" $\rightarrow$ Reset Origin" | The application is removed from the PLC. If a download is <br> done again afterwards, then the recipes are restored as for an <br> online reset warm. |  |
| Shutdown and restart of the <br> PLC | After a restart, the recipes are downloaded again from the <br> automatically created files. This will restore the same state as <br> before shutdown. |  |
| "Online $\rightarrow$ Online Change" | The recipe values remain unchanged. In runtime mode, a <br> recipe can be changed only via the function block command <br> RecipeManCommands. |  |
| "Debug $\rightarrow$ Stop" | The recipes remain unchanged when the PLC is stopped or <br> started. |  |
| "Debug $\rightarrow$ Start" |  |  |

Table 448: Option "Save recipe changes to recipe files automatically" is not selected.

| Actions | Recipes defined in the <br> project | Recipes defined at runtime |
| :--- | :--- | :--- |
| "Online $\rightarrow$ Reset Warm" | The recipes of all recipe def- <br> initions are downloaded with <br> the values from the current <br> projects. However, these are <br> set in the memory only. To <br> save recipes to a file, you <br> have to run the "Save Recipe" <br> command explicitly. | Dynamically generated rec- <br> ipes are lost. |
| "Online $\rightarrow$ Reset Cold" |  |  |$\quad$ Download" $\quad$| The application is removed |
| :--- |
| from the PLC. When a down- |
| load is performed afterwards, |
| the recipes are restored. |$\quad$| Dynamically generated rec- |
| :--- |
| ipes are lost. |


| Actions | Recipes defined in the <br> project | Recipes defined at runtime |
| :--- | :--- | :--- |
| Shutdown and restart of the <br> PLC | After the restart, the recipes are downloaded again from the <br> automatically created files. This will restore the same state as <br> before shutdown. |  |
| "Online $\rightarrow$ Online Change" | The recipe values remain unchanged. In runtime mode, a <br> recipe can be changed only via the function block command <br> RecipeManCommands. |  |
| "Debug $\rightarrow$ Stop" <br> "Debug $\rightarrow$ Start" | The recipes remain unchanged when the PLC is stopped or <br> started. |  |

See also

- \# Chapter 6.4.1.13.2.4 "Changing Values with Recipes" on page 2003
- ${ }^{*}$ Chapter 6.4.1.21.3.20.9 "Command 'Read and Save Recipe"' on page 2726
- $\Rightarrow$ Chapter 6.4.1.21.2.26 "Object 'Recipe Definition'" on page 2522
- Method Calls of the 'Recipe Management' Library


## Object 'Recipe Definition'

In the recipe definition (1), you define different data sets for the variables, which are termed recipes (2).
You can toggle the display of the recipe definition between the flat list view (3) and the structured view (4). In the structured view, CODESYS groups variables according to structure.


| "Type" | Entered automatically |
| :--- | :--- |
| "Name" | Optional |
| "Minimal Value" <br> "Maximal Value" | If the variable value is less than the "Minimal Value" or greater than the "Maximal <br> Value", then CODESYS sets the value to the "Minimal Value" or "Maximal <br> Value". |
| "Comment" | Additional information, for example the unit of the value. |
| "Current Value" | Current variable value; not shown in online mode |

Table 449: "Additional commands in the context menu in the structured view"

| "Add Sibling" | Adds a sibling variable to the recipe definition. |
| :--- | :--- |
| "Add Child" | Adds a child variable to the recipe definition. |

See also

- Chapter 6.4.1.13.2.4 "Changing Values with Recipes" on page 2003


## Object 'Symbol Configuration'

You can use the symbol configuration for creating symbol descriptions for project variables. Click "Project $\rightarrow$ Add Object" to add a symbol configuration object to the device tree. Then define specific presets. See dialog below: "Add Symbol Configuration".

Double-click the "Symbol Configuration" object to open the symbol configuration editor.

Dialog 'Add Function: This dialog is used to define the defaults for a "Symbol Configuration" object. Symbol Configuration'

Call: "Project $\rightarrow$ Add Object $\rightarrow$ Symbol Configuration" menu; context menu of the application object.

| "Include comments in XML" | Exports the symbol file with the comments assigned to the variables. |
| :--- | :--- |
| "Support OPC UA features" | Note: Availability and editability of this option depend on the device. |
|  | W: When downloading the symbol configuration, additional information is also <br> downloaded to the controller. The information below is necessary for operating <br> the OPC UA server. |
|  | - Base types of inherited function blocks |
|  | $\bullet \quad$ Contents of attributes that were assigned via compiler pragmas |
|  | $\bullet \quad$ Scopes (example: VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT) |

Table 450: "Client-Side Data Layout"

| For detailed information and examples of layout options, see the next section "Symbol Configuration Editor". |  |
| :--- | :--- |
| "Compatibility layout" | This setting is used for the compatibility of old projects. The data layout created <br> for the client is matched as much as possible to the layout created internally by <br> the compiler. |
| "Optimized layout" | Recommended for new projects. Calculates the output layout in optimized form <br> detached from the internal compiler layout. Does not generate any gaps for <br> unpublished elements and strictly fulfills the requirements for memory alignment <br> of the data types. Requires compiler version 3.5.7.0 or higher. |

Symbol configu- The editor includes a table with selected variables and a menu bar for editing. ration editor

Table 451: Menu bar

| ® "View" | You can use this button for activating and deactivating the following categories of variables used in the configuration editor: <br> - 圆 "Unconfigured from Project": Variables that have not been added to the symbol configuration, but are provided in the project. <br> - 追 "Unconfigured from Libraries": Variables that have not been added to the symbol configuration, but are provided in the project. <br> - \{\} "Symbols Exported via Attribute": This filter also lists the variables that have already been marked for export in the symbol file by means of the \{attribute 'symbol' := 'read'\} pragma. These symbols are displayed in gray. The "Attribute" column shows which access rights are set by the pragma. |
| :---: | :---: |
| \#3 "Build" | Compiles the project. Requirement for current preparation of variables in the configuration editor. |

- "Support OPC UA features":

Note: Availability and editability of this option depend on the device.
$\square$ : When downloading the symbol configuration, additional information is also downloaded to the controller. The information below is necessary for operating the OPC UA server. This currently includes the following information:

- Base types of inherited function blocks
- Contents of attributes that were assigned via compiler pragmas
- Scopes (example: VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT)
- <!> "Include comments in XML"
$\square$ : Exports the symbol file with the comments assigned to the variables.
- "Include Node Flags in XML"

V: The namespace node flags provide additional information about the origin of a node in the namespace. The node flags always in the symbol table when OPC UA is activated. However, its inclusion in the XML file can be deactivated as some defective parsers have problems with it.

- "Configure Comments and Attributes"

Opens the "Comments and Attributes" dialog. Here you configure the details of what should be included in the symbol configuration and XML file with respect to comments and attributes.

- "Configure synchronization with IEC tasks":

Opens the "Properties - <device name>" dialog, "Options" tab.
This setting allows for the symbolic clients (e.g. visualizations or database links based on the PLCHandler) to have consistent read/write access synchronized with the IEC tasks. For a detailed description of this setting, see the "Setting: Configure synchronization with IEC tasks" section below.
Note: Variable access which is synchronous with the IEC tasks can increase the jitter for all IEC applications on this device. Synchronized consistent access can interrupt the real-time capability.

- List box for defining the data layout type for the client of the symbol configuration:
Note: See the "Example of data layout types" section at the end of this help page.
- "Optimized layout": Recommend for new projects. Calculates the output layout in optimized form detached from the internal compiler layout. Does not generate any gaps for unpublished elements and strictly fulfills the requirements for memory alignment of the data types. Requires compiler version 3.5.7.0 or higher.
- "Compatibility layout": This setting is used for the compatibility of old projects. The data layout created for the client is matched as much as possible to the layout created internally by the compiler.
Due to the configuration possibilities of the symbol configuration which have grown over time, problematic offsets can still result. Causes of offsets
Memory gaps due to internal pointers or references in function blocks and structure components that are not released for symbol configuration. Memory gaps that occur differently in 32 -bit and 64 -bit systems depending on the data type, such as $\qquad$ XINT/ $\qquad$ XWORD.
Fields that are at uneven addresses. Some clients are not set up for this. Unintentional memory misalignment, which occurs when using the attributes 'pack_mode' or 'relative_offset'.
- "Use Empty Namespaces by Default (V2 Compatibility)": Required when using a CODESYS V2-compatible OPC server configuration.
V: Behavior same as in CODESYS V2.3.
- Program variables are exported without an application name (Application. PLC_PRG.MyVar --> PLC_PRG.MyVar
- Global variables are exported additionally without the GVL name (Application.GVL.MyGlobVar --> .MyGlobVar

|  | "Enable Direct I/O Access": This feature is potentially dangerous and not intended for operation in production. Activate only for error checking and tests, or when commissioning the machinery (for example, for checking cables connections). : In the symbol configuration, you can also use access to direct I/O addresses that correspond to IEC syntax (for example, "\%IX0.0"). Access to input addresses (I) is read-only*. Access to output addresses (Q) and memory addresses (M) can be read-write. <br> *Information: In simulation mode, write access to symbols is also possible for input addresses. <br> Because external clients for protocols such as OPC or OPC UA do not always support IEC syntax for direct addresses, access is also provided using an array syntax in the namespace $\qquad$ MIO of the implicit code. For example, you can also access $\qquad$ MIO.MIO _IX[2].x3 instead of \%IX2.3. However, the symbols for array access are hidden in browsers because some clients cannot handle the large number of nodes (several thousand depending on the size of the I/O ranges). <br> - "Support Calls of Functions, FBs, Methods, and Programs": <br> Note: Availability and editability of this option depend on the device. : The access rights "execute" can be set in the symbol table for symbols of POUs of type function, function block, method, or program. The "Support OPC UA features" option also has to be selected in the "Settings". <br> - "Include Call information in XML": : The information about called functions, function blocks, methods, or programs is also listed in the XML file of the symbol configuration. The option is enabled only if the "Support calls of functions, FBs, methods, and programs" option is supported by the device. <br> - "Enable Symbol Sets": : A toolbar with buttons and a list box is displayed above the symbol table. You can use this to configure symbol sets for client-specific assignment of access rights to the controller. See "Toolbar for symbol set configuration" below. |
| :---: | :---: |
| "Download" | If you use a device that supports its own application file for the symbol configuration, then this button is also available in the toolbar. If you change the symbol configuration in online mode, then you can load the new <application name>. symbols file immediately to the PLC. |
| "Tools" | "Save XSD Scheme File": This command opens the standard dialog for saving a file in the file system. With this command, you can prepare the XSD format of the symbol file, for example for use in external programs. |

Table 452: Symbol table

| "Access Rights" | You can change the access rights for a symbol by clicking the symbol in the "Access Rights" column. <br> Icons for access rights (in ascending order) <br> - Read only <br> - Write only <br> - Read and write <br> - 3: Execute <br> This permission allow for execute access to functions, function blocks, methods, and programs. <br> Requirements for the assignment: The device provides the "Support calls of functions, FBs, methods, and programs" and "Support OPC UA features" options. Both options are activated in the "Settings". <br> Note: In case the controller has a user management, you can use symbol sets to define client-specific access rights to the same symbols. |
| :---: | :---: |
| "Maximal" | Maximum access rights for this symbol |


| "Attribute" | If the access right was assigned by attribute, then a corresponding icon is dis- <br> played here. |
| :--- | :--- |
| "Type" | Alias data types are also displayed In CODESYS V3.5 SP6 and higher. <br> Example: MY_INT : INT for a variable declared with the data type MY_INT <br> (type INT). |
| "Members" | You can add variables of a structured data type also by selecting a check box <br> for symbol configuration in the "Symbols" column. This causes CODESYS to <br> export all member variable symbols. However, in the "Members" column, you <br> can click the ellipsis button (...) to select only specific structural components. <br> Note: This selection applies to all instances of this data type for which symbols <br> are exported. If a member of a structured type cannot be selected, then an <br> asterisk (*) is displayed in the check boxes of the members to indicate that all <br> exportable members of that type are exported. |

Table 453: Toolbar for symbol set configuration

| "List box" | Already defined symbol sets |
| :---: | :---: |
| $\pm$ "Add New Symbol Set" | Opens the "Add New Symbol Set" dialog for specifying a name for this set |
| ㅁ. "Add Duplicate from Selected Symbol Set" | Opens the "Add Duplicate from Selected Symbol Set" dialog. A copy is created for the set selected in the list box. You can change the default name (<group name>_duplicate). |
| "Rename Selected Symbol Set" | Opens the "Rename Selected Symbol Set" dialog for specifying another name for the set selected in list box. |
| "Delete selected Symbol Set" | Opens a dialog prompting whether or not the symbol set selected in the list box should be deleted. |
| "Configure Symbol Rights" | Opens the "Symbol Rights" tab of the device editor. When logged in there, you can assign different access rights for each user group (client) to the symbol set selected in the list box. |

See also

- ${ }^{\text {® }}$ Chapter 6.4.1.21.2.8.16 "Tab 'Symbol Rights"" on page 2458


## Dialog 'Com- <br> ments and <br> Attributes'

Table 454: "Symbol Table Contents"

| "Enable extended OPC UA <br> information" | Note: Availability and editability of this option depend on the device. <br> V: Additional information that can be evaluated by OPC UA servers is included <br> in the symbol table. This includes inheritance information of user-defined data <br> types and the namespace node flags. Additional information, such as comments <br> and attributes, can also be included if the OPC UA setting is active. <br> When the OPC UA setting is enabled, attributes are included in the symbol table <br> according to the following rule: |
| :--- | :--- |
|  | - In compiler versions V3.5.5.0 to V3.5.7.X, all attributes are included <br> according to the "Match simple identifiers" setting. <br> In compiler version V3.5.8.X, all attributes are included according to the <br> setting "Include all attributes". <br> In compiler version V3.5.9.0 and higher, you can customize the attributes <br> that are included. |
| "Include comments" | Requirement: "Enable extended OPC UA information" is activated. <br> V: Comments and attributes are also saved in the symbol table. |


| "Include attributes" |  |
| :--- | :--- |
| "Also include comments and | Requirement: "Include comments" is activated. |
| attributes for type nodes" | : The information for type nodes is also included (user-defined types, such as <br> STRUCT and ENUM elements). |
|  | $\square:$ Only directly exported variables have comments and attributes. |

Table 455: "XML symbol file contents"

| "Include namespace node flags" | The namespace node flags provide additional information about the origin of a node in the namespace. The node flags always in the symbol table when OPC UA is activated. However, its inclusion in the XML file can be deactivated as some defective parsers have problems with it. |
| :---: | :---: |
| "Include comments" | Comments can also be saved in the XML file. <br> In compiler versions V3.5.5.x to V3.5.8.0, this includes the setting "Prefer docucomments". |
| "Include attributes" | $\checkmark$ : Attributes can also be saved in the symbol file. |
| "Also include comments and attributes for type nodes" | Requirement: "Include comments" is activated. The information for type nodes is also included (user-defined types, such as STRUCT and ENUM elements). : Only directly exported variables have comments and attributes. |

Table 456: "Select Comments"
Requirement: "Include comments" is activated.
"Include docu comments"
"Include normal comments"
"Always include both types of
comments"
"Prefer docu comments, fallback to normal ones"
"Prefer normal comments, fallback to docu comments"

The options determines the comments that are saved in the symbol configuration.

Table 457: "Filter Attributes (Case-Insensitive)"

| Requirement: "Include attributes" is activated. |  |
| :--- | :--- |
| "Include all attributes" | Defines the attributes that are saved in the symbol configuration. |
| "Include attributes starting with" |  |
| "Filter attributes with regular |  |
| expression" |  |$\quad$| "Match simple identifiers" |
| :--- | | Exists primarily due to the backward compatibility to older versions in order to |
| :--- |
| emulate the old behavior. |

Setting: Configure synchronization with IEC tasks

For synchronously consistent access, the symbolic client waits in the runtime when processing a read or write request until a time is found when no IEC task is executed. When this gap is detected, restarting the IEC tasks is prevented until all values of the variable list have been copied. Then the IEC tasks are planned again as usual. Synchronized access can cause a delayed starting of IEC tasks, which is shown as increased jitter. As all applications in the runtime are managed by a common scheduler, this potential impairment of the real-time behavior
affects all applications on the device. All applications of the device are affected, regardless of whether or not they include a symbol configuration or they have been downloaded to the controller from one or more CODESYS projects. Therefore, the runtime permits synchronized consist access only if this it allows all applications that are downloaded to the controller at the time of access.

The setting is located in the editor of the symbol configuration of the "Settings" menu. In addition, the setting is also located in the context menu of the controller when you click the "Properties" command and then select the "Options" tab in the opened dialog.
For applications without symbol configuration, the setting can only be found in the properties dialog.

## NOTICE!

After changing the setting, all applications downloaded to the device by means of a download or online change have to be reloaded and all boot applications updated.

## In which cases is synchronized consistent access necessary?

As a rule, there is no need for consistent values for displayed values because it is mostly irrelevant from which IEC task cycle the changed values originate. It is completely irrelevant for seldom changed values. Even when writing there are almost no hard consistency demands because typically the machine must be in a kind of standby mode (for example when writing recipes) in which there is no direct access to the values written as recipes.
In contrast, consistent values are particularly necessary for database links to save production data. For clocked machines, however, these values must be synchronous with the production timing (one value set per produced product) and not consistent with reference to one or more IEC tasks. With reference to the machine clocking, the consistency must be already ensured by the IEC application. For this purpose, the values that arise during a production cycle are typically collected in a global variable list. At the end of the cycle, the symbolic client is notified by means of an additional variable (BOOL or counter) that the machine cycle has ended and the values are valid. Now the client has the chance to archive the values from the production cycle. Depending on necessity, the successful reading can also be displayed in the opposite direction by means of a released variable, so that the production can also be halted in case the production data cannot be archived. Synchronized consistent access is not necessary and helpful for this use case because the synchronization takes place at the application level.
In contrast, synchronized consistent access by symbolic clients is typically applied in the process industry with continuously running systems without production clocking when, for example when process values are written consistently and cyclically in a fixed time frame of 60s. This can take place either by synchronization on the application level similar to clocked machines (see above) or by synchronization of the synchronized consistent symbolic access. The advantage of the latter is that no logic has to be implemented in the IEC program and access is controlled entirely by the client.


## CAUTION!

Due to the increased jitter, the synchronized consistent monitoring is not suitable for motion or real-time critical applications. For these reasons, synchronized consistent access should be released and used only if it is absolutely necessary.

If a client uses synchronous consistent access released by this setting, then it has and effect on the client. Depending on the scheduler of the runtime, the response time can jitter more here for read/write access because the system might still have to wait for an execution gap of the IEC tasks. Read and/or write access can still fail when IEC tasks run for a long time (in the range of several 100 ms ) or the CPU load is close to $100 \%$ for an extended period of time with one or more IEC tasks (in the range of several 100 ms ). Therefore, the availability of the values also depends on the load of the controller by the IEC application.

Moreover, the client can minimize the effects on itself and on the runtime if it observes the following in the definition of the variable lists to be read or written:

- Synchronized consistent access only to those variables that are absolutely and consistently required.
- Separate variable lists for variables that have to be consistent and for variables that could be inconsistent.
- Divide variable lists with several consistent variables into several smaller lists.
- Select read intervals for cyclic reading of values as large as possible.

> Support for the current configuration and possible corrective actions

Entries marked in red in the symbol table show variables that they are configured for export to the symbol file but are currently invalid in the application. The cause for this can be that the declaration has been removed from the block.

In version 3.5.8.0 and higher, a warning appears in the editor if variables that have configured symbols are not used in the IEC code or are not mapped in the case of I/O variables. In addition, the compiler indicates variables that are referenced from outdated library versions $n$ the symbol configuration.

## NOTICE!

Object variables that are not used in the program code remain uncompiled by default and are therefore not available in the symbol configuration.

However, CODESYS provides variables from uncompiled objects in the symbol configuration when one of the following conditions is met:

- The "Link always" POU property is selected.
- The \{attribute 'linkalways'\} pragma is used.

See also

- $y^{2}$ Chapter 6.4.1.10 "Working with Controller Networks" on page 1936
- \& Chapter 6.4.1.21.4.11.20 "Dialog 'Properties' - 'Options'" on page 2765
- ⓨ Chapter 6.4.1.21.4.11.5 "Dialog 'Properties' - 'Build'" on page 2755
- 出 Chapter 6.4.1.20.6.3.25 "Attribute 'linkalways'" on page 2294
- ${ }^{4}$ Chapter 6.4.1.20.6.3.46 "Effects of Pragmas on Symbols" on page 2315


## Example for the data layout types

| Examples for the layout types | The following examples from an IEC application will show how gaps can result in the clientside memory layout caused by unpublished symbols, internal "invisible" pointers, or a "pack mode" definition in the device description. With the "Optimized layout" setting, the gaps are avoided. The symbol file contains different information about the size and offset of memory locations, depending on the selected layout setting. |
| :---: | :---: |
| Example: <br> Large structure | // Example of a big structure, where not all members get published STRUCT <br> \{attribute 'symbol':='readwrite'\} |
|  | PublicNumber : INT; |
|  | \{attribute 'symbol':='none'\} <br> InternalData : ARRAY[0..100] OF BYTE; |
|  | ```{attribute 'symbol':='readwrite'} SecondNumber : INT;``` |
|  | \{attribute 'symbol':='none'\} |
|  | MoreData : ARRAY[0..100] OF BYTE; |
|  | END_STRUCT |
|  | END_TYPE |
|  | Resulting entries in the symbol file; pay attention to "size" and "byteoffset": |
| Symbol file, large structure, compatibility layout option | <TypeUserDef name="T_LargeStructure" size="208" nativesize="208" typeclass="Userdef" pouclass="STRUCTURE" iecname="LargeStructure"> |
|  | <UserDefElement iecname="PublicNumber" type="T_INT" byteoffset="0" vartype="VAR" /> |
|  | <UserDefElement iecname="SecondNumber" type="T_INT" byteoffset="104" vartype="VAR" /> |
|  | </TypeUserDef> |
| Symbol file, large structure, optimized layout option | <TypeUserDef name="T_LargeStructure" size="4" nativesize="208" <br> typeclass="Userdef" pouclass="STRUCTURE" iecname="LargeStructure"> |
|  | ```<UserDefElement iecname="PublicNumber" type="T_INT" byteoffset="0" vartype="VAR" />``` |
|  | ```<UserDefElement iecname="SecondNumber" type="T_INT" byteoffset="2" vartype="VAR" />``` |
|  | </TypeUserDef> |
| Example: <br> Structure with uneven addresses |  |
|  | // The following mechanisms can cause memory misalignment: <br> // - \{attribute 'relative offset':='...'\} at a member |
|  | // - \{attribute 'pack_mode':='...'\} at a structure declaration |
|  | // - target setting 'memory-layout\pack-mode' in the device description |
|  | \{attribute 'pack_mode':='1'\} |
|  | TYPE UnevenAddresses: |
|  | STRUCT |
|  | \{attribute 'relative offset':='3'\} <br> \{attribute 'symbol':='readwrite'\} |
|  | PublicNumber : INT; |



```
<UserDefElement iecname="out" type="T_INT" byteoffset="2"
vartype="VAR_OUTPUT" />
</TypeUserDef>
Symbol file, <TypeUserDef name="T_POU" size="12" nativesize="12"
function block,
compatibility
layout option,
32-bit
typeclass="Userdef" pouclass="FUNCTION_BLOCK" iecname="POU">
<UserDefElement iecname="in" type="T_INT" byteoffset="8"
vartype="VAR_INPUT" />
    <UserDefElement iecname="out" type="T_INT" byteoffset="10"
    vartype="VAR_OUTPUT" />
    </TypeUserDef>
Symbol file, <TypeUserDef name="T_POU" size="4" nativesize="12"
function block,
optimized
layout option,
32-bit
typeclass="Userdef" pouclass="FUNCTION_BLOCK" iecname="POU">
<UserDefElement iecname="in" type="T_INT" byteoffset="0"
vartype="VAR_INPUT" />
    <UserDefElement iecname="out" type="T_INT" byteoffset="2"
    vartype="VAR_OUTPUT" />
    </TypeUserDef>
```

See also

-     * Chapter 6.4.1.10.3 "Symbol Configuration" on page 1941


## Object 'Text List'

Symbol:
This object is used to create, manage, and translate texts. It contains a table with texts where you can add new texts. You can select a text which you have composed here can be selected in a visualization in the "Dynamic texts" property of an element. In runtime mode, the visualization displays this text dynamically in the selected language.
When the object is assigned to an alarm group and is located below the "Alarm Configuration" object, CODESYS adds the texts of the alarm group to the table. You can also add texts.

| "ID" | Unique identifier of the text |
| :--- | :--- |
| "Standard": | Source text as character string (example: Information A: \% i options). <br> Use the keyboard shortcut [Ctrl]+[Enter] to add a line break. <br> Double-click in the field to edit the text. |
| The table contains as many language columns as you want to add. A language column is named with a language <br> code that you specified when you created the column by means of the "Insert Language" command. |  |
| "<language code>" | Name of the language as a language code. Example: en-US. This column con- <br> tains the translation of the text which is composed under "Standard". Under the <br> condition that the language code is selected as the language in the visualization <br> manager, a visualization displays the translated text in runtime mode. If a trans- <br> lation has not been composed, then CODESYS uses the text under "Standard" <br> A visualization in runtime mode can also change the language if requested by a <br> user. |
| Blank line | Edit the line to add your own text. |

See also

- $\quad>$ Chapter 6.4.1.21.2.11 "Object 'GlobalTextList'" on page 2465
- For descriptions about alarm management and alarm visualization, see the help for CODESYS Visualization.


## Object 'Task Configuration'

## General

Symbol:
The object is used to define and display the basic settings for the task configuration.
The "Task Configuration" object must be included exactly one time in each application.
"Task Configuration" tabs and functions

- "Properties": Display of the basic settings
- "System Events": Linking of POU calls with system events
- "Monitor": Display of the status and current statistics for the cycles times in online mode
- "Variable Usage": Overview of the tasks that access the variables and how they do it
- "Task Groups": Definitions of the tasks groups and their assignment to CPUs
- "CPU Load": Graphical representation of the CPU load in online mode

See also

- ${ }^{4}$ Chapter 6.4.1.9.17.2 "Creating a task configuration" on page 1914
- ${ }^{4}$ Chapter 6.4.1.21.2.29.2 "Tab 'Properties"' on page 2533
- ${ }^{\wedge}$ Chapter 6.4.1.21.2.29.3 "Tab 'System Events"' on page 2533
- $\Rightarrow$ Chapter 6.4.1.21.2.29.4 "Tab 'Monitor"' on page 2536
- ${ }^{4}$ Chapter 6.4.1.21.2.29.5 "Tab 'Variable Usage"" on page 2537
- ${ }^{4}$ Chapter 6.4.1.21.2.29.6 "Tab 'Task Groups"' on page 2537
- « ${ }^{\circledR}$ Chapter 6.4.1.21.2.29.7 "Tab 'CPU Load'" on page 2537
- $\boldsymbol{y y}^{\text {• }}$ Chapter 6.4.1.21.2.30 "Object 'Task"' on page 2538


## Tab 'Properties'

## Object: "Task Configuration"

In this tab, you define the basic settings of the task configuration as predefined by the target system, such as the maximum values for tasks and watchdog parameters.

## Tab 'System Events'

Object: "Task Configuration"
On the "System Events" tab, you define which event calls which function and whether or not the configuration is currently activated. You use this tab when a system event (instead of a task) should call a project function.

| "Add Event Handler" | Opens the "Add Event Handler" dialog |
| :--- | :--- |
| "Remove Event Handler" | Deletes the selected list assignment |
| "Event Info" | Shows information from the corresponding event library |


| "Open Event Function" | Opens the editor of the new function for the selected assignment You have <br> selected the implementation language of the new function in the "Add Event <br> Handler" dialog. |
| :--- | :--- |
| Ans |  |

Assignment of functions to call for events with: "Name", "Description", "Function to call", and "Active" (activate/deactivate configuration).

Table 458: "Add Event Handler"

| Adds a new assignment "Event - Function to call" to the list |  |
| :--- | :--- |
| "Event" | The possible selection depends on the target device. CODESYS marks unavail- <br> able events with a red symbol in front of the name. <br> A list of all possible system events is located at the end of this section. |
| "Function to call" | Function name ("POU", type "FUNCTION") <br> You have to specify the name of the new function. CODESYS inserts the func- <br> tion to the device tree after you confirm the dialog. |
| "Scope" | - "Application": The function is available to the application. <br> - "POUs": The function is available to the entire project. |
| "Implementation language" | Implementation language for the new function |
| "Description" | Short description of the selected event |

Features in The list of assignments from called functions to events also includes the following information:
"Online Mode" "Event Status", "Call Count", and the "Online Reset" button.

| "Event Status" | 0 : No error has occurred. |
| :--- | :--- |
| Does not equal 0: Error. You need to consult the respective runtime system |  |
| documentation. |  |


| Possible system <br> events | Event | Description | Task |
| :--- | :--- | :--- | :--- |
|  | PrepareStart | Call before starting the <br> application | Communication task |
|  | StartDone | Call after starting the <br> application | Communication task |
| PrepareStop | Call before stopping <br> the application | Communication task | No |
| StopDone | Call after stopping the <br> application | Communication task | No |
| PrepareReset | Call before resetting <br> the application | Communication task | No |
| ResetDone | Call after resetting the <br> application | Communication task | No |
| PrepareOnline- <br> Change | Call before online <br> change of the applica- <br> tion | Communication task | No |


| Event | Description | Task | Debugging |
| :---: | :---: | :---: | :---: |
| OnlineChangeDone | Call after online change of the application | Communication task | No |
| PrepareDownload | Call before downloading the application | Communication task | No |
| DownloadDone | Call after downloading the application | Communication task | No |
| PrepareDelete | Call before deleting the application | Communication task | No |
| DeleteDone | Call after deleting the application | Communication task | No |
| PrepareExit | Call before exiting the application | Communication task | No |
| ExitDone | Call after exiting the application | Communication task | No |
| CodelnitDone | Event is sent after Code Init. Called within the task safe section and only for an online change change (for example, the copy code for online change is executed here). | Communication task | No |
| Exception | The event is sent if an exception has occurred in the context of an application. | Exception handling task of the runtime, or the task itself if the runtime does not support exception handling | Depends on the task |
| Login | Login of a client to this application | Communication task | No |
| Logout | Logout of a client from this application | Communication task | No |
| BeforeReadinglnputs | Call before reading the inputs | IEC task | Yes |
| AfterReadingInputs | Call after reading the inputs | IEC task | Yes |
| BeforeWritingOutputs | Call before writing the outputs | IEC task | Yes |
| AfterWritingOutputs | Call after writing the outputs | IEC task | Yes |
| DebugLoop | Event is sent in cycles to the debug loop if the IEC task stops at a breakpoint. | Communication task | No |
| PrepareShutdown | Event is sent immediately before the runtime system is downloaded. | Runtime main loop | No |


| Event | Description | Task | Debugging |
| :--- | :--- | :--- | :--- |
| PrepareExitComm | Event is sent during <br> download before <br> exiting the communica- <br> tion server. | Runtime main loop | No |
| PrepareExitTasks | Event is sent during <br> download before <br> exiting all tasks. | Runtime main loop | No |

## Tab 'Monitor'

## Object: "Task Configuration"

In online mode, the tab shows the status of the tasks of the task configuration, as well as some current measurements of the cycles and cycle times. CODESYS updates the values in the same time interval as for the monitoring of values from the PLC.
The displayed values can be reset to 0 by means of the "Reset" context menu command.

| "Task" | Task name (as defined in the task configuration) |
| :--- | :--- |
| "Status" | - "Not created": The task has not been started since the last update (especially for <br> event tasks). <br> "Generated": The task is recognized in the runtime system, but not yet in opera- <br> tion. <br> "Valid": The task is operating normally. <br> - "Exception": The task has produced an exception status. |
| "IEC-Cycle Count" | Number of cycles executed since starting the application where the IEC code was <br> executed (0 if the target system does not support the counter function) |
| "Cycle Count" | Number of executed cycles since logging in to the PLC <br> It depends on the target system whether cycles are also counted where the appli- <br> cation is not running. In these cases, the "Cycle Count" may be greater than the <br> "IEC-Cycle Count". |
| "Last Cycle Time $(\mu s) "$ | Last measured cycle time [ $\mu \mathrm{s}$ ] |

See also

- $\mu_{\text {M }}$ Chapter 6.4.1.9.17.3 "Definitions of Jitter and Latency" on page 1915
- ② Chapter 6.4.1.9.17 "Task Configuration" on page 1914


## Tab 'Variable Usage'

Object: "Task Configuration"
The "Variable Usage" tab provides an overview of all variables and their usage. There you can see the tasks where variables are accessed.
When using multicore, write access (w) to a variable should take place only in a task because otherwise it can cause inconsistencies.

In the context menu, you can hide individual tasks and show the cross-reference list to variables.

| "Variables" | Name of the variable |
| :--- | :--- |
| "Type" | Data type |
| "Number" | Number of tasks that access these variables. |
| "<task name>" | Access to the variable (r: read, w: write, rw: read/write) |

See also

- $\quad$ ② Chapter 6.4.1.9.17 "Task Configuration" on page 1914


## Tab 'Task Groups'

## Object: "Task Configuration"

You define task groups on the "Task Groups" tab. Task groups can be distributed over the individual processor cores in multicore systems. The tasks of a task group are bound to the processor cores according to the strategy defined in the "Core" field.

| "Add Group" | The button adds a new task group named NewGroup_<no>. |
| :--- | :--- |
| "Remove Group" | Deletes the selected task group. |
| "Group Name" | The name can be changed by double-clicking in the field. |
| "Core" | Determines the processor core for process the tasks of this group. <br> - "Free floating": All tasks are bound dynamically to different processor cores. <br> The user does not have any influence over this. The operating system is <br> responsible for the distribution. |
|  | "Sequentially pinned": All tasks are bound and fixed to different processor <br> cores. The user does not have any influence over this. <br> "Fixed pinned": All tasks are bound to one processor core. By default, the <br> runtime system determines the processor core. |
| "<core number>": Fixed defined processor core. If the processor core is not |  |
| available, then an error message is issued. |  |

## Tab 'CPU Load'

## Object: "Task Configuration"

The "CPU Load" tab is available in online mode for multicore devices only. The load of the individual CPUs is presented in the trace editor.
You open the trace configuration by double-clicking the legend in the window on the right side. Adding more variables is not possible here.

See also

- $乡$ "Displaying the CPU load with DeviceTrace objects in the CODESYS project (example)" on page 2015


## Mapping of Task Priorities on a Linux System

Dependencies When the target system is running in Linux, specific priorities of the IEC tasks are mapped between IEC task priorities to specific priorities and scheduling policies. Depending on the priority, the scheduling policy SCHED_FIFO or SCHED_OTHER is displayed as in the table. and Linux priorities

| IEC Task Priority | Linux Priority Linux (Default) |
| :--- | :--- |
| 0 (highest real-time priority) | 56 (SCHED_FIFO) |
| 15 (lowest real-time priority) | 41 (SCHED_OTHER) |
| 16 (no real-time priority) | 0 (SCHED_OTHER) |
| 31 (no real-time priority) | 0 (SCHED_OTHER) |

## Object 'Task'

## General

Symbol:
In this object, you define the conditions for starting and calling the task.
You insert the object below "Task Configuration" in the device tree.

## Tab 'Configuration'

Object: "Task"

| "Priority" | Possible values: $0 . .31$, where 0 is the highest priority |
| :--- | :--- |
| "Task group" | Assigned task groups which can be assigned to specific processor cores in <br> multicore <br> Example: <br>  <br>  <br> IEC Tasks Note: The task group is shown in brackets after the task in the device <br> tree. |

When using a runtime system in Linux, there exist fixed dependencies between task groups and priorities. For more information, see: $\Rightarrow$ Chapter 6.4.1.21.2.29.8 "Mapping of Task Priorities on a Linux System" on page 2538

Table 459: "Type"

| Type "Cyclic" | Processing of the task is done cyclically. |
| :--- | :--- |
| Input field Interval | Required <br> Time span after which the task is restarted (task cycle time) <br> - As a time definition in the format TIME <br> Example: t\#200ms |
|  | As a number <br> Example: 200 Hint: the number is displayed automatically in the format TIME <br> when the input field is in focus again. <br> Note: Deviations of the task from this desired task cycle time are displayed at <br> runtime as periodic jitter on the Watchdog tab. |
| Time unit of the interval | If only a number and not a time definition is specified in the Interval input field, <br> then the unit selected here determines the time dimensions. <br> Example: ms <br> Note: A task cycle time in $\mu$ s is always displayed as a number. |
| Type "Event" | Processing of the task starts event-controlled on the rising edge of the event <br> variable. |
| Input field "Event" | Global variable (Boolean type) <br> The task starts as soon as the variable value switches from 0 to 1. |
| Type "External" | Processing of the task starts event-controlled on the rising edge of the event <br> variable. |
| List box "Event" | List with target system-dependent events (Boolean type) <br> Note: The target system determines which events are supported and offered in <br> the list box. <br> Hint: Not to be confused with system events |
| "Interval" | Time definition in TIME format or as a number with a time unit Note: Only <br> available when the event requires a time definition |
| Input field "Event" | Processing of the task starts automatically in a continuous loop at program start <br> and at the end of a complete pass <br> Note: A cycle time is not defined. |
| Processing of the task starts state-triggered by the event variables |  |
| Global variable (Boolean type) When the variable has the state TRUE hat, the |  |
| task starts. |  |

## NOTICE!

For fieldbuses, a fixed cycle matrix is necessary to assure a determined behavior. Therefore, you should not use TypeFreewheeling for a bus cycle task.

## NOTICE!

Note the following difference between the processing types Status and Event. If the given event yields TRUE, then the start condition of a task of type Status is fulfilled. In contrast, the start of a task of type Event requires a switch of the event from FALSE to TRUE. If the sampling rate of the task scheduler is too low, then the rising edge of the event can remain unnoticed.

## NOTICE!

When setting the task cycle time, you have to identify which bus system is currently being used. For example, the task cycle time in a CAN bus system must match the currently set baud rate and the number of frames used in the bus. In addition, the times set for heartbeat, node guarding, and sync should always be a multiple of the task cycle time. If not, then CAN frames can be lost.

Table 460: "Watchdog"
Defines the time monitoring for a task. If the target system supports an advanced watchdog configuration, then the following settings may be predefined in the device description.

- Upper and lower limit
- Default watchdog time
- Time specified as percentage

The default watchdog settings depend on the device.

| "Enable" | The watchdog is active. <br> If the task exceeds the currently set "Time" of the watchdog, then the task is <br> halted with an error status (exception). The application in whose task the error <br> occurred and its child applications are also halted. In this way, all tasks of the <br> affected applications are also halted. Then the currently defined "Sensititity" is <br> also taken into account. If you activate the option "Update I/Os" in the "PLC <br> Settings" of the PLC, then CODESYS resets the outputs to the defined default <br> values. <br> Possible cases: <br> - Multiple consecutive timeouts: <br> Sensitivity: 0, 1 - exception in cycle 1 <br> Sensitivity: 2 - exception in cycle 2 <br> Sensitivity: n - exception in cycle n <br> Single timeout: Exception if the cycle time of the current cycle is longer than <br> (time *sensitivity). Example: Time=t\#10ms, Sensitivity=5 (i.e., exception as <br> soon as the one-time task runs longer than 50 ms) |
| :--- | :--- |
| "Time (e.g. t\#200ms)" | Watchdog time <br> Defines (together with "Sensitivity") the watchdog for a task; description as for |
| "Enable". |  |
| Depending on the target system, the monitoring time span is given as a per- |  |
| centage of the task interval if possible. In this case, the list box for the unit is |  |
| disabled and displays "\%". |  |

Using the functions from the library CmpIecTask. Iibrary, you can deactivate a watchdog for specific PLC cycles. This is useful for cycles that demand more time due to initialization.

```
Example Deactivating/reactivating the watchdog:
    hIecTask := RTS_IEC_HANDLE //Declaration of the variable hIecTask
    hIecTask := IecTask\overline{GetCurrent(0);}
    IecTaskDisableWatchdog(hIecTask); // Watchdog disabled
    IecTaskEnableWatchdog(hIecTask); Watchdog enabled
```

List of "POU"s that the task calls
The calling order corresponds to the POU order in the list (from top to bottom).

| "Add Call" | Defines a new program call |
| :--- | :--- |
| "Open POU" | Opens the selected POU |
| "Move Up" | Changes the calling order |
| "Move Down" |  |

The normal watchdog of an IEC task is triggered when the execution time of the IEC task exceeds the watchdog time. The "Omitted Cycle" watchdog is triggered when the task does not start at all. This is the case when the task does not execute any cycle at all within the maximum of <Time * Sensitivity> or $<2$ * Interval>. The cause could be crowding by other tasks or a failure in the scheduler, which no longer enables the task.

## See also

- ${ }^{\Perp}$ Chapter 6.4.1.21.2.29 "Object 'Task Configuration"' on page 2533
- ${ }^{\text {• }}$ - Chapter 6.4.1.21.2.29.4 "Tab 'Monitor'" on page 2536
- $\&$ Chapter 6.4.1.21.2.29.6 "Tab 'Task Groups'" on page 2537


## Object 'Trace'

## Symbol: a

An object of type "Trace" is used for configuring and displaying application-specific trace data in one or more charts. At application runtime, value curves of trace variables, which you can monitor in the trace editor in CODESYS, are recorded on the controller. Requirements are that a trace configuration has been configured transferred to the controller, and the trace recording has been started. The recorded data is transferred to the development system and displayed in diagrams according to the configuration. You can navigate through the data when tracing.


If the controller supports a Trace Manager, then you can use the 'DeviceTrace' object type in the Trace Manager to access all traces that are running on the controller.

Double clicking the trace object opens the trace editor. The corresponding toolbar contains the most important trace commands. The trace variable list shows the variable whose value curve is recorded.

－（1）：Toolbar of the trace editor
－（2）：Trace editor
－（3）：Trace variable list
－（4）：Links for trace configuration
＂Configuration＂
＂Add Variable＂

## See also

－${ }^{4}$ Chapter 6．4．1．13．3＂Data Recording with Trace＂on page 2007
－${ }^{4}$ Chapter 6．4．1．21．2．7＂Object＇DeviceTrace＂＇on page 2426

Toolbar of the trace editor

See also


- D ：${ }^{凶}$ Chapter 6．4．1．21．3．22．16＂Command＇Start Trace＂＇on page 2741
- II：觉 Chapter 6．4．1．21．3．22．17＂Command＇Stop Trace＇＂on page 2741
- 赤：＂${ }^{4}$ Chapter 6．4．1．21．3．22．13＂Command＇Reset Trigger＂＇on page 2740
- ta：鿊 Chapter 6．4．1．21．3．22．9＂Command＇Mouse Zooming＇＂on page 2737
－ $\mathbf{Q}^{2}$ ：${ }^{2}$ Chapter 6．4．1．21．3．22．2＂Command＇AutoFit＂＇on page 2733
－Iy：${ }^{\circ}$ Chapter 6．4．1．21．3．22．5＂Command＇Cursor＇＂on page 2733
－$: ~ \sum_{2}$ Chapter 6．4．1．21．3．22．3＂Command＇Compress＇＂on page 2733
－炛： $\boldsymbol{\xi}^{\star}$ Chapter 6．4．1．21．3．22．3＂Command＇Compress＇＂on page 2733
－$\overleftrightarrow{4}$ ：${ }^{\circ}$ Chapter 6．4．1．21．3．22．18＂Command＇Stretch＇＂on page 2742


## Trace editor

At application runtime，the runtime system buffer of the trace component is filled with the recorded samples．The data is transferred to the development system and stored in its trace editor buffer．The trace editor accesses this data and displays it in diagrams as a graph over time．When you close the trace editor，the trace editor buffer will be freed．
Use menu commands for controlling the trace．In addition，you can use menu commands， keyboard shortcuts，and mouse input for navigating through the data．
See also
－$\Leftrightarrow$ Chapter 6．4．1．13．3．4＂Operating the data recording＂on page 2014
－$\Rightarrow$ Chapter 6．4．1．13．3．6＂Navigating into trace data＂on page 2015

Trace variable list

The trace variable list provides an overview of the current trace configuration. In the list, all charts with the respective trace variables are displayed in a table. When you double-click a trace variable, the "Trace Configuration" dialog also opens with the variable settings.

| $\nabla$ | A list box opens by means of the "Hide Instance Paths" command. |
| :--- | :--- |
| "Hide Instance Paths" | Display of the variable name in the list <br>  <br>  <br> - Variable name with full instance path <br> Example: PLC_PRG.iCounter |

Table 461: Charts

| Tabular display of the charts: |  |
| :--- | :--- |
| "Name" | List of charts with the respective variables <br> - "Chart <n>" <br> The chart name can be changed by clicking the selected name. <br> "<variable>" <br> The variable name variable is displayed. <br> editor. |
|  | When you select a "Chart <n>" in the table, the corresponding chart is also <br> selected in the editor. This also works the other way around. |
| "Cursor <n>" Y-value at the cursor position |  |

You can drag the charts and variables to sort them or move them to other diagrams. When the [Ctrl] key is pressed, the variable is copied. This is also possible in online mode.

Table 462: Context menu in the trace variable list

| "Add Variable" | Adds a new trace variable and opens the "Trace Configuration" dialog with its <br> variable settings. Select a variable in the input field of the "Variable" setting to <br> trace its value curve. |
| :--- | :--- |
| "Visible" | Toggles the visibility of the graph (value curve or trace variable) in the corre- <br> sponding diagrams: |
|  | $\bullet$ <br> $\bullet$ <br> When multiple charts or variables are selected in the trace variable list by multi- <br> selection, the visibility can be toggled for these graphs. |
| "Display Mode" | Opens the "Trace Configuration" dialog. Select a configuration item in the "Trace <br> Record" tree view or "Presentation (Diagrams)". |
| "Configuration" | Opens the "Trace Configuration" dialog. The "Variable Settings" are displayed on <br> the right. |

## See also

- ${ }^{4}$ Chapter 6.4.1.21.4.16.2 "Dialog 'Trace Configuration'" on page 2804

Navigating in the diagram
Table 463：With mouse input

| User input with the mouse | Mouse cursor symbol during user input | Effect |
| :---: | :---: | :---: |
| Drag the graph along the X －axis． | 4－＞ | Scrolls trace graphs of all diagrams at the same time along the time axis（X－axis）． |
| Hold down the［Ctrl］key and drag the graphs along the Y －axis． |  | Scrolls the trace graphs of the selected dia－ grams along the Y －axis． |
| Roll the mouse wheel backwards． |  | Compressed time axis（like the 信 symbol）． |
| Roll the mouse wheel forwards． |  | Stretches time axis（like the $\overleftrightarrow{m}$ symbol）． |
| Press and hold down the［Ctrl］key and roll the mouse wheel backwards． |  | Compresses the Y－axis． |
| Press and hold down the［Ctrl］key and roll the mouse wheel forwards． |  | Stretches the Y－axis． |
| Requirement：One or two trace cursors are activated． <br> Drag the triangle of a trace cursor to another position along the time axis． |  | Refreshes the Y －values in the trace variable list at the same time <br> －First value：$Y$－value at the position of the left cursor． <br> －Second value：Y－value at the position of the right cursor． <br> －Third value：Difference between both values． |
| Requirement：＂Mouse zooming＂is activated （田） <br> Stretch a rectangle． | ＋ | Zooms the trace graphs of all diagrams to the rectangle． |

Table 464：With keyboard shortcuts

| Shortcut | Effect |
| :---: | :---: |
| Requirement：No trace cursor is activated． <br> ［Arrow Left］ <br> ［Arrow Right］ | Scrolls trace graphs of all diagrams at the same time along the time axis． |
| ［Arrow Up］ <br> ［Arrow Down］ | Scrolls the trace graphs of the selected diagrams along the Y－axis． |
| Requirement：One or two trace cursors are acti－ vated． <br> ［AIt］＋［Arrow Left］ <br> ［Alt］＋［Arrow Right］ | Scrolls trace graphs of all diagrams at the same time along the time axis． |
| ［－］ | Compressed time axis（like the 焳 symbol）． |
| ［＋］ | Stretches the X－axis（like the $\overleftrightarrow{\sim}$ symbol）． |
| ［Ctrl］＋［－］ | Compresses the Y－axis of the selected diagram． |
| ［Ctrl］＋［＋］ | Stretches the Y－axis of the selected diagram． |
| ［Tab］ | Selects the next lower diagram． |


| Shortcut | Effect |
| :--- | :--- |
| Requirement: One or two trace cursors are acti- <br> vated. <br> [Arrow Left] | Moves the black trace cursor. |
| [Arrow Right] |  |
| Requirement: Two trace cursors are activated. <br> [Shift $]+[$ Arrow Left] | Moves the gray trace cursor. |
| [Shift] + [Arrow Right] $]$ |  |

See also

- \& Chapter 6.4.1.21.3.22.5 "Command 'Cursor"' on page 2733
- \& Chapter 6.4.1.21.3.22.9 "Command 'Mouse Zooming"' on page 2737
- $\Delta$ Chapter 6.4.1.13 "Application at Runtime" on page 1995


## Object 'Trend Recording Manager'

Symbol
A "Trend Recording Manager" object makes it possible to save data at runtime in a database for a long period of time. This data is recorded with the "CmpTraceMgr" runtime system component. In the device tree, this object is used as a node for trend recordings that are created below an application. It is available below an application only one time.
See also

- \& Chapter 6.4.1.13.4 "Data Recording with Trend" on page 2016
- $\#$ Chapter 6.4.1.21.3.5.1 "Command 'Add Object"' on page 2598
- $\Leftrightarrow$ Chapter 6.4.1.21.2.33 "Object 'Trend Recording'" on page 2545


## Object 'Trend Recording'

Symbol:
A "Trend Recording" object is always located below a "Trend Recording Manager" and enables editing of the trace configuration. At runtime, CODESYS transfers the configuration that is available to the runtime system component CmpTraceMgr. You can configure an application with any number of trend recordings.

## NOTICE!

## Timeout for trend recording

During a trend recording, it can happen that the application task triggers a timeout that is caught with an exception when transitioning from "Running" to "Stop". Causes can be that file operations with the SQLite database are taking too long or that too many variables are being recorded. This usually happens on a target device with weak performance.
You can avoid the occurrence of an exception:

- Configure the trend recording with less memory demand so that the amount of data that is stored is adapted to the target system.
- Reduce the number of variables.

The editor includes the configuration for trend recording. The tree view shows the trend configuration and enables navigation there.

The top entry contains the trend name. When this entry is selected, the "Record Settings" are displayed next to it. An entry is located here for each variable whose data was recorded continuously. When a variable is selected, the "Variable Settings" are displayed next to it.

| "Add Variable" | When you click the link, a new entry is displayed in the trend configuration with <br> its blank configuration below the "Variable Settings" group. |
| :--- | :--- |
| "Delete Variable" | The selected variable is removed. Requirement: A variable is selected. |

See also

- Command 'Edit Trend Recording'
'Recording Set- The data is recorded in the runtime system component by means of the functionality which is tings' also used for the trace. The settings that appear here are the same. The options that are not required here are deactivated.

| The settings that affect the trigger are deactivated. Only a trace configuration for a trace editor can configure triggering. |  |
| :---: | :---: |
| "Task" | Task where data was recorded. Click $\geqslant$ to open a list box with all tasks available in the project. In general, the trend recording runs in the same task as the main program. For example: MainTask |
| "Recording condition" | Condition under which the application records data: <br> - IEC variable of type INT. The condition is fulfilled for TRUE. <br> - Bit access to an integer variable. The condition is fulfilled for 1 . <br> As read access to a property. <br> The contents of a pointer are not permitted. <br> Note: If no condition is defined, then the recording starts automatically. |
| "Comment" | Comment (example: Data recording of sensor A) |
| "Resolution" | Resolution that the application saves the time stamp <br> Note: If the task where the trend object is executed has a cycle time of 1 ms or less, then you should set the resolution of the time stamp to " $1 \mu \mathrm{~s}$ ". |
| "Trend Storage" | The "Trend Storage" dialog opens. |
| "Advanced" | The "Advanced Trend Settings" dialog opens. |

- $\sum^{*}$ Chapter 6.4.1.21.4.17 "Dialog Box 'Trend storage"" on page 2809
- «y Chapter 6.4.1.21.4.18 "Dialog Box 'Advanced Trend Settings'" on page 2810
'Variable Settings'

| "Variable" | Variable for recorded value. <br> - IEC variable with valid data type <br> - Property <br> - Reference <br> - Contents of the pointer <br> - Array element of a valid data type <br> - Enumeration of a valid data type <br> Valid data types are all standard types, except STRING, WSTRING, and ARRAY. |
| :---: | :---: |
| "Parameter" | Parameter for the recorded value <br> The "Input Assistant" dialog lists all valid system parameters in the "Parameters" category of the "Categories" tab. |
| - | Click the symbol to toggle between "Variable" and "Parameter". |
| "Recording condition" | Condition under which the application records the data of these "Variables": <br> - IEC variable of type Int. The condition is fulfilled for TRUE. <br> - Bit access to an integer variable. The condition is fulfilled for 1 . <br> As read access to a property <br> The contents of a pointer are not permitted. <br> Note: If no condition is defined, then the recording starts automatically. |
| "Attached y axis" | Y -axis of the trend diagram that displays the "Variable". The list box provides the standard Y -axis and the configured Y -axes. <br> Requirement: This option is visible only when the "Trend" visualization element has configured additional Y-axes in the "Edit Display Settings" dialog. |
| "Display variable name" | The visualization shows the name of the IEC variable in the trend diagram at runtime. Either alone or in parentheses after the "Description" $\square$ The name of the IEC variable is shown and does not appear in parentheses after the "Description". <br> Requirement: If any text is typed in "Description", then you can disable the option. |
| "Description" | Text for the tooltip (example: Sensor A): When a visualization user focuses on the variable in the trend diagram, the visualization shows the text as a tooltip. The text is typed into the "GlobalTextList" object and can be localized there. <br> When the "Display Variable Name" property is activated, then the text is supplemented with the variable name in parentheses. Example: Sensor A (PLC_PRG.iSensor_A) <br> If "Description" does not contain any text, then "Display Variable Name" is enabled. The name is then alone without parentheses (for example, PLC_PRG.iSensor_A). <br> If a legend is assigned to the trend, then the trend variable is labeled in the legend and shown like the trend is configured here. |
| "Curve type" | - "Line" <br> - "Area": |
| "Graph color" | Color of the curve in the trend diagram |
| "Line type" | - "Line": Values are linked to form a line. <br> - "Step": Values are linked in the form of steps. <br> - "None": Values are not linked. <br> Requirement: The "Curve type" is "Line". |


| "Filling type" | - "No filling" <br> - "Plain color": <br> - "Gradient" <br> Requirement: The "Curve type" is "Area". |
| :---: | :---: |
| "Filling color" | The area is filled with the selected color. Requirement: The "Curve type" is "Area". |
| "Transparency" | Value (0 to 255) for defining the transparency of the selected color <br> Example 255: The color is opaque. 0 : The color is completely transparent <br> Requirement: The "Curve type" is "Area". |
| "Line width" | Value (in pixels) <br> Example: 1 |
| "Line style" | The display of the line is solid, dash, dot, dash-dot, or dash-dot-dot. |
| "Point type" | Display as scatter chart <br> - "Dot": Value as a dot. <br> - "Cross": Value as a cross. <br> - "None": No dot display <br> Hint: Select "None" for larger size data. |
| "Activate minimum warning" | $\square$ : Warning when below the lower limit. |
| "Critical lower limit" | If the variable value is below the limit, then the variables are displayed with the alert color in the trend diagram. |
| "Color" | Warning color on falling below the limit |
| "Activate maximum warning" | $\square$ : A warning is issued if the upper limit is exceeded. |
| "Critical upper limit" | If the variable value exceeds the limit, then the variables are displayed with the alert color in the trend diagram. |
| "Color" | Warning color on exceeding the limit |

- ${ }^{\text {© }}$ Chapter 6.4.1.21.4.16.2 "Dialog 'Trace Configuration"' on page 2804
- Dialog 'Display Settings'
- Visualization Element 'Legend'

See also

- ${ }^{2}$ Chapter 6.4.1.21.3.5.1 "Command 'Add Object"' on page 2598
- ${ }^{4}$ Chapter 6.4.1.13.4 "Data Recording with Trend" on page 2016
- Visualization Element 'Trend'


## Object 'Trend Recording Task'

Symbol
If you design a visualization with a "Trend" element, then CODESYS automatically extends the "Task Configuration" with a "Trend Recording Task". The task is below an application one time at most and calls the
VisuTrendStorageAccess.GlobalInstances.g_TrendRecordingManager.CyclicC all program to run the trend recording manager.
See also

- \& Chapter 6.4.1.21.3.5.1 "Command 'Add Object"' on page 2598
- ${ }^{\circ}$ Chapter 6.4.1.13.4 "Data Recording with Trend" on page 2016


## Object 'Unit Conversion'

Symbol 寇
A "Unit Conversion" object is used to define a conversion rule. The following table lists all defined conversion rules. You can edit a conversion rule in the input fields listed below the table.

Table of conver-
sion rules

| "Name" | <name> : <name> Impl is the name of the conversion rule. CODESYS automatically implements the entry as a function block <name> Impl and instances it as <name>. |
| :---: | :---: |
| "Type" | Type of conversion rule <br> - "Single scaling (offset)": adds an offset to the input variable. Result := Input + Offset <br> - "Single scaling (factor)": multiplies the input variable by a factor. Result := Input * Factor <br> - "Linear scaling 1 (factor and offset)": converts the input variable with a factor and offset. Result := Input * Factor + Offset <br> - "Linear scaling 2 (Base and target range)": converts the input variable for the output value to be within a target range. CODESYS calculates the functional linear equation internally. <br> - "User defined conversion": configures a user-defined conversion rule with IEC operators. The input variable is rValue. <br> - "Switchable conversion": defines a conversion rule that CODESYS executes independent of any specified language or variable. |
| "Setting" | Displays the configured conversion rule. |
| "Condition" | - "TRUE": CODESYS always executes the conversion. <br> - "Language" If the language in the visualization is the language defined here, then CODESYS executes the conversion. The current visualization language is located in the VisuElems. CurrentLanguage variable. <br> - "Variable": If the comparison is TRUE, then CODESYS executes the conversion rule. CODESYS can pass the comparison for a constant, variable, or IEC expression. <br> You can edit the comparison below the table in the "Condition Setting". |
| "Condition Setting" | If you select "TRUE" as the "Condition", then the field is hidden. <br> If you configure "Language" as the "Condition", then the field shows the current configuration, for example en, de. <br> If you select "Variable" as the "Condition", then the field shows the current configuration, for example PLC_PRG.bActual=PLC_PRG.bSet. <br> You can edit the current condition setting below the table in the input fields for "Condition Setting". |

Input field The input variable is added with an offset. 'Single scaling (offset)'

| "Offset" | $\bullet$ <br> $\bullet$ <br>  <br> $\bullet$ <br> as a a n IEC variable |
| :--- | :--- |

[^40]| "Factor" | $\bullet \quad$ as a number, including REAL |
| :--- | :--- |
|  | $\bullet \quad$ as an IEC variable |

Input field 'Li- The input variable is converted with the linear equation defined below. near scaling 1 (factor and offset)'

| "Factor" | $\bullet$ <br>  <br> $\bullet$ <br> •as a number, including REAL <br> "Offset"$\|$$\bullet$ as a number, including REAL <br>  <br> $\bullet$ <br> as an IEC variable |
| :--- | :--- |

Input field 'Li- The input variable is converted to be within a target range. CODESYS internally creates a linear near scaling 2 equation from the following input values.
(Base and target range)'

| "Base start value" | Lowest possible value for the input variable. <br> - as a number, including REAL <br> - as an IEC variable |
| :---: | :---: |
| "Base end value" | Highest possible value for the input variable. <br> - as a number, including REAL <br> - as an IEC variable |
| "Target start value" | Lowest possible value for the output variable. <br> - as a number, including REAL <br> - as an IEC variable |
| "Target end value" | Highest possible value for the output variable. <br> - as a number, including REAL <br> - as an IEC variable |

Example Conversion of electric current from a 10-bit input signal to an amperage range of 4-20 mA

| "Base start value" | 0 |
| :--- | :--- |
| "Base end value" | 1024 |
| "Target start value" | 4.0 |
| "Target end value" | 20.0 |

Input field 'User defined conversion'

| "Convert :=" | Conversion rule as mathematical function of $r$ Value The input variable is <br> rValue. |
| :--- | :--- |
| "Reverse :=" | Reverse function of the function defined in "Convert" |

Input field Use this conversion rule when you want to apply a conversion that is language-specific or 'Switchable con- variable-dependent. version'

| "Switchable conversion name" | Selected from a list of predefined conversion rules. Double-click directly into the <br> field for editing. |
| :--- | :--- |
| "Condition setting""" | Configured condition. Click into the input fields in "Condition setting" to edit the <br> condition. |

Example The Conv_A_LanguageDependent conversion rule that defines which conversion rule is executed for the English or German language.

| "Name" | "Type" | "Setting" | "Condition" | "Condition <br> setting" |
| :--- | :--- | :--- | :--- | :--- |
| Conv_A_Langu <br> ageDependent | "Switchable <br> conversion" | Conv_AInInch <br> , Conv_AInMM | "Language" |  |


| "Switchable conversion name" | "Condition setting" |
| :--- | :--- |
| Conv_AInInch | en |
| Conv_AInMM | de |

See also

### 6.4.1.21.3 Menu Commands

## Genera

By default the most important commands are already provided in the CODESYS user interface. If you want to customize the menu configuration individually, choose command "Tools $\rightarrow$ Customize $\rightarrow$ Menu".

When you have installed any packages or add-ons, additional menus and commands might be available.

## Menu 'File'

6.4.1.21.3.2.1 Command 'New Project' ..... 2552
6.4.1.21.3.2.2 Command 'Open Project' ..... 2553
6.4.1.21.3.2.3 Command 'Close Project' ..... 2554
6.4.1.21.3.2.4 Command 'Save project'. ..... 2554
6.4.1.21.3.2.5 Command 'Save Project as'. ..... 2555
6.4.1.21.3.2.6 Command 'Save Project and Install into Library Repository'. ..... 2556
6.4.1.21.3.2.7 Command 'Save Project as Compiled Library'.. ..... 2556
6.4.1.21.3.2.8 Command 'Save/Send Archive' ..... 2557
6.4.1.21.3.2.9 Command 'Extract Archive'. ..... 2558
6.4.1.21.3.2.10 Command 'Source Upload', ..... 2559
6.4.1.21.3.2.11 Command 'Source Download'. ..... 2560
6.4.1.21.3.2.12 Command 'Print' ..... 2560
6.4.1.21.3.2.13 Command 'Print Preview'. ..... 2560
6.4.1.21.3.2.14 Command 'Page Setup'. ..... 2560
6.4.1.21.3.2.15 Command 'Recent Projects' ..... 2561
6.4.1.21.3.2.16 Command 'Exit'. ..... 2561

## Command 'New Project'

Symbol: 徨, Shortcut: [Ctrl] + [N]
Function: This command opens the "New Project" dialog box for the creation of a new project file.
Call: "File" menu
'New Project' Function: Selection of a project category and a project template. dialog box

## Call: "File $\rightarrow$ New Project"

Depending on the template, you obtain a project that is automatically equipped with a certain range of objects.

Table 465: "Categories"

| "Libraries" |  |
| :--- | :--- |
| "Projects" |  |

Table 466: "Templates"

| "Projects" category:: |  |
| :--- | :--- |
| "Empty project" | Contains only the "Project Settings" object |
| "Standard project" | Contains a basic range of objects and libraries. A wizard assists with the creation <br> -see below. |
| "Standard project with |  |
| Application Composer" | Contains a basic range of objects and libraries for working with the Application <br> Composer. A wizard assists with the creation. |
| "Libraries" category: |  |
| "CODESYS container library" | Library that contains only further libraries, but no function blocks of its own. |
| "CODESYS interface library" | Library only for the definition of the interface of a software component. Thus <br> contains only objects that do not generate any code (constants, structures, inter- <br> faces, etc.). |
| "Empty library" | Contains only the "Project Settings" object |


| "External CODESYS library" | Target-system-specific library which is implemented as part of the runtime <br> system (in ANSI C or C++). |
| :--- | :--- |
| "Name" | Name of the project to be created. Depending on the template, a standard name <br> appears. The numerical suffix ensures the uniqueness of the name in the file <br> system. <br> You can change the file name, taking into consideration the file path conventions <br> of the operating system. Periods are not permitted in names. <br> CODESYS automatically adds the appropriate file extension to the selected <br> template. |
| "Location" | Location for the new project file. <br> ". opens a dialog box for browsing the file system. <br> - displays the history of previously entered paths |
| "OK" | CODESYS opens a new project. An error symbol $\mathbf{e}$ next to the input field draws <br> attention to missing specifications. If you place the mouse pointer on it, a tooltip <br> appears, informing you what to do. |

"Standard
Project" dialog
box

Function: Wizard for the creation of a standard project.
Project" dialog box

Call: Command "File $\rightarrow$ New Project"; in the "New Project" dialog box, select the "Projects" category and the "Standard project" template and click on "OK".

| "Device" | Selection list with PLC devices. The selected device is inserted as an object in <br> the Devices view below the root node |
| :--- | :--- |
| "PLC_PRG in" | Selection list with the programming languages. The automatically inserted pro- <br> gram PLC_PRG is created in the selected language. |

## Command 'Open Project'

Symbol: ※̋; shortcut: [Ctrl]+[O]
Function: The command opens the default dialog for loading a project. You can search for a CODESYS project in the file system and open it in the development system.
Call: Menu bar: "File"

## Dialog 'Open <br> Project'

| "File type" | Type of the CODESYS project to be loaded to the development system |
| :--- | :--- |
| "All supported files" | Filters by all projects which CODESYS can load <br> Hint: For example, you can select pro projects which have been created with <br> CoDeSys V2.3. These kinds of projects are also converted. |
| File extension project | Filters by projects which have been created with CODESYS V3 |
| File extension <br> projectarchive | Filters by project archives which have been created with CODESYS V3 |


| File extension library | Filters by library projects which have been created with CODESYS V3 |
| :--- | :--- |
| "Open" | Loads the project you selected to CODESYS <br> Note: Depending on the state of your CODESYS installation, it may be neces- <br> sary to update or supplement the installation. If this is the case, then first open <br> another "Open Project" dialog with options for installation management. |

See also

- $\&$ Chapter 6.4.1.3.2 "Opening a V3 Project" on page 1808
- „y Chapter 6.4.1.3.3 "Opening a V2.3 project" on page 1809


## Command 'Close Project'

Function: This command closes the currently opened project. CODESYS remains opened.
Call: "File" menu. In addition implicitly when opening a new/other project, while another project is still open.
If the project contains unsaved changes, a query appears, asking whether the project should be saved.
If you have not yet explicitly saved the project, a query appears asking whether you wish to delete the project files.

## Command 'Save project'

Symbol: ${ }^{[ }$, shortcut [Ctrl] + [S]
Function: this command saves the project file.
Call: "File" menu
This command saves the project file with the current project name, which appears in the title bar of the main window. If the project has been changed since it was last saved, the project name is given an asterisk.

The command is not available if the project is read-only.
Write protection exists if

- the project is identified in the project information (summary) as 'Released'
- the option "Open read-only" was selected in the dialog box "Open Project" when opening the project
Write protection is indicated by a line in the top right corner of the main window. A mouse-click on this line brings up a menu with commands for the possible actions:
- "Save project under a different file name on the disk": a mouse-click on this option leads to 'Save file as...'
- "Exit read-only mode": appears only if the option "Open read-only" was selected when opening the project.
- "Remove read-only attribute from the project on the disk": appears only if the project file had been provided with the property 'Read-only' on the disk at the time of opening.
- "Remove identification 'Released' in the project information": appears only if this attribute is currently set.


## Backup copy

Optionally a backup copy of the project file can be created. If the option "Create backup copy" is activated in the option dialog box 'Load and Save', the project is additionally copied to a file <projectname.backup> each time the project is saved.

## See also

- « Chapter 6.4.1.21.3.2.5 "Command 'Save Project as"' on page 2555
- \& Chapter 6.4.1.6.9 "Saving the Project" on page 1831
- \& Chapter 6.4.1.21.4.14.17 "Dialog 'Options' - 'Load and Save"' on page 2791


## Command 'Save Project as'

This command opens the standard Windows dialog box for saving a file. The project can be stored with the desired location and file type.

| "File type" | For both normal projects and library projects, this drop-down list contains the respective versions of the development system for which the project can be saved. If the current project contains add-ons that are not available in the selected memory format (profile), then the "Extend Profile" dialog box opens. <br> - "Project files (CODESYS v<version>) (*.project)": The project is saved as a CODESYS project file "<project name>.project " for the currently used or selected version of the development system. <br> - "Library files (CODESYS v<version>) (*.library)": The project is saved as a CODESYS library file "<project name>.library" for the currently used or selected version of the development system. <br> If the project should be opened later in an older version, then it makes sense to save for precisely this version, as you will then be informed immediately about possible data loss. |
| :---: | :---: |

Before saving a project as a library:

- Make sure that the rules for creating libraries have been followed.
- If it is to be possible to configure global constants provided by the library at a later time in an application, then you must define them in a parameter list. A parameter list is a special type of global variable list.
- When saving the project, no automatic check for errors is performed.
- Unlike CoDeSys V2.3, there is no distinction between 'external' and 'internal' libraries. Now you can define in the properties of each individual project object whether or not it should be treated as 'external'.
- Consider whether the library created is to be installed in the system library repository immediately. If so, then use the command 'Save project and install in the library repository'.
- If you want to protect the library project from later changes, then set the "Released" attribute in the "Project Information" dialog box. At the next attempt to save the project, a corresponding message will be displayed and the user must respond to the write protection with deliberate actions.
- If you save the project as a version of the development system other than the one currently in use, then you will be informed first about possible data loss.

Dialog box 'Extend Profile'

In this dialog box, the selected profile (memory format) can be extended by the add-ons that are contained in the current project. The profile is saved temporarily and then deleted after being saved or exported.

| "Add to profile" | Ø: The current profile is extended by the add-on so that the add-on data of the <br> current project is also saved. |
| :--- | :--- |
| "Add-on" | The add-on of the current project that is not contained in the selected memory <br> format. |
| "Version" | Version of the "Add-on" included in the current profile. <br> If several versions are installed, then the version can be selected. |


| "Save profile" | Opens the "Enter Profile Name" dialog box. In this dialog box, specify the name <br> for the new profile. The new profile is saved permanently at \$ProgramData\$/ <br> \$PRODUCT\$/CustomInformationalProfiles. |
| :--- | :--- |
| "Use saved profile" | The profile which was permanently saved in "Save profile" is used for saving or <br> exporting the current project. |



## Command 'Save Project and Install into Library Repository'

Function: this command saves the project as a library in the 'system' library repository.
Call: Main menu "File".
With this command CODESYS saves the project as a library in the 'system' library repository. This is an extension to the saving of a project as a library file using the "Save Project as" command. The library is installed on the local system and is immediately available for insertion into a project.
See also

- ${ }^{\circ}>$ Chapter 6.4.1.21.3.2.5 "Command 'Save Project as'" on page 2555


## Command 'Save Project as Compiled Library'

Function: The command saves a library project in encrypted form.
Call: Menu bar: "File"
The command opens the default dialog for saving a file in the file system. The "Compiled CODESYS Libraries" file type is already preset. The file extension is . compiled-library-v3 or . compiled-library (CODESYS < SP15). In this format, then source code of the library POU is not visible when the library is used in a project.

If the "Enforce signing of compiled libraries" option is selected in the "Security Screen" view on the "User" tab, then a library project has to be provided with a digital certificate-based signature when being saved. When a suitable certificate is available, it is provided in the "Security Screen" on the "User" tab in the "Digital Signature" section. In the "Project Information", on the "Summary" tab, a "Library compatibility" with a CODESYS version >= V3 SP15 is set by default. In this case, the project file is stored with the file extension . compiled-library-v3 when being saved as a compiled and signed library. If you still have not specified a suitable valid certificate for your user profile in the "Security Screen", then a dialog prompt opens next for you to do this. Afterwards, you can execute the save command again.

In all other respects, compiled library files behave just like *.library files, and therefore they can be installed and referenced with the same steps.
We recommend the use of compiled libraries signed with certificates. Besides the protection of the source code and the unauthorized use of a library, less memory is also used which therefore results in shorter loading times.

If you have the corresponding help files with translations, then as of CODESYS V3 SP15 you can extend the library documentation with the translation into other languages. This is done as follows:
Place the files created for the new languages __lmd_<language>. aux in a directory <library name>. Imd parallel to the library project <library name>. compiled-library-v3. If the files are correct, then they are included in the compiled library file when saving the library project by means of the "Save Project as Compiled Library" command.

Example: The directory standard. Imd is exists parallel to the library file standard. compiled-library-v3 and contains the file__lmd_fr.aux with the French translation of the library documentation. After the compiled library is saved, the French version of the documentation is also available in the Library Manager.

See also

- \& "Tab 'Summary"' on page 2515
- $\Leftrightarrow$ Chapter 6.4.1.21.3.4.18 "Command 'Security Screen'" on page 2592
- \& Chapter 6.4.1.17.3 "Information for Library Developers" on page 2035
- ${ }^{\text {- }}$, Chapter 6.4.1.21.3.2.6 "Command 'Save Project and Install into Library Repository'" on page 2556


## Command 'Save/Send Archive'

Function: This command opens the dialog "Project Archive" for the configuration of project archives.

## Call: Menu bar: "File $\rightarrow$ Project Archive"

An archive file (*.projectarchive) contains all files contained and referenced in the currently opened project. It can either be saved or dispatched as an e-mail attachment. The dispatch by email is very helpful for providing an employee with all project-relevant files. The file can be simply unpacked again with the command "Extract Archive".

## NOTICE!

The archiving function is not intended for the storage of a project, but rather for the simple summarizing of all project-relevant files.

See also

- $\boldsymbol{y y}^{4}$ Chapter 6.4.1.6.10 "Saving/Sending the project archive" on page 1832
- $\sum^{*}$ Chapter 6.4.1.21.3.2.9 "Command 'Extract Archive"' on page 2558

Dialog 'Project The dialog displays all the categories that can be added to the project archive. In this dialog, Archive' complete categories or individual objects from the categories can be added to the project archive by setting a check mark ( $\square$ ).


Entries that are display as red in the list require your attention. Move the mouse pointer over this library for more information.

| "Additional Files" | Opens the dialog "Additional Files". Here, further files can be added to the <br> archive with the "Add" button. |
| :--- | :--- |
| "Comment" | Opens the "Comment" dialog. Here, comments can be added to the archive. |
| "Save" | Creates the archive file and saves it. The storage location and the archive name <br> are specified in the subsequent dialog |
| "Send" | Creates a temporary archive file that is attached to an empty e-mail. A cor- <br> rect installation of the MAPI (Messaging Application Programming Interface) is <br> required for the successful execution of this operation. Failure is documented <br> by the display of a corresponding error message. The temporary archive is <br> automatically deleted after sending the e-mail. |

## Command 'Extract Archive'

Function: The command extracts a project archive, that was created with the command "Save/ Send Archive". You have to configure which objects of the archive CODESYS shall extract and in which directory of the file system they will be copied.

Call: Main menu "File $\rightarrow$ Project Archive"
The file extension of an archive is . projectarchive.
After the archive is selected, the dialog "Extract Project Archive" opens to configure the extract parameters.

Dialog Box 'Ex- This dialog box shows the contents of the project archive. You can exclude complete categories tract Project or single objects from categories by clearing the check boxes ( $\boxed{\square}$ ) from the extraction.
Archive'

Table 467: "Locations"

| "Extract into the same folder <br> where the archive is located" | The archive is extracted to the same directory. |
| :--- | :--- |
| "Extract into the following <br> folder" | The contents of the archive are extracted to the given path. |
| "Advanced" | Opens the "Advanced" dialog box for you to define where special and additional <br> files from the archive are extracted. |

Table 468: "Contents"

| "Items" | Shows the contents of the archive structured in object categories. <br> $\square:$ The object is extracted. <br> $\square$ |
| :--- | :--- |
| "Come object is not extracted. |  |


| "Extract" | If an extracted file has the same name as an existing file in the target directory, <br> then a dialog box opens, prompting whether the local file should be replaced. <br> The decision can be applied automatically to any additional conflicting names. In <br> this case, you have to select the $\square$ "Apply to all objects and files" check box. |
| :--- | :--- |

## Dialog 'Advan- <br> ced'

Table 469: "Repositories"

| "Install devices into" | Drop-down list with currently available repositories. Select the repositories, in <br> which CODESYS shall install the devices and the libraries of the archive. |
| :--- | :--- |
| "Install libraries into" |  |

Table 470: "Additional Files"

| By default the "additional files" are set to "Do not extract". Select the entries in the table and chose one of the <br> following options: |  |
| :--- | :--- |
| "Extract into project folder" | Folder of the project file |
| "Extract into folder" | User defined folder |
| "Do not extract" | Default |

See also

- ${ }^{4}$ Chapter 6.4.1.21.3.2.8 "Command 'Save/Send Archive"" on page 2557


## Command 'Source Upload'

Function: This command loads the project source code (as project archive) from the controller.
Call: Main menu "File".
Requirement: The network path for the controller must be configured.
After you execute the command, an overview opens with all devices in the network. Select a controller from this overview. The dialog box "Extract Project Archive" then opens with export settings.
See also

- ${ }^{4}$ Chapter 6.4.1.21.3.2.11 "Command 'Source Download'" on page 2560

Dialog Box 'Ex- This dialog box shows the contents of the project archive. You can exclude complete categories tract Project or single objects from categories by clearing the check boxes ( $\square$ ) from the extraction. Archive'

Table 471: "Locations"

| "Extract into the same folder <br> where the archive is located" | The archive is extracted to the same directory. |
| :--- | :--- |
| "Extract into the following <br> folder" | The contents of the archive are extracted to the given path. |
| "Advanced" | Opens the "Advanced" dialog box for you to define where special and additional <br> files from the archive are extracted. |

Table 472: "Contents"

| "Items" | Shows the contents of the archive structured in object categories. <br> $\square:$ The object is extracted. |
| :--- | :--- |
|  | $\square:$ The object is not extracted. |


| "Extract" | If an extracted file has the same name as an existing file in the target directory, <br> then a dialog box opens, prompting whether the local file should be replaced. <br> The decision can be applied automatically to any additional conflicting names. In <br> this case, you have to select the <br> $\square$ |
| :--- | :--- |
| "Apply to all objects and files" check box. |  |

## Command 'Source Download'

Function: This command loads the project source code (as project archive) to the controller.
Call: Main menu "File".
Requirement: The network path for the controller must be configured.
After you execute the command, an overview opens with all devices in the network. Select a controller from this overview. Then the Archiv.prj project archive is downloaded to this controller. You can click "Source Upload" to upload the complete source code to the CODESYS development system at a later time.
If you are already connected to a controller (in online mode), then the "Source Download to Connected Device" command is also available for this process.

See also
-

- ${ }^{4}$ Chapter 6.4.1.21.3.2.10 "Command 'Source Upload'" on page 2559
- $y^{*}$ Chapter 6.4.1.21.3.7.7 "Command 'Source Download to Connected Device"" on page 2631


## Command 'Print'

## Symbol:

Function: This command opens the default Windows dialog box for printing documents.
Call: Main menu "File"
See also

- ${ }^{\circ} \boldsymbol{y}$ Chapter 6.4.1.21.4.12.7 "Dialog 'Project Settings' - 'Page Setup'" on page 2772


## Command 'Print Preview'

Function: This command opens a print preview for the currently open element.
Call: Main menu "File"
Requirement: An object is open in the editor.
See also

- *y Chapter 6.4.1.21.4.12.7 "Dialog 'Project Settings' - 'Page Setup'" on page 2772
-     * $\boldsymbol{y}^{\boldsymbol{y}}$ Chapter 6.4.1.21.3.2.12 "Command 'Print'" on page 2560


## Command 'Page Setup'

Symbol:
Function: This command opens the "Page Setup" dialog box for configuring the layout of the printed version of the project contents.
Call: Main menu "File $\rightarrow$ Page Setup"

See also

- ${ }^{*} \boldsymbol{y}$ Chapter 6.4.1.21.4.12.7 "Dialog 'Project Settings' - 'Page Setup'" on page 2772
- $\quad$ Chapter 6.4.1.21.3.2.12 "Command 'Print"" on page 2560


## Command 'Recent Projects'

Function: Opens the list of the projects used recently, from which you can select a project to open.

Call: "File" menu

## Command 'Exit'

## Shortcut: [<Alt>]+[<F4>]

Function: this command exits from the programming system. If a project is currently opened that has been changed since it was last saved, a dialog box opens asking whether the project should be saved.

Call: "File" menu

## Menu 'Edit'

6.4.1.21.3.3.1 Standard Commands ..... 2562
6.4.1.21.3.3.2 Command 'Find', 'Find in Project'. ..... 2563
6.4.1.21.3.3.3 Command 'Replace', 'Replace in Project' ..... 2564
6.4.1.21.3.3.4 Command 'Find Next'. ..... 2565
6.4.1.21.3.3.5 Command 'Find Next (Selection)'. ..... 2565
6.4.1.21.3.3.6 Command 'Find Previous'.... ..... 2565
6.4.1.21.3.3.7 Command 'Find Previous (Selection)' ..... 2566
6.4.1.21.3.3.8 Command 'Insert File as Text' ..... 2566
6.4.1.21.3.3.9 Command 'Overwrite Mode'. ..... 2566
6.4.1.21.3.3.10 Command 'View Whitespace'. ..... 2566
6.4.1.21.3.3.11 Command 'View Indentation Guides'. ..... 2567
6.4.1.21.3.3.12 Command 'Go to Line'... ..... 2567
6.4.1.21.3.3.13 Command 'Make Uppercase' ..... 2567
6.4.1.21.3.3.14 Command 'Make Lowercase'. ..... 2567
6.4.1.21.3.3.15 Command 'Go to Matching Bracket'. ..... 2568
6.4.1.21.3.3.16 Command 'Select to Matching Bracket'. ..... 2568
6.4.1.21.3.3.17 Command 'Expand All Folds'. ..... 2568
6.4.1.21.3.3.18 Command 'Collapse All Folds' ..... 2568
6.4.1.21.3.3.19 Command 'Comment Out Selected Lines'. ..... 2569
6.4.1.21.3.3.20 Command 'Uncomment Selected Lines'. ..... 2569
6.4.1.21.3.3.21 Command 'Enable Inline Monitoring' ..... 2569
6.4.1.21.3.3.22 Command 'Toggle Bookmark'. ..... 2569
6.4.1.21.3.3.23 Command 'Next Bookmark (Active Editor)'. ..... 2570
6.4.1.21.3.3.24 Command 'Next Bookmark'. ..... 2570
6.4.1.21.3.3.25 Command 'Previous Bookmark (Active Editor)' ..... 2570
6.4.1.21.3.3.26 Command 'Previous Bookmark'. ..... 2570
6.4.1.21.3.3.27 Command 'Clear All Bookmarks (Active Editor)'. ..... 2571
6.4.1.21.3.3.28 Command 'Clear All Bookmarks'. ..... 2571
6.4.1.21.3.3.29 Command 'Browse Cross References'. ..... 2571
6.4.1.21.3.3.30 Command 'Browse Global Cross References'. ..... 2571
6.4.1.21.3.3.31 Command 'Browse Call Tree'. ..... 2572
6.4.1.21.3.3.32 Command 'Auto Declare'. ..... 2572
6.4.1.21.3.3.33 Command 'Input Assistant'. ..... 2574
6.4.1.21.3.3.34 Command 'Go to Source Position'. ..... 2575
6.4.1.21.3.3.35 Command 'Next Message'. ..... 2576
6.4.1.21.3.3.36 Command 'Previous Message'. ..... 2576
6.4.1.21.3.3.37 Command 'Go to Definition'. ..... 2576
6.4.1.21.3.3.38 Command 'Go To Reference'. ..... 2576
6.4.1.21.3.3.39 Command 'Go to Instance'. ..... 2577
6.4.1.21.3.3.40 Command 'Refactoring' - 'Rename <..>'. ..... 2577
6.4.1.21.3.3.41 Command 'Refactoring' - 'Update Referenced Pins'. ..... 2578
6.4.1.21.3.3.42 Command 'Refactoring' - 'Add Variable'. ..... 2578
6.4.1.21.3.3.43 Command 'Refactoring' - 'Remove <variable>' ..... 2580
6.4.1.21.3.3.44 Command 'Refactoring' - 'Reorder Variables'. ..... 2581
6.4.1.21.3.3.45 Command 'Advanced' - 'Format Document'. ..... 2581

## Standard Commands

CODESYS provides the following standard commands:
－Undo：$\curvearrowleft$ ，shortcut：［Ctrl］＋［Z］
－Redo：$c »$ ，shortcut：［Ctrl］＋［Y］

- Cut：品，shortcut：［Ctrl］＋［X］
- Copy：踃，shortcut：［Ctrl］＋［C］
- Paste：賭，shortcut：［Ctrl］＋［V］
－Delete：$\times$ ，shortcut：［Ctrl］
－Select all：shortcut：［Ctrl］＋［Ctrl］
Not all editors support the＂Insert＂command．In some editors it can be used with limitations． Graphical editors only support the command if the pasted elements will create a valid construct．

In object trees like POUs or device view the command refers to the currently selected object． Multi selection is possible．

## Command＇Find＇，＇Find in Project＇


Symbol 略，keyboard shortcut［Ctrl］＋［Shift］＋［F］
Function：These commands scan the project or parts of it for a specified character string．
Call：Menu bar：＂Edit $\rightarrow$ Find Replace＂
This command opens the＂Find＂dialog where the searched character string is specified and the search options are defined．

Dialog＇Find＇

| ＂Search for＂ | Character string to be searched． |
| :---: | :---: |
| ＂Match case＂： | $\square$ ：The search considers uppercase and lowercase． |
| ＂Match whole word＂： | V：Only character strings are found that exact matches． |
| ＂Search up＂： | The specified search range runs upwards． <br> ］：The specified search range runs downwards． |
| ＂Use regular expressions＂： | Use the $\geqslant$ button to receive support when specifying regular expressions． |
| ＂Search in＂ | $\checkmark$ ：Drop－down list with the areas of the project to be searched： <br> －＂Active editor＂ <br> －＂All open editors＂ <br> －＂Selected objects \＆Subobjects＂ <br> －＂Entire project＂ <br> －＂Entire project \＆Uncompiled libraries＂ <br> －＂Selection only＂ <br> ．．．．：Opens a dialog where you set the areas of the project to be searched（see below） |
| ＂Find next＂ | Start the search |
| ＂Find all＂ | All search results are listed in the message view with their object path，project name，object name，and object position．Possible additional information for posi－ tion：＂（Decl）＂＝Declaration part of the object；＂$(\mathrm{mpl})$＂＝Implementation part of the object <br> Double－clicking the entry in the list opens the match position in the respective object editor． |
| ＂Replace＂ | Switches to the＂Replace＂dialog |



## See also

- «y "Tab 'Text Area'" on page 2799


## Dialog for setting the objects to be searched

| "Entire project" | All editable positions in all objects of the project are searched. |
| :---: | :---: |
| "Entire project and all uncompiled libraries" | All editable positions in all objects of the project, including integrated uncompiled libraries, are searched. |
| "Within the following objects" | Only the editable positions within the objects defined here are searched: |
|  | - "Scheme": The "Save" command saves the current search configuration by the specified name. All saved schemes are available in the drop-down list ( $\vee$ ). <br> - "Object types": $\checkmark$ : The object is searched. <br> - "Name filter": Name filter for the searched objects. The placeholder "*" can be used. <br> Example: Filter "*CAN*": All objects are searched that have "CAN" in the name. |
| "All open editors" | All editors are searched that are currently open in a window. |
| "Active editor" | Only the editor is searched where the cursor currently is. |
| "Selection only" | Only the text is searched that is currently selected in an object. |

See also

- ${ }^{4}$ Chapter 6.4.1.21.3.3.3 "Command 'Replace', 'Replace in Project'" on page 2564
- $\Longleftrightarrow$ Chapter 6.4.1.9.15 "Searching and replacing in the entire project" on page 1910


## Command 'Replace', 'Replace in Project'

Symbol ${ }^{A}$ ? ; keyboard shortcut: $[\mathrm{Ctrl}]+[H]$
Symbol ${ }^{A}$, keyboard shortcut [Ctrl]+[Shift] $+[H]$
Function: These commands scan the project or parts of it for a specified character string and replaces it.
Call: Menu bar: "Edit $\rightarrow$ Find Replace"
Requirement: The application is in online mode.
This command opens the "Replace" dialog where the search and replace character strings are specified and the search options are defined.

Table 473: In addition to the options of the "Search" dialog, the following settings are still possible:

| "Replace with" | Input field for the new character string. |
| :--- | :--- |
| "Replace" | Each next found string is highlighted in the editor and replaced (step-by-step <br> replace). |


| "Replace all" | All found strings are replaced at one time without them being displayed in the <br> editors. |
| :--- | :--- |
| "Leave changed objects open <br> after "Replace all"" | The editors of the found objects remain open. |

Replacement in referenced libraries is not possible.

See also

- $\quad \begin{gathered} \\ \text { - }\end{gathered}$ Chapter 6.4.1.21.3.3.2 "Command 'Find', 'Find in Project'" on page 2563
- $\Leftrightarrow$ Chapter 6.4.1.9.15 "Searching and replacing in the entire project" on page 1910


## Command 'Find Next'

Symbol ${\underset{n}{n}, \text {, keyboard shortcut [F3] }}^{2}$
Function: During the search for a certain string within the project, this command selects the next match at its position in the respective editor.
Call: Menu "Edit $\rightarrow$ Search Replace"
Requirement: You have already started searching the project for a certain string by using the commands "Find" or "Replace".

See also

- $\sum^{\wedge}$ Chapter 6.4.1.21.3.3.2 "Command 'Find', 'Find in Project"" on page 2563
- $\sum^{4}$ Chapter 6.4.1.21.3.3.3 "Command 'Replace', 'Replace in Project"' on page 2564
- $\Longleftrightarrow$ Chapter 6.4.1.9.15 "Searching and replacing in the entire project" on page 1910


## Command 'Find Next (Selection)'

Keyboard shortcut [Ctrl] + [F3]
Function: The command searches the project for the next string matching the string which is currently selected or in which you have currently placed the cursor.
Call: Menu "Edit $\rightarrow$ Find Replace"
Requirement: You have the cursor placed in an editable string in your project, or you have selected an editable string.
See also

- ${ }^{*}$ Chapter 6.4.1.21.3.3.2 "Command 'Find', 'Find in Project"" on page 2563
- $\quad$, Chapter 6.4.1.21.3.3.3 "Command 'Replace', 'Replace in Project"" on page 2564
- $\Leftrightarrow$ Chapter 6.4.1.9.15 "Searching and replacing in the entire project" on page 1910


## Command 'Find Previous'

Symbol
Function: During the search for a certain string within the project, this command selects the next match at its position in the respective editor.
Call: Menu "Edit $\rightarrow$ Search Replace"
Requirement: You have already started searching the project for a certain string by using the commands "Find" or "Replace".

See also

- ${ }^{*}$ Chapter 6.4.1.21.3.3.2 "Command 'Find', 'Find in Project"" on page 2563
- $\quad>$ Chapter 6.4.1.21.3.3.3 "Command 'Replace', 'Replace in Project"' on page 2564
- ${ }^{*}$ Chapter 6.4.1.9.15 "Searching and replacing in the entire project" on page 1910


## Command 'Find Previous (Selection)'

Keyboard shortcut [Ctrl] + [Shift] + [F3]
Function:The command searches the project for the previous string matching the string which is currently selected or in which you have currently placed the cursor.

## Call: Menu "Edit $\rightarrow$ Find Replace"

Requirement: You have the cursor placed in an editable string in your project, or you have selected an editable string.
See also

- ${ }^{*}$ ) Chapter 6.4.1.21.3.3.2 "Command 'Find', 'Find in Project"" on page 2563
- $\quad>$ Chapter 6.4.1.21.3.3.3 "Command 'Replace', 'Replace in Project"" on page 2564
- ${ }^{*}$ Chapter 6.4.1.9.15 "Searching and replacing in the entire project" on page 1910


## Command 'Insert File as Text'

Function: This command copies the contents of a text file to the active editor as the current cursor position.

Call: The command is not in any menu by default. You can add it to a menu by using the dialog box from "Tools $\rightarrow$ Customize" (command category "Text Editor").

Requirement: The file must have the extension .txt. The command is available in a text editor only.
Many development environments and text processing applications provide the option of exporting code and text as a plain text file. This command can copy the contents of this file to the editor.
See also

- $\quad \Leftrightarrow$ Chapter 6.4.1.21.4.14.26 "Dialog 'Options' - 'Text Editor"' on page 2798


## Command 'Overwrite Mode'

Shortcut: [Insert]
Function: This command activates the overwrite mode.
Call: Menu "Edit $\rightarrow$ Advanced"
Requirement: A text editor is opened.
If the overwrite mode is activated, characters in front of the cursor are overwritten when entering new characters. If the overwrite mode is deactivated, characters are inserted and existing characters in front of the cursor are retained.

See also

- 出 Chapter 6.4.1.21.4.14.26 "Dialog 'Options' - 'Text Editor'" on page 2798


## Command 'View Whitespace'

Symbol: a.b

Function: This command causes control characters for spaces and tabs to be shown.
Call: Menu "Edit $\rightarrow$ Advanced"
Requirement: A text editor is opened.
CODESYS visualizes spaces by a period and tabs by an arrow.
See also

- ⓨ Chapter 6.4.1.21.4.14.26 "Dialog 'Options' - 'Text Editor"' on page 2798


## Command 'View Indentation Guides'

Function: This command activates the indentation help lines.
Call: Menu "Edit $\rightarrow$ Extended"
Requirement: A text editor is opened.
If the indentation help lines are activated, a broken line is inserted for each manual indentation in the code. This facilitates the overview of the different levels in the code. You can insert manual indentations with the [Tab] key.
See also

- ② Chapter 6.4.1.21.4.14.26 "Dialog 'Options' - 'Text Editor'" on page 2798


## Command 'Go to Line'

Function: With this command the cursor jumps to a defined line in the code.
Call: Menu "Edit $\rightarrow$ Extended"
Requirement: A text editor is opened.
This command opens a dialog box with an input field "Line number".
See also

- $\sum^{\circledR}$ Chapter 6.4.1.21.4.14.26 "Dialog 'Options' - 'Text Editor'" on page 2798


## Command 'Make Uppercase'

Shortcut: [Ctrl]+[Shift]+[U]
Function: This command converts all lowercase letters in the selected code into uppercase letters.
Call: Menu "Edit $\rightarrow$ Advanced"
Requirement: A text editor is opened and code is selected, or the declaration editor is opened and variable declarations are selected.

See also

- ${ }^{〔}$, Chapter 6.4.1.21.4.14.26 "Dialog 'Options' - 'Text Editor'" on page 2798


## Command 'Make Lowercase'

Shortcut: [Ctrl]+[U]
Function: This command converts all uppercase letters in the selected code into lowercase letters.
Call: Menu "Edit $\rightarrow$ Advanced"

Requirement: a text editor is opened and code is selected, or the declaration editor is opened and variable declarations are selected.
See also

- $\quad \Leftrightarrow$ Chapter 6.4.1.21.4.14.26 "Dialog 'Options' - 'Text Editor"' on page 2798


## Command 'Go to Matching Bracket'

Function: This command makes the cursor jump to the other part of the selected code parenthesis.
Call: Menu "Edit $\rightarrow$ Advanced"
Requirement: A text editor is opened and the cursor is positioned at an opening or closing code parenthesis. If you position the cursor at a code parenthesis, CODESYS displays the corresponding parenthesis in color, provided you have activated the option "Associated parentheses" in the CODESYS options in the "Text Editor" category, "Text Area" tab.
See also

- \& "Tab 'Text Area'" on page 2799


## Command 'Select to Matching Bracket'

Function: This command selects the entire code section within the currently selected code parentheses.

Call: Menu "Edit $\rightarrow$ Extended"
Requirement: A text editor is opened and the cursor is positioned at an opening or closing code parenthesis. If you position the cursor at a code parenthesis, CODESYS displays the corresponding parenthesis in color, provided you have activated the option "Associated parentheses" in the project options in the "Text Editor" category, "Text Area" tab.

See also

- ② Chapter 6.4.1.21.4.14.26 "Dialog 'Options' - 'Text Editor'" on page 2798


## Command 'Expand All Folds'

Function: This command expands all collapsed code segments in the textual editor or result locations in the cross-reference list so that the code and all search locations are displayed in full again.
Requirement: A textual editor is active and indentation is activated in the "Options" ("Text Editor" category); or the cross-reference list is active.
Call: Textual editors: main menu"Edit $\rightarrow$ Advanced", or right-click. In the cross-reference list: right-click.
See also

- ${ }^{2}$ ) Chapter 6.4.1.21.4.14.26 "Dialog 'Options' - 'Text Editor'" on page 2798
- ${ }^{4} \boldsymbol{y}$ Chapter 6.4.1.21.3.3.18 "Command 'Collapse All Folds'" on page 2568
- ㄹy "Right-click commands in the cross-reference list" on page 2589


## Command 'Collapse All Folds'

Function: This command collapses all expanded code segments in the textual editor or result locations in the cross-reference list. In this way, only the uppermost level of code and only the root node of the result locations displayed.

Requirement: A textual editor is active and indentation is activated in the "Options" ("Text Editor" category); or the cross-reference list is active.
Call: In textual editors: main menu "Edit $\rightarrow$ Advanced", or right-click. In the cross-reference list: right-click.

See also

- 出 Chapter 6.4.1.21.4.14.26 "Dialog 'Options' - 'Text Editor"' on page 2798
- $\sum^{4}$ Chapter 6.4.1.21.3.3.17 "Command 'Expand All Folds'" on page 2568
- ㄹy "Right-click commands in the cross-reference list" on page 2589


## Command 'Comment Out Selected Lines'

Symbol //c; keyboard shortcut: [Ctrl]+[O]
Function: The command inserts comment marks ('//') at the beginning of the selected lines.
Call: Menu bar: "Edit $\rightarrow$ Advanced"; context menu
Requirement: In the ST editor, either the cursor is located in a line of the implementation or multiple lines are selected.

See also

- ${ }^{\wedge}$ Chapter 6.4.1.21.3.3.20 "Command 'Uncomment Selected Lines'" on page 2569


## Command 'Uncomment Selected Lines'

Symbol //c; keyboard shortcut: [Ctrl]+[I]
Function: The command removes any comment marks ('//') at the beginning of the selected lines.
Call: Menu bar: "Edit $\rightarrow$ Advanced"; context menu
Requirement: In the ST editor, either the cursor is located in a line of the implementation or multiple lines are selected.

See also

- $\quad$ \& Chapter 6.4.1.21.3.3.19 "Command 'Comment Out Selected Lines'" on page 2569


## Command 'Enable Inline Monitoring'

Function: This command enables or disables the inline monitoring function. This works the same way as the check box with the same name in the CODESYS options ("Text Editor" category).
Requirement: A text editor is active.
Call: Context menu of the text editor in the "Advanced" submenu.
See also

- « "Tab 'Monitoring'" on page 2800
- ${ }^{4}$ " Chapter 6.4.1.13.2 "Monitoring of Values" on page 1995


## Command 'Toggle Bookmark'

Symbol ח, keyboard shortcut [Ctrl]+[F12]
Function: The command sets or removes a bookmark at the current position.
Call: Menu bar: "Edit $\rightarrow$ Bookmarks"

Requirement: A POU is open in the editor and the cursor is at a program line.
See also

- $\xi^{4}$ Chapter 6.4.1.9.14.3 "Setting and using bookmarks" on page 1908


## Command 'Next Bookmark (Active Editor)'

Symbol: 께; keyboard shortcut: [F12]
Function: The command jumps to the next bookmark in the active editor.
Call: Menu bar: "Edit $\rightarrow$ Bookmarks"
Requirement: A POU is open in the editor and the cursor is positioned in the POU.
See also

- $\Rightarrow$ Chapter 6.4.1.21.3.3.24 "Command 'Next Bookmark'" on page 2570
- ${ }^{\text {• }} \boldsymbol{y}$ Chapter 6.4.1.9.14.3 "Setting and using bookmarks" on page 1908


## Command 'Next Bookmark'

## Symbol: \#\#

Function: The command jumps to the next bookmark in the "Bookmarks" view and in the project, and opens the respective POU. The order of jumping to bookmarks corresponds to the order of bookmarks in the table of the "Bookmarks" view.
Call:

- $\rightarrow$ "Next Bookmark" button in the "Bookmarks" view
- The command is not in any menu by default. You can add it to a menu by using the dialog from "Tools $\rightarrow$ Customize" (command category "Bookmarks").


## Requirement:

- A project is open.
- The "Bookmarks" view is open.

See also

- ${ }^{\wedge}$ Chapter 6.4.1.21.3.3.23 "Command 'Next Bookmark (Active Editor)"' on page 2570


## Command 'Previous Bookmark (Active Editor)'

Symbol: *; keyboard shortcut: [Shift]+[F12]
Function: The command jumps to the previous bookmark in the active editor.
Call: Menu bar: "Edit $\rightarrow$ Bookmarks"
A POU is open in the editor and the cursor is positioned in the POU.
See also

- ${ }^{*}$ Chapter 6.4.1.21.3.3.26 "Command 'Previous Bookmark'" on page 2570
- \#y Chapter 6.4.1.9.14.3 "Setting and using bookmarks" on page 1908


## Command 'Previous Bookmark'

Symbol: *t
Function: The command jumps to the previous bookmark in the "Bookmarks" view and in the project, and opens the respective POU. The order of jumping to bookmarks corresponds to the order of bookmarks in the table of the "Bookmarks" view.

## Call：

－$\Rightarrow$＂Next Bookmark＂button in the＂Bookmarks＂view
－The command is not in any menu by default．You can add it to a menu by using the dialog from＂Tools $\rightarrow$ Customize＂（command category＂Bookmarks＂）．
Requirement：
－A project is open．
－The＂Bookmarks＂view is open．
See also
－${ }^{\text {• }}$ Chapter 6．4．1．21．3．3．25＂Command＇Previous Bookmark（Active Editor）＂＇on page 2570
－$\Rightarrow$ Chapter 6．4．1．9．14．3＂Setting and using bookmarks＂on page 1908

## Command＇Clear All Bookmarks（Active Editor）＇

Symbol：埚
Function：The command deletes all bookmarks in the active editor．
Call：Menu bar：＂Bookmarks＂
Requirement：A POU is open in the editor and the cursor is positioned in the POU．
See also
－$\Leftrightarrow$ Chapter 6．4．1．21．3．3．28＂Command＇Clear All Bookmarks＇＂on page 2571
－乡y Chapter 6．4．1．9．14．3＂Setting and using bookmarks＂on page 1908

## Command＇Clear All Bookmarks＇

Symbol：梋
Function：The command deletes all bookmarks in the open project．
Call：The command is not in any menu by default．You can add it to a menu by using the dialog from＂Tools $\rightarrow$ Customize＂（command category＂Bookmarks＂）．
Requirement：A POU is open in the editor and the cursor is positioned in the POU．
See also
－$\Rightarrow$ Chapter 6．4．1．21．3．3．27＂Command＇Clear All Bookmarks（Active Editor）＇＂on page 2571
－„y Chapter 6．4．1．9．14．3＂Setting and using bookmarks＂on page 1908

## Command＇Browse Cross References＇

Symbol：
Function：The command shows all occurrences of a variable in the＂Cross Reference List＂view．
Call：Menu bar：＂Edit $\rightarrow$ Browse＂；cross reference view：toolbar
Requirement：A POU is open in the editor and the cursor is set at a variable．Or the＂Cross Reference List＂view is open and a variable is specified in the＂Name＂field．
See also
－${ }^{\circledR}$ ，Chapter 6．4．1．21．3．3．30＂Command＇Browse Global Cross References＇＂on page 2571

## Command＇Browse Global Cross References＇

Symbol：

Function: The command shows all occurrences of all variables with the same name in the "Cross Reference List" view. In contrast to the "Browse Cross References" command, these can be different variables.
Call: Menu bar: "Edit $\rightarrow$ Browse"; cross reference view: toolbar
Requirement: A POU is open in the editor and the cursor is set at a variable. Or the "Cross Reference List" view is open and a variable is specified in the "Name" field.

See also

- ${ }^{*}>$ Chapter 6.4.1.9.14.1 "Using the cross-reference list to find occurrences" on page 1906
- $\quad>$ Chapter 6.4.1.21.3.3.29 "Command 'Browse Cross References'" on page 2571


## Command 'Browse Call Tree'

## Symbol: $=$

Function: The command opens the view "Call Tree", which displays the calls of a module and also its callers.

## Call:

- Menu "Edit $\rightarrow$ Browse"
- Context menu, see below: Requirement

Requirement: A module is opened in the editor and the cursor is placed in a variable, or a module is selected in the "Devices" view or in the "POUs" view.
See also

- $\Leftrightarrow$ Chapter 6.4.1.21.3.4.16 "Command 'Call tree"" on page 2590


## Command 'Auto Declare'

Keyboard shortcut: [Shift]+[F2]
Function: The command opens the "Auto Declare" dialog, which supports the declaration of a variable.

Call: Menu bar: "Edit"
Requirement: An object or a device of the project is opened in the editor.
With the auto-declaration function, the "Auto Declare" dialog also appears when the cursor is located in the implementation part of a POU in a line containing the name of an undeclared variable. The requirement for this is that you must have clicked "Tools $\rightarrow$ Options" and enabled the "Declare unknown variables automatically (AutoDeclare)" option in the "SmartCoding" category.
With the smart tag function, the "Auto Declare" command also appears when you place the cursor over an undeclared variable in the implementation part of the ST editor and then click $\quad$.

Dialog 'Auto
Declare'

| "Scope" | Scope of the variable that is not declared yet. <br> Example: VAR (default setting for local variables) |
| :--- | :--- |
| "Name" | Variable name not declared yet <br> Example: bIsValid |


| "Type" | Example: BOOL <br> - $\quad$ : Lists the standard data types. $\square$ <br> - "Input Assistant": Opens the "Input Assistant" dialog <br> - "Array Assistant": Opens the "Array" dialog |
| :---: | :---: |
| "Object" | Object where the new variable is declared. By default, the object that you are editing now. <br> Example: fbA <br> : Lists that objects where the variable can be declared. <br> If no objects are available for the selected "Scope", the entry "<create object>" appears. When you select the "<create object>" entry, the "Add Object" dialog opens for generating a suitable object. |
| "Initialization" | Example: FALSE <br> If you do not specify an initialization value, then the variable is initialized automatically. <br> ...: Opens the "Initialization Value" dialog. This procedure is helpful for the initialization of structured variables. |
| "Address" | Memory address of the application for the variable that is not declared yet. <br> Example: \%IX1.0 <br> Note: <br> Possible only for the following scopes: <br> - Local variable (VAR) <br> - Global variable (VAR_GLOBAL) <br> - Or for a persistent variable (PERSISTENT). |
| "Flags" | Atribute keywords <br> - COnstant: Keyword for a constant. <br> - RETAIN: Keyword for a remanent variable. <br> - persistent: Keyword for a persistent variable (stricter than Retain). <br> The selected attribute keyword is added to the variable declaration. |
| "Comment" | Example: New input In1 <br> In the tabular declaration editor, the comment entered is displayed in the "Comment" column, while in the textual declaration editor it is displayed above the variable declaration. |
| "Apply changes using refactoring" | When you exit the dialog, the variable is not declared yet, but then it opens the "Refactoring" dialog. You can continue editing your changes here. <br> The option appears for the following scopes: <br> - Input variable (VAR_INPUT) <br> - Output variable (VAR_OUTPUT) <br> - VAR_IN_OUT variables (input variable and output variable) |
| "OK" | The variable is declared and appears in the declaration. Example: ```VAR RETAIN // New input In1 xIn1 AT %IX1.0: BOOL := FALSE; END VAR``` |

## Dialog 'Array'

| "Ranks and base type <br> specification" | Definition of the field sizes ("Dimension") by entering the lower and upper limits <br> and the "Base type" of the array. You can enter the basic type directly or with the <br> help of the "Input Assistant" or "Array" dialogs when you click the $>$ button. |
| :--- | :--- |
| "Result" | Display of the defined array |

## NOTICE! <br> CODESYS reinitializes variables only if you have modified the initialization values of the variables.

## Dialog 'Initializa-

tion Value'

| List of the variables with name ("Expression"), "Initialization Value" and "Data Type". <br> Modified initialization values are displayed in bold fonts. |  |
| :--- | :--- |
| Input field below the list | Input of an initialization value for the selected variable(s) |
| "Apply value to selected lines" | Change of the initialization value of the selected line(s) according to the value of <br> the input field |
| "Reset selected lines to default <br> values" | Resets the default initialization values |
| "OK" | CODESYS applies the initialization values in the "Auto Declare" dialog. |

In the case that the variable to be initialized by means of this dialog is a function block instance with an extended FB_Init method, an additional table is displayed above the "Initialization Value" table. The additional FB_Init parameters are listed in this table. The meaning and operation essentially correspond to the lower table with the following differences:

- All variables have to be assigned with initialization values. Otherwise "OK" remains disabled.
- For complex data types (structures, arrays), no components contained within are displayed (type cannot be expanded). In this case, the complex type has to be initialized with a corresponding variable.
For FB_Init parameters configure this way, a corresponding symbol is displayed after the initialization value in the "Auto Declare" dialog.

See also

- ${ }^{*}$ Chapter 6.4.1.20.10 "Methods 'FB_Init', 'FB_Reinit', and 'FB_Exit'" on page 2336
- 范 "Dialog box 'Refactoring'" on page 2578
- ${ }^{4}$ Chapter 6.4.1.20.4.12 "Addresses" on page 2231
- ${ }^{\circ}$ Chapter 6.4.1.21.4.14.22 "Dialog 'Options' - 'Refactoring'" on page 2794
- $\Leftrightarrow$ Chapter 6.4.1.20.1.4.1 "ST Editor" on page 2049


## Command 'Input Assistant'

Symbol: 주; keyboard shortcut: [F2]
Function: This command opens the "Input Assistant" dialog which helps you to select one of the possible programming elements at the current cursor position.

Call: Menu bar: "Edit"; context menu.
Requirement: A POU is open in the editor and the cursor is at a program line.

Dialog 'Input The input assistant provides all program elements that you can insert at the current cursor

Assistant' - Tab
'Categories'
position in the editor.
The elements are sorted by "Categories". In the category "Variables", you can also set a "Filter" for the scope, for example "Local variables", "Global variables", or "Constants".

| "Structured view" | The elements are displayed in a structure tree. You can show/hide the columns "Type", "Address", and "Origin" by right-clicking the column title and selecting/clearing the column name in the dropdown list. $\square$ : The elements are displayed in a flat structure. |
| :---: | :---: |
| "Show documentation" | $\checkmark$ : The dialog is extended with the "Documentation" field. |
| "Insert with arguments" | Elements that include arguments (for example, functions) also insert with these arguments at the cursor position. <br> Example: If you insert the function block fb1, which contains an input variable fb1_in and an output variable fb1_out, "with arguments", then this appears in the editor as follows: $f$ b1 ( $\mathrm{fb} 1 \mathrm{in}:=$, fb1 out=> ). |
| "Insert with namespace prefix" | Inserts the selected element with the appended namespace. In the case of library modules, the check box remains disabled if the requirement for a namespace has been defined in the library properties. |

If you create objects with the same name in the same category, whether globally ("POUs" view) or assigned to an application ("Devices" view), then only one entry appears in the input assistant. The usage conforms to the usual call priority (application assigned before global).

Dialog 'Input
Assistant' - Tab
'Text Search'

This tab allows you to search for specific objects. When you begin typing a search string into the search field, the names of all objects are listed whose names include the search string. Double-click an object to insert it at the current cursor position in the editor.

| "Filters" | Limits the search to a specific variable category |
| :--- | :--- |

See also

- ${ }^{\sharp}$ Chapter 6.4.1.9.7 "Using input assistance" on page 1885
- « "Dialog 'Properties"' on page 2714
- ${ }^{\bullet}$ - Chapter 6.4.1.21.3.3.33 "Command 'Input Assistant"' on page 2574


## Command 'Go to Source Position'

Function: The command sets the cursor to the position in the source code that causes the message.
Call: Main menu "Edit", context menu of the message in the message view.
Requirements: A message is selected in the message view.
Use the command "Next Message" or "Previous Message" to display the source code position of the next or previous message.
See also

- $y^{\prime}>$ Chapter 6.4.1.21.3.3.35 "Command 'Next Message'" on page 2576
- $\Leftrightarrow$ Chapter 6.4.1.21.3.3.36 "Command 'Previous Message"" on page 2576
- $\Leftrightarrow$ Chapter 6.4.1.21.3.4.5 "Command 'Messages'" on page 2583


## Command 'Next Message'

Keyboard shortcut: [F4]
Function: This command selects the next message in the messages view.
Call: Main menu "Edit".
If the last message in the list has been reached, then the marking jumps to the beginning.
See also

- \# Chapter 6.4.1.21.3.3.36 "Command 'Previous Message"' on page 2576


## Command 'Previous Message'

Keyboard shortcut: [Shift] + [F4]
Function: This command selects the previous message in the messages view.
Call: Main menu "Edit"
If the first message in the list has been reached, then the marking jumps to the end.
See also

- © Chapter 6.4.1.21.3.3.35 "Command 'Next Message"' on page 2576


## Command 'Go to Definition'

Symbol: $\qquad$
Function: This command shows the definition locations of a variable or function.
Call: Main menu "Edit $\rightarrow$ Browse"
Requirement: A POU is open in the editor and the cursor is at a variable or function.
See also

- ※ Chapter 6.4.1.9.14.2 "Finding declarations" on page 1908


## Command 'Go To Reference'

Symbol:
Function: The command opens the declaration location of the variable that is referenced by the pointer currently in focus in online mode.
Call:

- Context menu in the declaration part or implementation code
- Menu bar: "Edit $\rightarrow$ Browse"

Requirement: Online mode. A POU is open in the editor and the cursor is at a pointer. The referenced variable is stored in static memory.


If the pointer does not point exactly to the beginning of the variable, then a corresponding message is displayed when you switch to the variable declaration.

See also

-     * Chapter 6.4.1.20.5.13 "Pointers" on page 2243


## Command 'Go to Instance'

Symbol: $\rightarrow$
Function: This command opens the instance of a function block in a new window.
Call: Menu bar: "Edit $\rightarrow$ Browse for Symbol"
Requirement: The application is in online mode. A POU is open in the editor and the cursor is at an instance of a function block.

The command is not available for temporary instances or instances from compiled libraries.
See also

- \& Chapter 6.4.1.9.14.2 "Finding declarations" on page 1908


## Command 'Refactoring' - 'Rename <...>'

Function: This command opens a dialog box for renaming an object or variable across the project.

Call: Main menu "Edit $\rightarrow$ Refactoring" or right-click.
Requirement: An object is selected in the device tree or in the "POUs" view, or the cursor is placed before or on a variable identifier in the declaration section of a programming object.

You can rename the following:

- Variables
- POUs
- GVLs
- Methods
- Properties
- Devices
- Variables and unit conversions in the unit conversion edit

Dialog box 'Re-
name'

| "Current name" | Name of the object or variable |
| :--- | :--- |
| "New name" | Input field for a new name. <br> If the name already exists, then CODESYS reports this directly below this input <br> field. |
| "OK" | Can be activated if you have typed a valid name in "New name". <br> Opens the "Refactoring" dialog box. |
|  | The affected objects and occurrences are highlighted in both views. <br> You can determine how to handle the occurrences in each view by right-clicking <br> the occurrences and clicking the available commands. |

## Dialog box 'Refactoring'

This dialog box displays all occurrences in the project.
The affected objects and occurrences are highlighted in both views.

| Right view | Displays the occurrence within an object where "Current name" occurs. |
| :--- | :--- |
| Left view | Device tree of the project with the object. |

You can determine how to handle the occurrences in each view by right-clicking the occurrences and clicking the available commands.

| "Reject this change" | Reject the single change in view on the right. |
| :--- | :--- |
| "Accept this object" | Accept all changes in the affected object. |
| "Reject this object" | Reject all changes in the affected object. |
| "Accept whole project" | Accept all changes in the project. |
| "Reject whole project" | Reject all changes in the project. |
| CODESYS highlights the accepted changes in yellow and the rejected changes in gray. |  |

See also

- $\Rightarrow$ Chapter 6.4.1.9.16 "Refactoring" on page 1910


## Command 'Refactoring' - 'Update Referenced Pins'

## NOTICE!

Currently, this command applies only to the CFC, FBD, LD, and IL editors. It is a combination of the "Reset Pins" and "Update Parameters" commands.

Function: This command modifies the pins according to the latest block declaration in all affected occurrences of the block.

Call: Main menu "Edit $\rightarrow$ Refactoring" or right-click.
Requirement: The cursor is placed in the name of the block in the first line of the block declaration or in the device tree.

See also

- $y^{*}$ Chapter 6.4.1.9.16 "Refactoring" on page 1910
- H Chapter 6.4.1.21.3.13.24 "Command 'Reset Pins'" on page 2694
- ${ }^{*}$, Chapter 6.4.1.21.3.14.38 "Command 'Update Parameters'" on page 2710


## Command 'Refactoring' - 'Add Variable'

## Symbol:

Function: This command enables the declaration of variables in a POU, as well as the automatic update to the occurrence of the POU.
Call: Main menu "Edit $\rightarrow$ Refactoring", or right-click.
Requirements: The declaration part is in focus.
The command opens the default dialog box for declaring variables.
See also

- ② "Dialog 'Auto Declare'" on page 2572

Dialog box 'Re- After clicking "OK" to close the declaration dialog, the "Refactoring" dialog box opens with two factoring' frames.


| "Right dialog frame"" | Declaration part and implementation of the POU where the variable is added. <br> Colored highlighting of changed locations: New added declarations have a blue <br> font and are highlighted in yellow (1). |
| :--- | :--- |
| "Left dialog frame" | Device tree or POUs tree of the project. <br> Colored highlighting of blocks where the POU is used: red font and yellow <br> highlight (2). <br> After you double-click the POU object, the detail view opens. |

Before you decide which changes to accept at which locations, select the required option from the drop-down list (3) at the upper right part of the window:

| "Add inputs with placeholder <br> text" | Default placeholder text:_REFACTOR_; editable <br> The placeholder text defined here is used at the occurrence locations of the new <br> added variables in the implementation code. This is used for searching for the <br> affected locations. |
| :--- | :--- |
| "Add inputs with the following |  |
| value" | Initialization value for the new variable. |

You can accept or reject changes by right-clicking the changed locations or by executing commands in the left or right area of the dialog box. Refer to the description of the "Refactoring $\rightarrow$ Rename" command.

```
Examples 1. By refactoring, the fun block receives a new input variable input3 with the initialization
        value 1. The change has the following effect:
Before:
fun(a + b, 3, TRUE);
fun(input1:= a + b , input2 :=3 , inputx := TRUE);
After:
fun(a + b, 3, 1, TRUE);
fun(input1:= a + b , input2 :=3 , _REFACTOR_, inputx := TRUE);
2. By refactoring, the "fun" block receives a new input variable input 3 with the placeholder text "_REFACTOR_":
```


## Before:

```
inst(input1 := a + b, input2 := 3, inputx := TRUE);
```

inst(input1 := a + b, input2 := 3, inputx := TRUE);
fun(a + b, 3, TRUE);
fun(a + b, 3, TRUE);
After:

```
```

inst(input1 := a + b, input2 := 3, input3 := _REFACTOR_, inputx

```
inst(input1 := a + b, input2 := 3, input3 := _REFACTOR_, inputx
:= TRUE);
:= TRUE);
fun(a + b, 3, _REFACTOR_, TRUE);
```

fun(a + b, 3, _REFACTOR_, TRUE);

```

\section*{See also}
- ※ Chapter 6.4.1.9.16 "Refactoring" on page 1910
- 公 Chapter 6.4.1.21.3.3.40 "Command 'Refactoring' - 'Rename <...>"' on page 2577

\section*{Command 'Refactoring' - 'Remove <variable>'}

Symbol: 厚
Function: This command removes an input or output variable from the POU and all occurrences of the POU.
Call: Main menu "Edit \(\rightarrow\) Refactoring", or right-click.
Requirements: In the declaration part of the POU, the cursor is located in the identifier of the variable to be removed.
Then, the command opens a dialog box with information about the removal. After you confirm this, the "Refactoring" dialog box opens. For a description of the "Refactoring" dialog box, refer to the "Edit \(\rightarrow\) Refactoring \(\rightarrow\) Rename" help page.
When you accept the changes in the "Refactoring" dialog box, the respective input and output parameters are deleted at the occurrence locations of the affected POU.


In CFC, only the connection is removed between the removed input or output to the block. The input or output itself remains in the chart.

Example in ST In a POU, refactoring removes the input4 input variable. The occurrences are updated automatically:
Before removal:
```

inst(input1 := a + b, input2 := 3, input4 := 1, input5 := TRUE);
fun(a + b, 3, 1, TRUE);

```

After removal:
```

inst(input1 := a + b, input2 := 3, input5 := TRUE);
fun(a + b, 3, TRUE);

```

See also
- \({ }^{*} \boldsymbol{y}\) Chapter 6.4.1.9.16 "Refactoring" on page 1910
- «y Chapter 6.4.1.21.3.3.40 "Command 'Refactoring' - 'Rename <...>"' on page 2577

\section*{Command 'Refactoring' - 'Reorder Variables'}

Function: This command allows changing the order of variables in the declaration editor for the selected scope: VAR_INPUT, VAR_OUTPUT, or VAR_IN_OUT.
Call: "Edit \(\rightarrow\) Refactoring"; context menu of the focused scope in the declaration editor.
Requirement: One of the above scopes is selected in the declaration, and more than one variable is declared in it.
The command opens the "Reorder" dialog box with a list of all declarations of the selected scope. You can drag a selected declaration up or down to another position.
See also
- 出 Chapter 6.4.1.9.16 "Refactoring" on page 1910

\section*{Command 'Advanced' - 'Format Document'}

Symbol:
Function: The command starts an automatic formatting of the code in the open ST editor.
Call: Menu bar: "Edit \(\rightarrow\) Advanced"; context menu of the window in focus in the ST editor
Requirement: The focus is in the ST editor. The syntax of the ST code does not contain any errors.

The following formatting is performed automatically:
- Keywords are converted to uppercase letters.
- Spacing is standardized.
- Indentations are changed according to syntax.
- Long lines are wrapped in sensible places.

See also
- \(\boldsymbol{y}^{*}\) Chapter 6.4.1.20.1.4.1 "ST Editor" on page 2049

\section*{Menu 'View'}
6.4.1.21.3.4.1 Standard Menu in View 'Devices', 'POUs', 'Modules' ..... 2582
6.4.1.21.3.4.2 Command 'Devices'. ..... 2582
6.4.1.21.3.4.3 Command 'POUs'. ..... 2583
6.4.1.21.3.4.4 Command 'Modules'. ..... 2583
6.4.1.21.3.4.5 Command 'Messages' ..... 2583
6.4.1.21.3.4.6 Command 'Element properties'. ..... 2584
6.4.1.21.3.4.7 Command 'ToolBox' ..... 2584
6.4.1.21.3.4.8 Command 'Watch' - 'Watch <n>' ..... 2584
6.4.1.21.3.4.9 Command 'Watch' - 'Watch All Forces'. ..... 2584
6.4.1.21.3.4.10 Command 'Add All Forces to Watchlist' ..... 2585
6.4.1.21.3.4.11 Command 'Bookmarks'. ..... 2585
6.4.1.21.3.4.12 Command 'Breakpoints' ..... 2586
6.4.1.21.3.4.13 Command 'Cross Reference List'. ..... 2587
6.4.1.21.3.4.14 Command 'Browse Cross References in Classic View' ..... 2589
6.4.1.21.3.4.15 Command 'Call Stack'. ..... 2590
6.4.1.21.3.4.16 Command 'Call tree'. ..... 2590
6.4.1.21.3.4.17 Command 'Memory'. ..... 2592
6.4.1.21.3.4.18 Command 'Security Screen'. ..... 2592
6.4.1.21.3.4.19 Command 'Settings of Memory Reserve for Online Change' ..... 2595
6.4.1.21.3.4.20 Command 'Start Page'. ..... 2596
6.4.1.21.3.4.21 Command 'Full Screen'. ..... 2597
6.4.1.21.3.4.22 Command 'Properties'. ..... 2597

\section*{Standard Menu in View 'Devices', 'POUs', 'Modules'}

The views "Devices", "POUs" and "Modules" provide the button \(\square\) in the top right corner to open a menu with the following commands:
- \(\quad\) "Open in editor": Opens the selected object in the corresponding editor.
- .Find object": Opens the dialog "Find Object" for the object tree. Starting to enter a search string all matching objects will be displayed with their path. Use the button "Open" to open the selected search result in the editor.
- "Sort by type": Sorts the objects in the view alphabetic by type.
- "Sort by name": Sorts the objects in the view alphabetic by name.
- "Sort ascending": Displays the chosen sorting in ascending order.
- "Sort descending": Displays the chosen sorting in descending order.
- "Track active editor": CODESYS selects the object, that is opened in the active editor, in the device tree of the view.

See also
- \(y^{*}\) Chapter 6.4.1.21.3.4.2 "Command 'Devices'" on page 2582
- \(\quad \xi^{*}\) Chapter 6.4.1.21.3.4.3 "Command 'POUs'" on page 2583
- «y Chapter 6.4.1.21.3.4.4 "Command 'Modules'" on page 2583

\section*{Command 'Devices'}

Symbol: view: [AIt] + [0]
Function: The command opens the view "Devices" in the CODESYS main window. The view contains the project's "device tree", where you configure your applications

Button opens the standard menu for navigating in the tree view.
Call: Menu "View"

See also
- \& Chapter 6.4.1.21.3.4.3 "Command 'POUs'" on page 2583
- © Chapter 6.4.1.21.3.4.1 "Standard Menu in View 'Devices', 'POUs', 'Modules"' on page 2582

\section*{Command 'POUs'}

Symbol: [], Shortcut: [Alt] + [1]
Function: This command opens the "POUs" view in the CODESYS main window. POUs located here are available in the entire project.
Call: Menu "View"
See also
- \& Chapter 6.4.1.21.3.4.1 "Standard Menu in View 'Devices', 'POUs', 'Modules"' on page 2582

\section*{Command 'Modules'}

Symbol:
Function: This command opens the "Modules" view and shows the modules of the application composer in a tree structure.
Call: Main menu "View"
See also
- * Chapter 6.4.1.21.3.4.1 "Standard Menu in View 'Devices', 'POUs', 'Modules"' on page 2582

\section*{Command 'Messages'}

Symbol:
Function: This command opens the "Messages" view.
Call: Menu bar: "View".

View 'Messages'
\begin{tabular}{|c|c|}
\hline Message category & The messages are categorized by component or functionality for selection from a drop-down list. Filter the message display by selecting a category. \\
\hline Message type & \begin{tabular}{l}
Click the symbol of the message type to show or hide messages. CODESYS displays the number of messages next to each symbol. \\
- ©: Error \\
- ©: Warning \\
- © : Message
\end{tabular} \\
\hline 区 & Deletes all messages in the selected message category. \\
\hline 圈 & Deletes all messages in all message categories. \\
\hline "Description" & \multirow{4}{*}{Message text with the reported object and the location in the object. Double-click a message in the table to jump to the source text location.} \\
\hline "Project" & \\
\hline "Object" & \\
\hline "Position" & \\
\hline
\end{tabular}

Table 474: "Commands in the context menu"
\begin{tabular}{|l|l|}
\hline "Next Message" & The source text position of the next message is displayed. \\
\hline "Previous Message" & The source text position of the previous message is displayed. \\
\hline "Go to Source Position" & The source position of the selected message is displayed. \\
\hline
\end{tabular}

\section*{Command 'Element properties'}

Symbol:
Function: This command opens the "Element Properties" view.
Call: Main menu "View"
This command opens the properties view for the open object. This view is available only for a few objects, for example visualization and POU (SFC).

The properties are displayed in a structured table. You change the property values by clicking into the value fields. You can filter or sort the properties view.

\section*{Command 'ToolBox'}

Symbol:
Function: This command opens the "ToolBox" view.
Call: Main menu "View"
This command opens the toolbox view for the open object. By default, this view is available for graphical editors and visualizations. It includes the graphical programming elements that you can drag into the editor.

\section*{Command 'Watch' - 'Watch <n>'}

Symbol: 展
Function: This command opens the "Watch <n>" view. You can populate a watchlist with variables from your project in order to monitor, force, or write these variable values in an individual view in online mode. The value " \(n\) " can be 1, 2, 3, or 4 for a total of up to four watchlists.

Call: Main menu "View"
See also
- ② Chapter 6.4.1.13.2.3 "Using watch lists" on page 2002

\section*{Command 'Watch' - 'Watch All Forces'}

Symbol: 原
Function: The command opens the "Watch All Forces" view, which is a special kind of watch list.

\section*{Call: Menu bar: "View \(\rightarrow\) Watch \(\rightarrow\) Watch All Forces"}

Requirement: A project is in offline mode or online mode.
The view contains all variables currently prepared for forcing, and all forced variables of the application in one list. Actions are possible in the list which are also possible in other watch lists. Moreover, the following commands are available in the "Unforce" list box of the view:

Table 475: "Watch All Forces"
Tabular view of all forced variables of the application, including variables prepared for forcing
\begin{tabular}{|l|l|}
\hline "Expression" & Variable name \\
\hline "Data Type" & Data type of the variable \\
\hline "Value" & Currently forced value of the variable \\
\hline "Prepared Value" & Value prepared for forcing \\
\hline \begin{tabular}{l} 
"Overwritten value at start of \\
cycle"
\end{tabular} & \begin{tabular}{l} 
For inputs, the actual value is already overwritten by the force value before the \\
application code is executed. As a result, this is the forced value. \\
For outputs, this is the forced value.
\end{tabular} \\
\hline \begin{tabular}{l} 
"Overwritten value at end of \\
cycle"
\end{tabular} & \begin{tabular}{l} 
For outputs, this is the value which is calculated in the cycle. However, this value \\
is overwritten by the force value at the end of the cycle. \\
For inputs, this is the forced value.
\end{tabular} \\
\hline
\end{tabular}
- "Unforce and Keep All Selected Values": For all selected entries in the list, the variables will be set to the forced value and the forcing will be lifted
- "Unforce and Restore All Selected Values": For all selected entries in the list, the variables will be reset to the values they had before they were forced, and the forcing will be lifted.

See also
- \# Chapter 6.4.1.12.5 "Forcing and Writing of Variables" on page 1987
- 出 Chapter 6.4.1.13.2.3 "Using watch lists" on page 2002

\section*{Command 'Add All Forces to Watchlist'}

Function: The command adds all variables of the active application, which are currently prepared for forcing, or which are already forced, to the watchlist. Please regard, that this works only for docked watch list views.

Call: Context menu of view "Watch"
Requirement: Online mode, a watch list view is active.


There is a special watch list: "Watch All Forces". This view shows automatically all variables currently prepared for forcing or already being forced. It provides additional commands for releasing any forces.

See also
- ⓨ Chapter 6.4.1.13.2.3 "Using watch lists" on page 2002
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.4.8 "Command 'Watch' - 'Watch <n>'" on page 2584
- \(\boldsymbol{«}^{*}\) Chapter 6.4.1.21.3.4.9 "Command 'Watch' - 'Watch All Forces'" on page 2584
- \& Chapter 6.4.1.12.5 "Forcing and Writing of Variables" on page 1987

\section*{Command 'Bookmarks'}

Symbol: 気
Function: This command opens the "Bookmarks" view.
Call: Menu bar: "View".
\begin{tabular}{|c|c|}
\hline *"Previous Bookmark" & Jumps to the bookmark that above the selected bookmark in the table and opens the respective POU in the editor. \\
\hline 7/ "Next Bookmark" & Jumps to the bookmark that below the selected bookmark in the table and opens the respective POU in the editor. \\
\hline \(\times\) & Deletes the selected bookmark from the table and in the respective POU. \\
\hline \multicolumn{2}{|l|}{\begin{tabular}{l}
List of bookmarks in the project with the following information: "Bookmark", "Object", and "Position". \\
You can edit the bookmark order per drag\&drop. \\
When you double-click a row, CODESYS opens the respective "Object" in the editor and jumps to this bookmark.
\end{tabular}} \\
\hline "Bookmark" & \begin{tabular}{l}
Name of the bookmark as assigned by CODESYS in ascending numerical order: "Bookmark_0", "Bookmark_2" etc. \\
If the bookmark is selected and you click in the field, then it is editable and you can modify the bookmark name.
\end{tabular} \\
\hline "Object" & \begin{tabular}{l}
Name and project path of the POU where the bookmark is set \\
Example: POU_Add [PLC_1: SPS-Logic: Application]
\end{tabular} \\
\hline "Position" & \begin{tabular}{l}
Position of the bookmark in the POU \\
Example: Row 3, Column 1 (Impl) \\
(Impl) : in the implementation part of the POU \\
(Decl) : in the declaration part of the POU
\end{tabular} \\
\hline
\end{tabular}

See also
- ⓨ Chapter 6.4.1.9.14.3 "Setting and using bookmarks" on page 1908
- \({ }^{4}\) Chapter 6.4.1.21.3.3.24 "Command 'Next Bookmark'" on page 2570
- \(\quad{ }^{\circ}\) Chapter 6.4.1.21.3.3.26 "Command 'Previous Bookmark'" on page 2570

\section*{Command 'Breakpoints'}

Symbol: 四
Function: This command opens the "Breakpoints" view.
Call: Menu bar: "View".
This view shows an overview of all defined breakpoints for an application. You have access to all breakpoint commands within this view.

Table 476: Table of current breakpoints
\begin{tabular}{|l|l|}
\hline "Application" & Select the required application from the list. \\
\hline "POU" & Name of the function block that will receive the breakpoint \\
\hline "Location" & \begin{tabular}{l} 
Location of the breakpoint in the POU \\
\(\bullet \quad\) Text editor: Line number and column number \\
\(\bullet \quad\) Graphical editor: Network number or element number \\
For function blocks, "(Impl)" indicates that the breakpoint is located in the imple- \\
mentation of the function block, not in an instance.
\end{tabular} \\
\hline "Instance Path" & Complete object path of the breakpoint location. \\
\hline "Tasks" & \begin{tabular}{l} 
Names of tasks that will be effective when the breakpoint is executed. If there \\
are no restrictions, then "(all)" is displayed here.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline ＂Condition＂ & \begin{tabular}{l}
－＂Break always＂：No additional enable condition defined；the breakpoint is \\
always enabled． \\
Boolean expression．The expression must yield TRUE for the breakpoint to \\
be enabled．
\end{tabular} \\
\hline ＂Hit Count Condition＂ & When the breakpoint should be in effect（depending on the hit count） \\
\hline ＂Current Hit Count＂ & \begin{tabular}{l} 
How often the breakpoint has already been reached up to now during the execu－ \\
tion
\end{tabular} \\
\hline
\end{tabular}

Table 477：Toolbar
\begin{tabular}{|c|c|c|}
\hline \multirow[t]{2}{*}{景} & ＂New Breakpoint＂ & This command opens the＂Breakpoint Properties＂dialog． \\
\hline & －＂New Data Breakpoint＂ & This command opens the＂New Breakpoint＂dialog． \\
\hline \(X\) & ＂Clear Breakpoint＂ & Removes the breakpoint（not the same as disable） \\
\hline 煓 & ＂Enable／Disable Breakpoint＂ & \begin{tabular}{l}
Toggles the status of the breakpoint or execution point between＂enabled＂and ＂disabled＂ \\
－Breakpoint enabled \\
－O Breakpoint disabled \\
－Execution point enabled \\
－Execution point disabled \\
－Data breakpoint enabled \\
－Data breakpoint disabled \\
－Data execution point enabled \\
－Data execution point disabled \\
As opposed to＂Clear breakpoint＂，a disabled breakpoint remains in the list and can be enabled again．
\end{tabular} \\
\hline 嘓 & ＂Properties＂ & The＂Breakpoint Properties＂dialog opens for editing the breakpoint parameters． This dialog is the same as＂New Breakpoint＂．In online mode，you can change the breakpoint into an execution point． \\
\hline 里 & ＂Go to Source Position＂ & Opens the online view of the affected block．The cursor is set at the breakpoint location． \\
\hline 8930］ & ＂Clear All Breakpoints＂ & Deletes all breakpoints and execution points in the application．The list is cleared． Not to be confused with＂deactivate＂． \\
\hline 阶而 & ＂Enable All Breakpoints＂ & Enables all currently disabled breakpoints and execution points． \\
\hline 所的 & ＂Disable All Breakpoints＂ & Disables all currently enabled breakpoints and execution points．The points remain in the list and can be enabled again． \\
\hline
\end{tabular}

See also
－\＆Chapter 6．4．1．21．4．6＂Dialog＇Breakpoint Properties＇＂on page 2747
－« \({ }^{*}\) Chapter 6．4．1．21．3．8．4＂Command＇New Breakpoint＂＇on page 2645
－\＆Chapter 6．4．1．21．3．8．5＂Command＇New Data Breakpoint＂＇on page 2645
－\({ }^{*} \boldsymbol{y}\) Chapter 6．4．1．21．3．8．7＂Command＇Enable Breakpoint＇＂on page 2646
－\({ }^{*} \Rightarrow\) Chapter 6．4．1．21．3．8．8＂Command＇Disable Breakpoint＂＇on page 2646
－« Chapter 6．4．1．21．3．8．9＂Command＇Toggle Breakpoint＂＇on page 2646

\section*{Command＇Cross Reference List＇}

Symbol：
Function：This command opens the＂Cross Reference List＂view．

\section*{Call: Menu bar: "View", or "Edit \(\boldsymbol{\rightarrow}\) Browse \(\boldsymbol{\rightarrow}\) Browse Cross References".}

This view shows a list of cross-references for a symbol in the project. The symbol can be a variable, a POU (program, function block, function), or a user-specific data type (DUT). The cross-reference list offer two basic types of searches:
- Text search: By specifying a symbol name, the cross-references of all symbols in the project are displayed with their names. If multiple symbols with the same are found, then the display can be limited to individual declarations by means of the context menu.
- Declaration search: The symbol can be selected by means of the input assistant or by specifying a qualified path (for example, Device.Application.PLC_PRG.i or POOL.POU.a). Then only the occurrence locations of this symbol are displayed, even if there exist other symbols with the same name.
\begin{tabular}{|c|c|}
\hline Input field & \begin{tabular}{l}
Symbol name (variable name, POU name, DUT name). Input options: \\
- Selection of a declared symbol by means of the input assistant ( \(Q\) button). Manual input of the symbol name. Triggering of the search by pressing the \(\Rightarrow\) button or the [Enter] key. \\
For the text search, you can use the placeholders " "*" (any number of characters) or "?" (exactly any one character) in combination with a partial string of a variable identifier. \\
Use the percent sign "ஃ" to search for IEC addresses. Examples: " \(\% \mathrm{MW} 8\) ", "ьм" \\
More options outside of cross-reference list view: \\
- Use the command "Browse for Symbol \(\rightarrow\) Browse Cross References" if the name of a declared symbol is selected in an editor, or if the cursor is in the name field. A search is also possible if the object is selected in the device tree or POU pool. \\
- Automatic if the name of a declared symbol is selected in an editor, or if the cursor is in the name field. A automatic search is also possible if the object is selected in the device tree or POU pool. \\
Requirement: CODESYS option "Automatically list selection in cross reference view" is activated (category "SmartCoding". \\
The following input is valid: \\
- Variable name, simple or qualified. Examples: "iVar", "PLC_PRG.iVar". \\
- POU name: Examples: "PLC_PRG", "myFB". \\
- DUT name: Example: "mystruct". \\
- Strings combined with placeholders: asterisk (*) for any character or question mark (?) for exactly one characters). \\
Example: "iVar*" applies to iVar1, iVar_glob2, iVar45, etc. \\
"iVar?" refers to iVar1, iVar2, iVarx, and so on, but not \\
iVar_glob2, iVar45 and so on... \\
- "\%<IEC address>": CODESYS searches for variables that are assigned to this address and direct memory access. Example: " \(\circ Q \mathrm{QB} 0\) ", \(\%\) \(\Omega 0:=2\).
\end{tabular} \\
\hline Q & Open input assistant for selecting a symbol \\
\hline \(\Rightarrow\) & Perform a search \\
\hline 8 & Define columns to search for the string. \\
\hline Input field & String that is searched for in the selected columns. The result locations are marked in yellow. Cross references without this string are hidden. \\
\hline - & Show source position of previous cross-reference, [Shift]+[F4] \\
\hline 4 & Show source position of next cross-reference, [F4] \\
\hline E- & Limit results to current declaration: Available if multiple declarations are found for a symbol. Limits the display to the declaration that you have selected in the list. \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline \(\pm\) & Show source position of selected cross-reference: The focus jumps to the occurrence location of the symbol. \\
\hline 즐 & Print cross-reference list: The default dialog opens for setting up a print job. \\
\hline \multicolumn{2}{|l|}{The cross references are displayed with the following information:} \\
\hline "Symbol" & \begin{tabular}{l}
The result locations for the symbols (variables, POUs, or DUTs) are grouped by declaration. The declaration occurrence comprises the root node and the occurrence locations in the project are indented below. The precise expression is displayed that has the symbol at the occurrence location. \\
Example: If there is a global variable \(i\) in the project and a local declared variable \(i\) in a POU, then two root node entries will be listed after a text search for cross-references with the occurrences of the variable i below each.
\end{tabular} \\
\hline "POU" & Block name; also a task name if a block call in the task configuration. \\
\hline "Variable" & Only the variable name (for example, iVar) \\
\hline "Access" & \begin{tabular}{l}
Type of access to the variable at the occurrence location: "Declaration"/ "Read"/ "Write"/ "Call". \\
Special case for pointers: An assignment type p := ADR (var1) is displayed as write | address when searching for var1. The reason for this: Any write access to p is not displayed when searching for var1. Write access is also possible by means of pointer variables.
\end{tabular} \\
\hline "Type" & Data type of the variable \\
\hline "Address" & IEC address if variables are assigned Example: "AT \%QB0". \\
\hline "Position" & Location of the occurrence in the POU editor, for example line number, network number, declaration part, or implementation part. Example: "line 1, column 1 (Impl)". \\
\hline "Object" & POU name plus complete path of the occurrence location in brackets (if this is found in the "Devices" view). Example: "PLC_PRG [Device:Plc Logic:Application]" \\
\hline "Comment" & Comments if available in the declaration of the variable \\
\hline
\end{tabular}

The search yields all result locations in the project and in included, uncompiled libraries.

Right-click com- "Show source position": Opens the respective POU and marks the occurrence: for root entries, mands in the cross-reference list the declaration, and for subordinate entries, the respective occurrence location. As an alternative, you can double-click a line.
"Limit Results to Selected Declaration": Limits the display of results to the selected symbol declaration if multiple declarations are found.
"Expand All": In the list, every single result location is shown.
"Collapse All": In the list, only the root nodes of the result locations are shown.

See also
- \({ }^{*}\), Chapter 6.4.1.21.3.3.29 "Command 'Browse Cross References'" on page 2571
- \({ }^{2} \gg\) Chapter 6.4.1.9.14.1 "Using the cross-reference list to find occurrences" on page 1906
- \({ }^{\star}\) Chapter 6.4.1.21.3.23.3 "Command 'Limit Results to Current Declaration'" on page 2744
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.3.18 "Command 'Collapse All Folds'" on page 2568
- Ey Chapter 6.4.1.21.3.3.17 "Command 'Expand All Folds'" on page 2568

\section*{Command 'Browse Cross References in Classic View'}

Symbol

Function: This command opens the "Classic Cross Reference List" view.
Call: The command is not in any menu by default. You can add it to a menu by using the dialog box from "Tools \(\rightarrow\) Customize" (command category "Browse Project").

The view corresponds to the "Cross Reference List" view before CODESYS V3.5 SP6.

\section*{Command 'Call Stack'}

\section*{Symbol: 路}

Function: This command opens the "Call Stack" view.
Call: Main menu "Debug".
This view is very useful when you want to step into programs. It shows the current location with the complete call path.
\begin{tabular}{|l|l|}
\hline "Application" & Name of the active application that controls the current POU \\
\hline "Task" & Name of the task that controls the current POU \\
\hline \multicolumn{1}{|l|}{} & \begin{tabular}{l} 
Name of the POU where program execution has halted \\
The first line in the list describes the current execution location (marked with a \\
yellow arrow). If this location is in a block that is called by another block, then \\
the call location is described in the second line. In turn, if the caller is called by \\
yet another block, then that call location is described in the third line, and so \\
on.
\end{tabular} \\
\hline "Location" & \begin{tabular}{l} 
Position within the POU where program execution has halted \\
\(\bullet\) \\
\(\bullet\) \\
- Line and column numbers for textual editors
\end{tabular} \\
\hline "Instance & Instance where program execution has halted \\
path"
\end{tabular}

The call stack is also available in offline mode and normal online mode when you are not currently using any debugging functions). In this case, it receives the last displayed location during a stepped execution, but it is displayed in gray.

The "Call Tree" view, in contrast to the "Call Stack", at any time provides information on the calls of a POU.

\section*{See also}
- \({ }^{4}\) Chapter 6.4.1.12.3 "Using Breakpoints" on page 1981
(2) Chapter 6.4.1.21.3.4.16 "Command 'Call tree'" on page 2590

\section*{Command 'Call tree'}

Symbol: \(=5\)
Function: This command opens the "Call Tree" view.
Call:
- "View" menu
- Context menu of a callable block in the "Devices" or "POUs" view.

View 'Call tree' The call tree is available at all times, even before compiling the application. It is a static representation of the caller and the calls of the block that you specify explicitly. Therefore, the tree always contains two root nodes above the respective call order is displayed as successive indented entries. Recursive calls are detected quickly in this tree representation.

Example of a call tree (1) for the (2) PLC_PRG block:

- (3) Node "<block name> is called by:"
- (4) Node "<block name> calls:"
\begin{tabular}{|c|c|}
\hline "Block name" & \begin{tabular}{l}
Name of the program block; specified manually, by dragging from another view, or by means of the button \\
The drop-down list includes the last specified block names.
\end{tabular} \\
\hline \multicolumn{2}{|l|}{Toolbar and keyboard usage} \\
\hline ¢ : Find block \(^{\text {a }}\) & CODESYS searches for the block specified in "Block name" and displays its caller and calls. \\
\hline B: Use block from the input assistant & The "Input Assistant" dialog box opens for selecting a block call or instance call. The call tree is refreshed automatically after the selection. \\
\hline Show source code position of the selected block & CODESYS jumps to the occurrence location of the block in the source code of the program. \\
\hline \begin{tabular}{l}
[F4]: Show source code position of the next block \\
[Shift]+[F4]: Show source code position of the previous block
\end{tabular} & \begin{tabular}{l}
The selection in the call tree jumps to the next or previous block in the call structure. At the same time, the associated source code position is opened in the respective editor. \\
Note: Double-clicking an entry in the call tree also opens the associated source code position.
\end{tabular} \\
\hline \multicolumn{2}{|l|}{Display of the call tree:} \\
\hline "Symbol" & \begin{tabular}{l}
"<block name> is called by": The call order is displayed for below this node. The bottom entry in this tree structure shows the start of the calls. \\
"<block name> calls": The calls from this block are displayed below this node. The bottom entry in this tree structure shows the end of the call chain.
\end{tabular} \\
\hline "Position" & \begin{tabular}{l}
For the root node in the call tree: Line numbers of the declaration ("Decl") of the block. \\
For the caller or calls below the root node: Line number, column number, and network number of the position, depending on the implementation language.
\end{tabular} \\
\hline \multicolumn{2}{|l|}{Context menu for the entry selected in the tree:} \\
\hline "Collapse All" & The expanded entries in the call tree are collapsed, except for the two root nodes. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Show Source Position" & \begin{tabular}{l} 
CODESYS jumps to the occurrence location of the block in the source code of \\
the program.
\end{tabular} \\
\hline "Set as New Root Node" & \begin{tabular}{l} 
The entry selected in the call tree is displayed in "Block name". The tree is \\
refreshed automatically for the new root nodes.
\end{tabular} \\
\hline
\end{tabular}

The "Call list" view is provided for immediate information when stepping through a program, as opposed to the static call tree that provides call information about a block. The call list always shows the full call path of the current position that is reached.

See also
- \({ }^{\wedge}>\) Chapter 6.4.1.12.4 "Stepping Through a Program" on page 1985
- \({ }^{4}\) Chapter 6.4.1.21.3.4.15 "Command 'Call Stack"' on page 2590
- \(\quad \Leftrightarrow\) Chapter 6.4.1.21.3.3.31 "Command 'Browse Call Tree"' on page 2572

\section*{Command 'Memory'}

Symbol:
Function: In CODESYS V3.5 version earlier than SP11, the command opens the "Memory" view.
Call: Menu bar: "View \(\rightarrow\) Memory".
As of SP11, the command provides the notice that you must install the CODESYS Memory Tools package (available in the CODESYS Store) in order to use the memory view. After installation, you can open the "Memory" view by clicking "View \(\rightarrow\) Show Memory View".

\section*{Command 'Security Screen'}

Symbol:
Function: The command opens the "Security Screen" view.
Call:
- "View" menu
- (3) icon or 3 in the status bar

The icon is displayed in blue when a valid certificate is specified for the digital signature. When only one client certificate is specified for the encrypted communication, the icon remains gray, resulting in the client certificate providing no increased security for the user.

The following security features of CODESYS are configured and displayed in the view:
- Personal user certificate
- Encrypted communication
- Encryption and signatures of IEC projects
- Encryption and signature of download, online change, and boot application
- Security level

\section*{NOTICE!}

When the "Security Screen" is opened and closed, the current settings are applied in the user options, even when no active changes have been made.

If the CODESYS Security Agent add-on product is installed, then the "Security Screen" view provides an additional "Devices" tab. This allows for the configuration of certificates for the encrypted communication with controllers.

\begin{abstract}
Tab 'User' On this tab, certificates are configured that are required for the encrypted communication and the digital signature of the user. Only certificates with private keys can be specified here. The user profile is saved as an XML file in the user options.
\end{abstract}
\begin{tabular}{|c|c|}
\hline "User Profile and Certificate Selection" & By default, the login name for Windows is specified as the user profile. \\
\hline List box with existing user profiles & \begin{tabular}{l}
Opens the "User Profiles" dialog. Here you specify the name for a new user profile. \\
Deletes the selected user profile. This user profile is no longer displayed in the list box.
\end{tabular} \\
\hline "Digital Signature" & \begin{tabular}{l}
Opens the "Certificate Selection" dialog for selecting the certificate for the digital signature. \\
One certificate can be selected. The certificate has to have a private key. \\
Deletes the displayed certificate. \\
One certificate can be selected. The certificate has to have a private key.
\end{tabular} \\
\hline "Project File Decryption" & \begin{tabular}{l}
Opens the "Certificate Selection" dialog for selecting the certificate for decrypting project files. \\
One certificate can be selected. The certificate has to have a private key. \\
Deletes the displayed certificate.
\end{tabular} \\
\hline
\end{tabular}

See also
- \#y Chapter 6.4.1.21.4.19 "Dialog 'Certificate Selection'" on page 2811

Table 478: "Security Level"
\begin{tabular}{|l|l|}
\hline "Activate the Use of Certificates for Enhanced Security" \\
\hline \begin{tabular}{l|l} 
"Enforce encrypted \\
communication"
\end{tabular} & \begin{tabular}{l} 
V: When the user communicates with the controller, the server certificate of \\
the controller is used for establishing an encrypted connection. Then the entire \\
communication is encrypted.
\end{tabular} \\
\hline "Enforce encryption of project \\
files" & \begin{tabular}{l} 
V: All project files of the user are encrypted with a certificate. When the project \\
is saved, it is encrypted with the certificate specified in the project settings \\
("Project Settings \(\rightarrow\) Security" dialog). The selected certificate is displayed on \\
the "Project" tab in the "Project file encryption" group.
\end{tabular} \\
& \begin{tabular}{l} 
To open this project, the certificate to be encrypted has to be specified in "Project \\
file decryption" with a private key.
\end{tabular} \\
\hline "Enforce signing of project & \begin{tabular}{l}
\(\checkmark:\) All project files of the user are signed with a certificate. In "Digital Signature", \\
a certificate has to be specified with a private key. \\
When a project is saved, a signature file <project name>. project. p7s is \\
generated in the project directory containing the signature.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Enforce encryption of downloads, online changes and boot applications" & \begin{tabular}{l}
The data that is downloaded to the controller has to be encrypted with a controller certificate. \\
This certificate is defined directly either in the properties dialog of the application on the "Encryption" tab, or in the security screen, on the "Project" tab, in the "Encryption of Boot Application, Download and Online Change" group. \\
Controller certificates are located in the local Windows Certificate Store in the "PLC Certificates" directory. If the certificates of your controller are not available in the directory, then they first have to be loaded from the controller and installed to the directory. For instructions, see the ""Controller Certificates"" chapter.
\end{tabular} \\
\hline "Enforce signing of downloads, online changes and boot applications" & \begin{tabular}{l}
The online code (downloads, online changes, and boot applications) have to be signed with a certificate with a personal key. The certificate is selected from the "Digital Signature" area. \\
Requirement: The "Encryption of boot application, download and online change" option is selected.
\end{tabular} \\
\hline "Enforce signing of compiled libraries" & \begin{tabular}{l}
The "File \(\rightarrow\) Save Project as Compiled Library" command generates a signed library <library name>. compiled-library-v3. \\
Requirements \\
- A certificate with a private key that supports code signing is available. \\
- A library compatibility >= CODESYS V3 SP15 is set in the project information. \\
"Enforce timestamping of signed compiled libraries": \(\square\) : The URL of the time stamp server which created the time stamp has to be entered in the \\
"Timestamping server" field. Example: timestamp.comodoca. com/rfc3161.
\end{tabular} \\
\hline \multicolumn{2}{|r|}{\begin{tabular}{l}
See also \\
- \({ }^{4}\) Chapter 6.4.1.16 "Using the Command-Line Interface" on page 2028 \\
- ② Chapter 6.4.1.21.3.2.7 "Command 'Save Project as Compiled Library'" on page 2556 \\
- \(\boldsymbol{y y}^{\boldsymbol{y}}\) Chapter 6.4.1.17.3 "Information for Library Developers" on page 2035
\end{tabular}} \\
\hline
\end{tabular}

Tab 'Project' All project-specific settings are configured on this tab. These elements are active only when a primary project is loaded.
\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|l|}{"Project file encryption"} \\
\hline "Technology" & \begin{tabular}{l}
Opens the "Project Settings \(\rightarrow\) Security" dialog \\
When you select the "Encryption" project setting and then "Certificates" in the dialog, you can choose a corresponding certificate by clicking \(\square\) For more information, see the description of the "Project Settings: Security" dialog.
\end{tabular} \\
\hline "Certificates of Users Sharing this Project" & Area for listing the certificates that encrypt the project file. \\
\hline
\end{tabular}
"Encryption of Boot Application, Download and Online Change"

List of the applications of the controller

Double-clicking an application in the list opens the "Properties \(\rightarrow\) Encryption" dialog. Depending on the settings of the "Security Level" on the "User" tab of the "Security Screen", the following fields are available in the open properties dialog:
- "Encryption" tab with active "Certificates" area
- "Encryption" tab with "Encryption Technology" list box.

In the "Properties \(\rightarrow\) Encryption" dialog, click the button to select the controller certificate for "Encryption of Boot Application, Download and Online Change". For more information, see the description of the "Properties: Encryption" dialog.
Controller certificates are located in the local Windows Certificate Store in the "PLC Certificates" directory. If the certificates of your controller are not available in the directory, then they first have to be loaded from the controller and installed to the directory. For instructions, see the "Protecting and Saving a Project" - " Encryption with Certificates" chapter.

See also
- \(\Leftrightarrow\) Chapter 6.4.1.21.4.12.8 "Dialog 'Project Settings' - 'Security"' on page 2772
- \(\Rightarrow\) Chapter 6.4.1.21.4.11.4 "Dialog: Properties: Security" on page 2754
- \(\quad\) Chapter 6.4.1.6.8 "Encrypting Projects with Certificates" on page 1829

Tab 'Devices'
This tab is available only after you have installed the CODESYS Security Agent add-on. For a description of this tab, see the help for the CODESYS Security Agent.

\section*{Command 'Settings of Memory Reserve for Online Change'}

Function: This command opens the "Online Change Memory Reserve" view.
Call: Menu bar: "View".
In the view, memory reserves are configured for the function blocks during the online change.
\begin{tabular}{|l|l|}
\hline "Scan Application" & \begin{tabular}{l} 
- Searches the selected application for function blocks and displays them in \\
the "Function blocks" area
\end{tabular} \\
& \begin{tabular}{l} 
- Updates the "Function blocks" area after the application is built again. \\
- Updates the "Function blocks" area after an online change.
\end{tabular} \\
\hline \begin{tabular}{l} 
Drop-down list with the applica- \\
tions of the open project
\end{tabular} & \begin{tabular}{l} 
Selection of the application whose function blocks should be displayed and/or \\
edited in this view.
\end{tabular} \\
\hline
\end{tabular}

Table 479: "Function Blocks"
\begin{tabular}{|l|l|}
\hline "All" & All function blocks of the selected application are displayed. \\
\hline "Pool" & \begin{tabular}{l} 
All function blocks of the "POUs" view that are displayed which are referenced in \\
the application.
\end{tabular} \\
\hline "No memory-reserve" & All function blocks with a memory reserve of 0 bytes are displayed. \\
\hline "<memory reserve> bytes" & \begin{tabular}{l} 
Display of all function blocks with the number of bytes is displayed that is defined \\
in "Memory reserve".
\end{tabular} \\
\hline \begin{tabular}{l} 
Information about the function blocks
\end{tabular} \\
Multiple selection is also possible when selecting a POU for the configuration of the memory reserve.
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Function block" & Name of the function block \\
\hline "Size" & \begin{tabular}{l} 
Size of the function block \\
Size of an instance of a function block \\
Specified in bytes
\end{tabular} \\
\hline "Number of instances" & Number of instances of a function block in the project \\
\hline "Memory reserve" & Display of the memory reserve for each instance of the function block \\
\hline \begin{tabular}{l} 
"Additional memory for all \\
instances"
\end{tabular} & Product of "Number of instances" and "Memory reserve" \\
\hline "Remaining memory reserve" & Number of bytes that are available as reserve. \\
\hline
\end{tabular}

Table 480: "Settings"
\begin{tabular}{|l|l|}
\hline "Memory reserve (in bytes)" & \begin{tabular}{l} 
Input field for the memory reserve for the selected function block. \\
Specified in bytes \\
Requirement: the application is not located on the controller yet or you have \\
allowed the memory reserve to be changed by clicking the "Edit" button in the \\
"Allow editing" area.
\end{tabular} \\
\hline "Apply for Selection" & \begin{tabular}{l} 
The "Memory reserve (in bytes)" is assigned to the function block and the table \\
column "Memory Reserve" is updated. \\
In multiple selection, the specified value is assigned to each function block. \\
In order to update the columns "Size", "Number of Instances", "Additional \\
Memory for All Instances", and "Remaining Size of the Memory Reserve", click \\
"Build \(\boldsymbol{\rightarrow}\) Build", and then click the "Scan Application" button.
\end{tabular} \\
\hline
\end{tabular}

Table 481: "Enable Editing"
\begin{tabular}{|l|l}
\hline "Enable" & \begin{tabular}{l} 
The input field "Memory reserve (in bytes)" is editable. \\
This button is modified in "Editable".
\end{tabular} \\
\hline
\end{tabular}

Table 482: "Information"
\begin{tabular}{|l|l|}
\hline "Number of FBs" & Total number of function blocks in the application \\
\hline \begin{tabular}{l} 
"Additional memory for all \\
instances"
\end{tabular} & \begin{tabular}{l} 
Sum of the memory reserves of all function block instances of the application. \\
Specified in bytes
\end{tabular} \\
\hline
\end{tabular}

See also
- * Chapter 6.4.1.21.3.7.6 "Command 'Online Change'" on page 2629

\section*{Command 'Start Page'}

Symbol: 包
Function: This command opens the "Start Page" view.
Call: Main menu "View"
The view includes some basic commands and a list of recently opened projects. In addition, the CODESYS homepage is displayed.

If you access the Internet through a proxy, then you can save the authentication data in the project options ("Proxy Settings") so you do not have to provide this data every time you use this command.

By moving the mouse pointer over the list of recently opened projects, you can remove or pin individual projects in the list. Pinned projects remain in this list until you remove the pin.

In the project options ("Load and Save"), you can configure whether this start page should open automatically when you start CODESYS.

See also
- \(\quad>\) Chapter 6.4.1.21.4.14.17 "Dialog 'Options' - 'Load and Save'" on page 2791
- \(\Rightarrow\) Chapter 6.4.1.21.4.14.21 "Dialog 'Options' - 'Proxy Settings"" on page 2793

\section*{Command 'Full Screen'}

Symbol: 回, keyboard shortcut [Ctrl]+[Shift]+[F12]
Function: This command switches the CODESYS display to full screen mode.
Call: Main menu "View"
Choosing this command displays the main window of the CODESYS user interface in fullscreen mode. You can return to the previous setting by choosing the command again or with the keyboard shortcut [Ctrl]+[Shift]+[F12].

\section*{Command 'Properties'}

Symbol: 啗
Function: This command opens the properties of the currently selected object in the POUs tree or device tree.

Call: Main menu "View"

\section*{Menu 'Project'}
6.4.1.21.3.5.1 Command 'Add Object'. ..... 2598
6.4.1.21.3.5.2 Command 'Add Folder'. ..... 2599
6.4.1.21.3.5.3 Command 'Insert Device'. ..... 2599
6.4.1.21.3.5.4 Command 'Plug Device'. ..... 2600
6.4.1.21.3.5.5 Command 'Scan for Devices'. ..... 2600
6.4.1.21.3.5.6 Command 'Update Device' ..... 2602
6.4.1.21.3.5.7 Command 'Acknowledge Diagnosis', 'Acknowledge Diagnosis for Subtree' ..... 2602
6.4.1.21.3.5.8 Command 'Edit Object'. ..... 2603
6.4.1.21.3.5.9 Command 'Edit Object with'. ..... 2603
6.4.1.21.3.5.10 Command 'Check integrity' ..... 2603
6.4.1.21.3.5.11 Command 'Edit Object (Offline)' ..... 2603
6.4.1.21.3.5.12 Command 'Set Active Application'. ..... 2603
6.4.1.21.3.5.13 Command 'Project information'. ..... 2604
6.4.1.21.3.5.14 Command 'Project Settings'. ..... 2604
6.4.1.21.3.5.15 Command 'Project Environment'. ..... 2604
6.4.1.21.3.5.16 Command 'Project Localization' - 'Create Localization Tem- plate'. ..... 2604
6.4.1.21.3.5.17 Command 'Project Localization' - 'Manage Localizations'. ..... 2605
6.4.1.21.3.5.18 Command 'Project Localization' - 'Toggle Localization' ..... 2606
6.4.1.21.3.5.19 Command 'Document' ..... 2606
6.4.1.21.3.5.20 Command 'Compare objects'. ..... 2607
6.4.1.21.3.5.21 Command 'Compare'... ..... 2607
6.4.1.21.3.5.22 Command 'Commit Accepted Changes'. ..... 2611
6.4.1.21.3.5.23 Command 'Map pool devices'. ..... 2611
6.4.1.21.3.5.24 Command 'Export'. ..... 2611
6.4.1.21.3.5.25 Command 'Import'. ..... 2612
6.4.1.21.3.5.26 Command 'Export PLCopenXML' ..... 2612
6.4.1.21.3.5.27 Command 'Import PLCopenXML'. ..... 2612
6.4.1.21.3.5.28 Command 'User management' - 'Log in User'. ..... 2613
6.4.1.21.3.5.29 Command 'User management' - 'Log out User'. ..... 2613
6.4.1.21.3.5.30 Command 'User management' - 'Rights.. ..... 2613
6.4.1.21.3.5.31 Command 'Insert Device'. ..... 2614
6.4.1.21.3.5.32 Command 'Generate EtherCAT XML'. ..... 2614
6.4.1.21.3.5.33 Command 'Generate Sercos SCI XML'. ..... 2614
6.4.1.21.3.5.34 Command 'Disable Device' - 'Enable Device'. ..... 2614
6.4.1.21.3.5.35 Command 'Edit I/O Mapping'... ..... 2615
6.4.1.21.3.5.36 Command 'Import Mappings from CSV'. ..... 2615
6.4.1.21.3.5.37 Command 'Export Mappings to CSV'. ..... 2616
6.4.1.21.3.5.38 Command 'Read PLC Parameter File to Configuration'. ..... 2616
6.4.1.21.3.5.39 Command 'Online Config Mode'. ..... 2616
6.4.1.21.3.5.40 Command 'Runtime licensing'... ..... 2617

\section*{Command 'Add Object'}

Symbol: 蒀
Function: This command opens a submenu with objects that contain all objects that can be inserted, depending on the current position in the "Devices" or "POUs" view.

Call: "Project" menu, context menu in the "Devices" or "POUs" view.

Requirement: If CODESYS is to insert the object in the device tree, select an already existing object under which the new one is can be inserted indented. If CODESYS is to insert the object in the POUs tree, set the focus in any free place in the CODESYS window.

\section*{Command 'Add Folder'}

Symbol:
Function: This command opens a dialog box for defining a new folder in the Devices or POUs view.

Call: "Project" menu, context menu in the Devices or POUs view

You cannot structure the arrangement of device nodes and device objects through folders that you have created yourself.

This command inserts the folder below the object that has just been selected in the tree. If no object is selected, CODESYS inserts the folder right at the top in the tree directly under the root node.

\section*{Command 'Insert Device'}

Function: this command opens the dialog box "Add Device" for the selection of a device object that is to be inserted in the device tree below the currently selected object.
Call: Context menu of a device object in the device tree.
Requirement: An object is selected in the device tree below which a device object can be inserted.

See also
- \({ }^{\text {² }}\) Chapter 6.4.1.8 "Configuring I/O Links" on page 1835

Dialog box 'Add Function: Depending on the currently selected position in the device tree, the dialog box device' offers a selection of the devices that can be inserted at this point. In addition, it contains the commands also available in the context menu: "Insert Device", "Add Device", "Plug Device", "Update Device".
Requirement: The devices are installed in the device repository on the local system.

If you have opened the dialog box, it always displays the selection to suit the object currently selected in the device tree until you click "Close".
\begin{tabular}{|l|l|}
\hline "Name" & \begin{tabular}{l} 
Name with which the device is to appear in the device tree. Must be a valid IEC \\
identifier.
\end{tabular}
\end{tabular}

Table 483: "action"
\begin{tabular}{|l|l|}
\hline "Add device" & \begin{tabular}{l} 
CODESYS inserts the selected device indented below the selected object in the \\
device tree.
\end{tabular} \\
\hline "Insert device" & \begin{tabular}{l} 
CODESYS inserts the selected device at the same level as the selected object \\
below it in the device tree.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Plug device" & \begin{tabular}{l} 
CODESYS inserts the selected device in the selected slot. If the slot is already \\
occupied, the existing module is replaced by the new one.
\end{tabular} \\
\hline "Update device" & \begin{tabular}{l} 
CODESYS replaces the device selected in the device tree by the one selected. \\
Please note: Depending on the device, this may cause the configuration already \\
done in the device editor to be overwritten with the default values!
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "String for the full text search" & \begin{tabular}{l} 
This field is editable after clicking in it. For any character string entered, only \\
those devices that include the character string are displayed in the lower view. \\
The matched string is highlighted in yellow for these devices.
\end{tabular} \\
\hline "Vendor:" & Drop-down list with manufacturers whose available devices are displayed. \\
\hline "Group by category" & \begin{tabular}{l}
\(\square:\) The available devices (newest version) are sorted by category. The category \\
is defined in the device description file. \\
\(\square\)
\end{tabular} \\
\hline \begin{tabular}{l} 
"Display all versions (for \\
experts only)" available devices appear flat and alphabetically sorted.
\end{tabular} \\
\hline "Display outdated versions" & \begin{tabular}{l}
\(\square\) \\
\(\square\)
\end{tabular} \\
\hline\(\square:\) In addition, all other available versions of the devices can also be selected. \\
\(\square\) & \begin{tabular}{l}
\(\square\) \\
versions result, for example, from the update of plug-ins. \\
\(\square\)
\end{tabular} \\
\hline
\end{tabular}

The information provided by the device description file is displayed:
device name, vendor, categories, version, order number and a short description, device-specific bitmap.
See also
- \({ }^{4}\) Chapter 6.4.1.21.3.5.31 "Command 'Insert Device"" on page 2614
- \({ }^{4}\) Chapter 6.4.1.21.3.5.4 "Command 'Plug Device"' on page 2600
- \({ }^{\Perp}\) Chapter 6.4.1.21.3.5.6 "Command 'Update Device"' on page 2602

\section*{Command 'Plug Device'}

Function: Like the command "Add Device", this command opens the dialog box "Add Device" for the selection of a device object that is to be inserted in the device tree in the currently selected slot.

Call: Context menu of the slot of a device object in the device tree.
Requirement: The slot of a device object is selected in the device tree.
An empty slot is identified by the symbol k and the entry "<empty> (<empty>)". An occupied slot is given the symbol and the name of the device.
In the case of an occupied slot, this command replaces the existing module with the new one.
See also
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.5.3 "Command 'Insert Device'" on page 2599
- \(y_{y}\) Chapter 6.4.1.8 "Configuring I/O Links" on page 1835

\section*{Command 'Scan for Devices'}

Function: The command establishes a brief connection to the hardware and determines the devices in the network. Then you can apply the devices found into the device tree of your project.
Call: Menu bar: "Project"; context menu of a device object in the device tree

Requirement: The communication settings to the controller are correct. The gateway and the PLC are started. The device supports the scan function.
The following devices provide the scan function: EtherCAT master, EtherNet/IP Scanner (IEC), Sercos master, CANopen Manager, CANopen Manager SIL2, PROFINET controller and PROFIBUS DP master.


The command refers to the master controller selected in the device tree. For example, an already inserted PROFINET IO controller can be selected and the command used to determine the I/O devices and I/O modules assigned to it.
After performing the scan operation, the "Scan Devices" dialog opens and displays the found devices.

\section*{Dialog 'Scan \\ Devices'}

Table 484: "Scanned Devices"
\(\left.\begin{array}{|l|l|}\hline \text { "Device name, Device type, } \\ \text { Address, Station name, etc." } & \begin{array}{l}\text { Data about the scanned device depending on network type. } \\ \text { When you change a value in the list of scanned devices, the value is shown } \\ \text { in italics. This indicates that the new value has been changed in the editor in } \\ \text { CODESYS, but not in the device. When you download the value to the device, it } \\ \text { is shown normally. } \\ \text { Value that indicate differences between the project and the scanned device are } \\ \text { shown in orange. } \\ \text { If multiple device descriptions are available for the scanned device, then the }\end{array} \\ \text { name is displayed in bold. The selection of the matching device description is } \\ \text { resolved differently for different fieldbuses. For more detailed information, see } \\ \text { the corresponding fieldbus chapters. } \\ \text { If a device description cannot be found, then the following message is shown: } \\ \text { "Attention! The device was not found in the repository." Depending on the bus } \\ \text { system, additional information is displayed, such as manufacturer number and } \\ \text { product number. The device cannot be inserted into the project without the } \\ \text { installed device description. }\end{array}\right\}\)

NOTICE!
If you insert devices, which are available in the device tree, to the device tree with "Copy All Devices to Project", then the following should be noted. The data of the "Process Data" and "<...> I/O Mapping" tabs of the existing devices can be overwritten with the data of the recently inserted devices.

Table 485: "Configured Devices"
This part of the dialog is visible only when you select the "Show differences to project" option.
Differences between the scanned and configured devices are color-coded. Devices displayed in green are identical on both sides. Devices displayed in red are available only in the view of the scanned or configured devices.
\begin{tabular}{|l|l|}
\hline & \begin{tabular}{l} 
If you have selected a device in both views, then the scanned devices are \\
inserted above the selected configured device.
\end{tabular} \\
\hline If you have selected a device in both views, then the scanned devices are \\
inserted below the selected configured device. \\
\hline & \begin{tabular}{l} 
If you have selected a device in both views, then the configured devices are \\
replaced by the selected scanned device.
\end{tabular} \\
\hline
\end{tabular}

The dialogs for the scan differ depending on the type of device. See the help pages for the respective device editor.

\section*{Command 'Update Device'}

Function: Like the command "Add Device", this command opens the dialog box "Add Device" for the selection of a device object. This object is inserted in the device tree in place of the currently selected object.
Call: Context menu of a device object in the device tree.
Requirement: An object is selected in the device tree below which a device object can be inserted.

With this command you can insert either a different version of a device or a different type of device in place of the previous one.
The symbolic device name used in the device tree is retained, but the device type specified in parentheses behind it changes if a different type has been selected. Thus if only the device version is changed, the object entry appears unchanged.
If the device type does not change, the configuration tree indented below the device entry concerned is retained. In this case the configuration settings also remain the same. Inconsistencies in the configuration resulting from the device update are reported by CODESYS at the next compilation of the application. This also concerns implicitly inserted libraries, which CODESYS does not remove accordingly during a device update.
See also
- \({ }^{4}\) Chapter 6.4.1.8 "Configuring I/O Links" on page 1835
- \(\sum^{*}\) Chapter 6.4.1.21.3.5.3 "Command 'Insert Device"' on page 2599

\section*{Command 'Acknowledge Diagnosis', 'Acknowledge Diagnosis for Subtree'}

Function: The command acknowledges a diagnosis message.
Call: Context menu of a device object in the device tree
Requirement: The project is in online mode.

The "Acknowledge diagnosis" command acknowledges the diagnosis messages of an individual device. The "Acknowledge Diagnosis for Subtree" command also acknowledges the diagnosis messages of all subordinate devices. The diagnosis message of a pending malfunction is indicated by a red exclamation mark at the device object. The diagnosis message of a corrected malfunction is indicated by a gray exclamation mark.

\section*{Command 'Edit Object'}

Function: This command opens the object in its editor.
Call: Main menu "Project", context menu.
Requirement: An object is selected in the device tree or in the "POUs" view.

\section*{Command 'Edit Object with'}

Function: When multiple objects are available for an object, this command opens a dialog box for selecting an editor.

If only one editor is available for an object, then this command opens the object in that editor.
Call: Main menu "Project" or shortcut menu (right-click)
Requirement: An object is selected in the device tree or in the "POUs" view.
In the standard installation of CODESYS, there is no object that has multiple available editors.

\section*{Command 'Check integrity'}

Function: Automation Builder checks the project integrity for the complete project ("Project integrity" checks if all devices in the device tree are installed in the device repository).
Call: Main menu "Project", Context menu.
Requirement: A project is open.

\section*{Command 'Edit Object (Offline)'}

Function: The command opens the object offline in the editor.
Call: Main menu "Project", Context menu
Requirement: The application is in online mode. An object is selected in the device tree or in the "POUs" view.

The command allows you to edit objects in online mode. After editing you transfer the changes to the controller by use of the command "Online \(\rightarrow\) Online Change" or "Online \(\rightarrow\) Load".
See also
- \({ }^{〔}\) Chapter 6.4.1.21.3.7.6 "Command 'Online Change'" on page 2629
- \(\quad\) Chapter 6.4.1.21.3.7.5 "Command 'Load'" on page 2628

\section*{Command 'Set Active Application'}

Function: This command sets the selected application as the active application.
Call: Main menu "Project", or right-click the "Application" object.
Requirement: The project has at least two applications. The selected application is not active.
Online actions apply only to the active application. The name of an active application is displayed in bold typeface in the device tree.

\section*{Command 'Project information'}

Symbol:
Function: This command opens the dialog box "Project Information".
Call: Main menu"Project"
When you execute the command in the project for the first time, CODESYS creates the "Project Information" object.

See also
- \({ }^{4}\) Chapter 6.4.1.3.4.2 "Retrieving and Editing Project Information" on page 1813
- \(\sum^{*}\) Chapter 6.4.1.3.4.3 "Making project settings" on page 1814
- \& Chapter 6.4.1.21.2.24 "Object 'Project Information'" on page 2515

\section*{Command 'Project Settings'}

Symbol:
Function: This command opens the "Project Settings" dialog box.
Call: "Project" menu or double-click on the object "Project Settings" in the "POUs" view
Requirement: A project is open.
See also
- \(\sum^{*}\) Chapter 6.4.1.3.4.3 "Making project settings" on page 1814
- \({ }^{\circ}\) Chapter 6.4.1.21.4.12 "Dialog 'Project Settings'" on page 2766
- \(\sum^{*}\) Chapter 6.4.1.21.2.23 "Object 'Project Settings'" on page 2514

\section*{Command 'Project Environment'}

Function: This command opens the "Project Environment" dialog box.
Call: "Project" menu
Requirement: A project is open.
this command is for checking the currentness of software and files integrated in the project and enables them to be updated.

See also
- \(\quad{ }^{2}\) Chapter 6.4.1.21.4.13.2 "Dialog 'Project Environment' - 'Library Versions'" on page 2778
-
- \({ }^{〔}\) Chapter 6.4.1.21.4.13.3 "Dialog 'Project Environment' - 'Compiler Version'" on page 2778
- ⓨ Chapter 6.4.1.21.4.13.4 "Dialog 'Project Environment' - 'Device Versions'" on page 2779
- \({ }^{4}\) Chapter 6.4.1.21.4.13.5 "Dialog 'Project Environment' - 'Visualization Profile"" on page 2779
- 号 Chapter 6.4.1.21.4.13.6 "Dialog 'Project Environment' - 'Visualization Styles'" on page 2780
- \({ }^{4}\) Chapter 6.4.1.21.4.13.7 "Dialog 'Project Environment' - 'Visualization Symbols"" on page 2780

\section*{Command 'Project Localization' - 'Create Localization Template'}

Function: This command opens the "Create Localization Template" dialog. Define here which information should be exported from the project to a translation template (*.pot file).
Call: Menu bar: "Project \(\rightarrow\) Project Localization".
Requirement: A project is open.

Dialog 'Create
This dialog is used for selecting the textual information that should be used in the localization
Localization
template'
Table 486: "Include the Following Information"
\begin{tabular}{|c|c|}
\hline "Names" & Texts, such as dialog captions and object names in the device tree \\
\hline "Identifier" & Variable identifier (example: Counter) \\
\hline " Strings" & Example: 'count' in the following declaration: strVar: STRING := 'count'; \\
\hline "Comments" & Comment texts in the POUs \\
\hline "Position information" & \begin{tabular}{l}
Selection of which positions of the selected text categories in the project should be included in the translation file. The position information is located in the first line(s) of a segment for a translation. Example:
```

\#: D:\Proj1.project\Project_Settings:1
msgid "Project settings"
msgstr ""

``` \\
- "All": All detected positions of the text are listed. \\
- "First appearance": In the translation file, the position is included in the project where the text to be translated appears for the first time. \\
- "None"
\end{tabular} \\
\hline "Generate" & This button opens the dialog for saving a file. The translation template is created in a text file of type *.pot (portable object template). Each further generation creates a completely new template file. \\
\hline
\end{tabular}

See also
- Help about CODESYS Visualization: Multi-language capability

\section*{Command 'Project Localization' - 'Manage Localizations'}

Function: This command opens the "Manage localizations" dialog. Select the desired localization language in the dialog or the original version of the project. You can still accept the localization files *.<language>.po into the project or remove them.

Call: Menu bar: "Project \(\boldsymbol{\rightarrow}\) Project localization".
Requirement: A project is open.

Dialog 'Manage localizations'
\begin{tabular}{|l|l|}
\hline "Available Localizations" & \begin{tabular}{l} 
List of the localization files available in the project. Example: \\
proj1-de.po \\
proj1-en.po \\
<original version> \\
The original version is always available. The project can be edited only in the \\
original version.
\end{tabular} \\
\hline "Add" & \begin{tabular}{l} 
This button opens the dialog for selecting an additional po file from the file \\
system.
\end{tabular} \\
\hline "Remove" & \begin{tabular}{l} 
This button removes the po file, which is selected on the left side, from the \\
project.
\end{tabular} \\
\hline "Default localization" & \begin{tabular}{l} 
■: The selected localization is for the default localization. The entry is display in \\
bold.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Switch Localization" & Use this button to switch to the selected localization. \\
\hline "OK" & \begin{tabular}{l} 
The project is displayed in the language that is provided by the file selected \\
below the files. If you select "<original version>", then the project appears in the \\
editable non-localized version.
\end{tabular} \\
\hline \multicolumn{2}{|c|}{ See also }
\end{tabular}
- \({ }^{\star}\) Chapter 6.4.5.8 "Setting Up Multiple Languages" on page 2906

\section*{Command 'Project Localization' - 'Toggle Localization'}

Symbol:
Function: This command switches between the currently set project localization and the <original version>.

Call: Menu bar: "Project \(\rightarrow\) Project Localization"; button in the "Manage Localizations" dialog; button on the toolbar.
Requirement: A project is open. A default localization for the project is defined in the "Manage Localizations" dialog.
See also
- Help about CODESYS Visualization: Multi-language capability
- \(\Rightarrow\) Chapter 6.4.1.21.3.5.17 "Command 'Project Localization' - 'Manage Localizations'" on page 2605

\section*{Command 'Document'}

\section*{Symbol:}

Function: This command opens the "Document Project" dialog box, where you can define the project documentation. This includes the selection of objects in the open project that you want to print.
Call: Main menu "Project"

Table 487: "Document Project" dialog box
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
"Please select the objects \\
which are to be printed"
\end{tabular} & \begin{tabular}{l} 
Project tree view \\
In this view, you can select or clear objects for printing. \\
All objects are selected by default.
\end{tabular} \\
\hline "Title page" & \begin{tabular}{l} 
CODESYS creates a title page named "Project Documentation" with the fol- \\
lowing information: \\
- File: project file name \\
- \(\quad\) Date: Creation date of the project documentation \\
\(\bullet\) \\
Profile: CODESYS profile of the project
\end{tabular} \\
\hline "Table of contents" & CODESYS creates a table of contents for the project documentation. \\
\hline "Preview" & CODESYS creates and opens a print preview of the project documentation. \\
\hline "Select" & \begin{tabular}{l} 
CODESYS opens a drop-down list of all or single object types for the project \\
documentation.
\end{tabular} \\
\hline "Deselect" & \begin{tabular}{l} 
CODESYS opens a drop-down list of all or single object types that should be \\
excluded from the project documentation.
\end{tabular} \\
\hline "OK" & The "Print" dialog box opens. \\
\hline
\end{tabular}

See also
- \(\quad>\) Chapter 6.4.1.21.4.12.7 "Dialog 'Project Settings' - 'Page Setup'" on page 2772
- „y Chapter 6.4.1.21.3.2.12 "Command 'Print'" on page 2560

\section*{Command 'Compare objects'}

Function: To compare similar objects within a project.
Call: Main menu "Project", Context menu.
Requirement: Both projects have to be open.
ⓨ Chapter 6.3.2.2.3.3 "Comparing objects" on page 1456

\section*{Command 'Compare'}

\section*{Symbol: \(\sqrt{7}\)}

Function: This command opens the "Project Comparison" dialog. In this dialog, you define the reference project to compare with the current project. You configure the comparison process by means of options. When the dialog is exited, the comparison starts and the result is shown in the view "Project Compare - Differences".
Call: Menu bar: "Project \(\rightarrow\) Compare".
Requirement: A project is open.
See also
- \(\xi^{*}\) Chapter 6.4.1.5 "Comparing projects" on page 1817
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.5.22 "Command 'Commit Accepted Changes'" on page 2611

Dialog 'Project
Comparison'
Table 488: "Compare the currently open project with:"
\begin{tabular}{|l|l|}
\hline "Project on disk" & Path of the reference project on the file system. \\
\hline "Project in a source control & "Host": Name of the host where the source code management is located. \\
database" & "Port": Number of the port for connecting to the source code management. \\
& \begin{tabular}{l} 
"Location": Path of the reference project. \\
Requirement: The project is linked to source code management.
\end{tabular} \\
\hline
\end{tabular}

Table 489: "Compare Options"
\begin{tabular}{|l|l|}
\hline "Ignore whitespace" & \begin{tabular}{l}
\(\boxed{\square}:\) Whitespace differences between the current project and the reference project \\
are ignored.
\end{tabular} \\
\hline "Ignore comments" & \(\boxed{\square}\) : Comments in the programming code are excluded from the comparison. \\
\hline "Ignore properties" & \(\boxed{\square}\) : Object properties are excluded from the comparison. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "OK" & \begin{tabular}{l} 
Starts the project compare and displays the result in the view "Project compare - \\
Differences".
\end{tabular} \\
\hline
\end{tabular}

View 'Project
Comparison' 'Differences'

The project compare view opens when you click "OK" to close the "Project Compare" dialog.

(1) Object tree of the current project
(2) Object tree of the reference project
(3) Command 'Accept Block', command 'Accept Single'
(4) Compare options, configured in 'Project Compare' dialog Compare statistics: added, deleted, and changed objects

Table 490: Toolbar
\begin{tabular}{|c|c|}
\hline - & Switches to the detailed compare view "Project Comparison' - '<object name> Differences" for the object selected in the tree. Alternative: Double-click the object. \\
\hline \(\checkmark\) & Selects the next bottom object in the device tree where differences were detected. \\
\hline 令 & Selects the next top object in the device tree where differences were detected. \\
\hline * "Accept Block" & \begin{tabular}{l}
The block (selected object with all subordinate objects and units) is selected for acceptance from the reference block to the current block. \\
Repeated clicking of \(\approx\) "Accept Block" undoes the effects of its last change.
\end{tabular} \\
\hline " "Accept Single" & The object is selected in the current object for acceptance from the reference line. \\
\hline * & \begin{tabular}{l}
Requirement: The properties, access rights, or contents of the objects selected in the object tree are different. \\
Opens the "Accept" dialog.
\end{tabular} \\
\hline
\end{tabular}

Table 491: Display of differences with colors, and symbols
\begin{tabular}{|l|l|}
\hline Black font & Objects are identical. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline Object name with & Child objects of the object are different \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline Gray highlight & Objects are different. \\
\hline Gray highlight + bold blue font & Object is only in the reference project. \\
\hline \begin{tabular}{l} 
Gray highlight + bold green \\
font
\end{tabular} & Object is only in the open project (not in reference project). \\
\hline Gray highlight + red font + & Object has different properties. \\
\hline Gray highlight + red font + Q & Access rights of object and reference object are different. \\
\hline \begin{tabular}{l} 
Gray highlight + bold red font + \\
-
\end{tabular} & \begin{tabular}{l} 
Implementation of objects is different. \\
Double-click the line to display the object-specific compare view.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline Yellow highlight & Object is activated for acceptance. \\
\hline Yellow highlight + \({ }^{\text {+ }}\) & Adding the reference object to the open project is activated. \\
\hline Yellow highlight \(+{ }^{\text {E }}\) & Deleting the object (in the open project) is activated. \\
\hline Yellow highlight + 蒌 & Acceptance of the properties of the reference project is activated. \\
\hline Yellow highlight + red font + & Acceptance of the access rights of the reference project is activated. \\
\hline Gray highlight + bold red font + & Acceptance of the implementation of the reference project is activated. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Compare options" & Defined comparison options in the "Project Comparison" dialog. \\
\hline "Compare statistics" & \begin{tabular}{l} 
Number of additions, deletions, and changes in the current project, as compared \\
to the reference project. "Change" means differences of an object available in \\
both projects.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline\(\times\) & \begin{tabular}{l} 
The dialog prompt opens: "Do you want to commit the changes which you made \\
in the diff view?" \\
"Yes": The contents, properties, or access rights of the objects highlighted in \\
yellow are modified in the project. Now they correspond to the reference project. \\
Then the project compare view is closed completely.
\end{tabular} \\
\hline
\end{tabular}
\(\begin{array}{ll}\text { View 'Project } & \text { Function: Detail compare view } \\ \text { Comparison' - } & \text { Call in the project compare view: } \\ \text { '<object name> } & \text { Select an object that is identified as having different contents which you need to view in } \\ \text { Differences' } & \text { • } \begin{array}{l}\text { detail. Click } \Theta \text {. }\end{array} \\ & \text { • } \quad \text { Double-click the object. }\end{array}\)

Table 492: Toolbar
\begin{tabular}{|l|l|}
\hline\(\ominus\) & Switch back to the project compare view. \\
\hline\(\downarrow\) & Selects the next line below in the code where differences were detected. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline 令 & Selects the next line above in the code where differences were detected. \\
\hline \multirow{4}{*}{ "Accept Block"2 } & \begin{tabular}{l} 
The block (with all subordinate lines) is selected for acceptance of the reference \\
blocks into the current project. \\
A block in the detailed compare view consists of the unit where the cursor is \\
located and all corresponding units that have the same difference markers. A \\
unit is a line, network, or element. Subsequent lines of a line are examples of \\
corresponding units. \\
Repeated clicking of \(\approx\) "Accept Block" undoes the effects of its last change.
\end{tabular} \\
\hline\(\checkmark\) "Accept Single" & The line is selected in the current object for acceptance of the reference line.
\end{tabular}\(|\)\begin{tabular}{l} 
Switches between the default display where different units (lines, networks, ele- \\
ments) are displayed in red and another display: The units are displayed as \\
recently added in the open project. In the reference project, they are displayed \\
as deleted. \\
Available within the detailed compare view only. \\
Note: Depending on the display, detected differences in the statistics are \\
counted as changed, inserted, or deleted.
\end{tabular}

Table 493: Display of differences with colors, and symbols
\begin{tabular}{|l|l|}
\hline Black font & Objects are identical. \\
\hline \begin{tabular}{|l|l|}
\hline Gray highlight + bold blue font & Code is only in the reference project. \\
\hline \begin{tabular}{l} 
Gray highlight + bold green \\
font
\end{tabular} & Code is only in the current project (not in reference project). \\
\hline
\end{tabular}
\end{tabular}
\begin{tabular}{|l|l|}
\hline Yellow highlight & The object is activated for acceptance. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline\(\times\) & \begin{tabular}{l} 
The dialog prompt opens: "Do you want to commit the changes which you made \\
in the diff view?"
\end{tabular} \\
"Yes": The code highlighted in yellow is accepted into the project. The code \\
corresponds to the reference project. Then the detailed view is closed and the \\
project view is displayed. You can continue working with project compare.
\end{tabular}

\section*{Dialog 'Accept'}

Table 494: "Which meta data should be accepted?"
\begin{tabular}{|l|l|}
\hline "Access rights" & \(\boxed{ } \quad\) : Access rights that are selected for acceptance.
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Accepted groups" & \begin{tabular}{l} 
Grouping with access rights accepted by the reference project. A group is \\
accepted if it is present in both projects with different access rights. \\
Example: Group_A
\end{tabular} \\
\hline \begin{tabular}{l} 
"Unaccepted groups (missing \\
in a project)"
\end{tabular} & The group is not accepted if it is not present in one of the two projects. \\
\hline
\end{tabular}

\section*{"Properties"}
\(\square\) : Properties activated for accept
Requirement: The properties of the reference object and object are different.
"OK" \(\quad\) Settings are accepted.

\section*{Command 'Commit Accepted Changes'}

Symbol: 제
Function: This command commits the accepted differences from the project comparison to the current project.

Call: "Project \(\boldsymbol{\rightarrow}\) Commit Accepted Changes".
Requirement: Changes from the project comparison have been accepted.


See also
- « \({ }^{\circ}\) Chapter 6.4.1.5.3 "Opening the detailed compare view" on page 1818

\section*{Command 'Map pool devices'}

Symbol: [
Function: Maps imported devices from the device pool to already configured devices below a PLC.
Call: Main menu "Project", Context menu.
Requirement: A project is open.
* Chapter 6.6.1.1.8 "Arrange or map devices imported to the device pool" on page 4194

\section*{Command 'Export'}

Function: This command opens a dialog box for exporting objects from a project to an XML file. Call: Menu bar: "Project".

Dialog 'Export' This dialog box lists all objects from the device tree, POU tree, and module tree that CODESYS can export.
\begin{tabular}{|l|l|}
\hline One file per subtree & \begin{tabular}{l} 
V: CODESYS generates a separate export file for each subtree that is located \\
directly under the root node and includes selected files. \\
\(\square\)
\end{tabular} \\
\hline : CODESYS generates one export file for all selected objects.
\end{tabular}

See also
- \# Chapter 6.4.1.21.3.2.5 "Command 'Save Project as"' on page 2555
- * Chapter 6.4.1.21.3.5.25 "Command 'Import'" on page 2612
- « Chapter 6.4.1.4.2 "Exporting and importing projects" on page 1815

\section*{Command 'Import'}

Function: This command opens a dialog box for importing objects from an XML file.
Call: Menu "Project"
Requirement: A project is open.

Dialog box 'Im- The dialog box lists all objects from the export file that CODESYS can import at this point. port'
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
"Currently selected target \\
objects"
\end{tabular} & Object that is selected in the Device tree \\
\hline "Insertable items" & \begin{tabular}{l} 
Displays all objects of the export file that CODESYS can insert below the \\
selected object.
\end{tabular} \\
\hline "Show contents" & Displays the contents of the export file in a tree structure \\
\hline
\end{tabular}

\section*{Command 'Export PLCopenXML'}

Function: This command opens a dialog box for exporting objects from a project into an XML file in the PLCopen format.
Call: Menu "Project"

Dialog box 'Ex- The dialog box lists all objects from the Device tree that CODESYS can export into an XML file port PLCo- in accordance with the PLCopen format. penXML'

The PLCopenXML scheme does not permit VAR_GLOBAL and VAR_GLOBAL CONSTANT POUs to be in the same variable list. Therefore, if you wish to export both, you must first divide the variables into two separate variable lists.

See also
- \({ }^{\wedge}\), Chapter 6.4.1.21.3.5.27 "Command 'Import PLCopenXML"' on page 2612

\section*{Command 'Import PLCopenXML'}

Function: This command opens a dialog box for importing objects from an XML file in PLCopen format.
Call: Menu "Project"
Requirement: A project is open.

Dialog box 'Im- The dialog box lists all objects from the PLCopen export file that CODESYS can import at this port PLCo- point. penXML'
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
"Currently selected target \\
object"
\end{tabular} & Object that is selected in the Device tree \\
\hline "Insertable items" & \begin{tabular}{l} 
Displays all objects of the export file that CODESYS can insert below the \\
selected object.
\end{tabular} \\
\hline
\end{tabular}

The PLCopenXML scheme does not permit VAR_GLOBAL and VAR_GLOBAL CONSTANT POUs to be in the same variable list. Therefore, if you wish to export both, the variables must first be divided into two separate variable lists.

\section*{Command 'User management' - 'Log in User'}

\section*{Symbol: 9}

This command opens the dialog box "Login". Here you specify the project that you wish to edit and enter the login data for a user account with the corresponding rights. In addition, you can open the password manager from this dialog box.
The command is available in the menu "Project \(\rightarrow\) User Management".
See also
- ⓨ Chapter 6.4.1.6.7 "Logging in via User Account and Password Manager" on page 1827

\section*{Command 'User management' - 'Log out User'}

Symbol:
The user currently logged in to the project is logged out again with this command. This takes place without a dialog box or message, unless no user is currently logged in.

The command is available in the menu "Project \(\rightarrow\) User Management".
If the user is currently logged in to several projects or to libraries integrated in them (it does not have to be the same user account), then the dialog box "Logout" opens, in which the specific project or library project can be selected from which the current user is to be logged out.
The status bar always displays the user who is currently logged into the project.


A double-click on the field "Current user" in the status bar enables quick access to the "Login" or "Logout" dialog box.
See also
- \& Chapter 6.4.1.6.7 "Logging in via User Account and Password Manager" on page 1827
- \({ }^{*}>\) Chapter 6.4.1.21.3.5.28 "Command 'User management' - 'Log in User'"' on page 2613

\section*{Command 'User management' - 'Rights...'}

This command opens the dialog box "Rights", in which you define the actions that may be carried out, the user groups that may carry them out and the project objects on which they may be carried out.

The command is available in the menu "Project \(\rightarrow\) User Management".
See also
- \({ }^{\wedge}\) Chapter 6.4.1.6.6 "Protecting Objects in the Project by Access Rights" on page 1826

\section*{Command 'Insert Device'}

Function: Like the command "Add Device", this command opens a dialog box "Insert Device" for the selection of a device object. This object is inserted in the device tree at the same level as the currently selected object.

Call: Context menu of a device object in the device tree.
Requirement: An object is selected in the device tree below which a device object can be inserted at the same level.
See also
- \({ }^{4}\) Chapter 6.4.1.8 "Configuring I/O Links" on page 1835
- \(\sum^{*}\) Chapter 6.4.1.21.3.5.3 "Command 'Insert Device"' on page 2599

\section*{Command 'Generate EtherCAT XML'}


Function: This command opens the standard dialog box for saving a file in the local file system. You can define a name and a storage location for an xml file, in which CODESYS is to store the EtherCAT configuration of the EtherCAT master currently selected in the device tree. This may be necessary in order to operate an external EtherCAT stack.
Call: Context menu of an EtherCAT master device object in the device tree.
See also
- \({ }^{4}\) Chapter 6.4.1.8 "Configuring I/O Links" on page 1835

\section*{Command 'Generate Sercos SCI XML'}

The command is not integrated in the standard menu. You can add it via the dialog box "Tools \(\rightarrow\) Customize" from the category "Devices".

Function: This command opens the standard dialog box for saving a file in the local file system. You can define a name and a location for an xml file in which CODESYS then stores the configuration data of the sercos master currently selected in the device tree. This may be necessary in order to operate an external sercos stack.
Call: Context menu of a sercos master device object in the device tree.
See also
- \(\boldsymbol{y}^{*}\) Chapter 6.4.1.8 "Configuring I/O Links" on page 1835
- \(\Leftrightarrow\) Chapter 6.4.1.2.2.2 "Customizing menus" on page 1802

\section*{Command 'Disable Device' - 'Enable Device'}

Function: This command switches back and forth between the enabled (activated) and disabled (deactivated) states of a device in the bus system.
Call: Context menu of a device object in the device tree.
Requirement: The project is in offline mode. The bus driver must support the function.

A disabled device is not taken into account and is not addressed. Note that with some bus systems the deactivation of a node can lead to the master stopping.
The entry of a disabled device in the tree appears in light-gray lettering.
② Chapter 6.4.1.8 "Configuring I/O Links" on page 1835

\section*{Command 'Edit I/O Mapping'}

Function: This command opens the "Edit I/O Mapping" dialog box. This displays all I/O mappings of the currently selected device object, including I/O mappings of all additional device objects that are inserted in the device tree below this object.

Call: Context menu of a device object in the device tree.

Dialog box 'Edit You can edit the I/O mapping in this dialog box in exactly the same way as in the dialog box "//O I/O mapping' mapping" of the individual device editors. The respective other dialog boxes are directly updated accordingly.
\begin{tabular}{|l|l|}
\hline "Search" & \begin{tabular}{l} 
Input field for a search string for the mapping table. The search results are \\
marked in yellow.
\end{tabular} \\
\hline "Filter" & Drop-down list for filtering I/O assignments displayed listed in the mapping table: \\
& \(\bullet\) "Show all" \\
& \(\bullet\) "Show outputs only" \\
& \(\bullet\) "Show inputs only" \\
& - "Show unmapped variables only" \\
& - "Show mapped variables only" \\
& \(\bullet\) "Show mappings to existing variables only" \\
& \(\bullet\) "Show mappings to new variables only" \\
\hline
\end{tabular}

In the context menu you will find among other things the following commands:
"Export Mappings to CSV": Stores the mappings of a device and its sub-devices in an external file. To do this you select the device in the device tree or in the mapping list.
"Import Mappings from CSV": Inserts mappings from a file created beforehand by export.
See also
- \({ }^{4}\) Chapter 6.4.1.8 "Configuring I/O Links" on page 1835
- « \({ }^{*}\) Chapter 6.4.1.21.3.5.35 "Command 'Edit I/O Mapping'" on page 2615
- \({ }^{\star}\) Chapter 6.4.1.21.3.5.37 "Command 'Export Mappings to CSV'" on page 2616
- \({ }^{\bullet}\) Chapter 6.4.1.21.3.5.36 "Command 'Import Mappings from CSV'" on page 2615

\section*{Command 'Import Mappings from CSV'}

Function: The command opens the default dialog for opening a file in the local file system. The filter is set to the file format CSV in order to import the I/O mapping configuration of a device from the file which was exported previously by means of the "Export Mappings to CSV" command. CODESYS writes the configuration to the selected device.

Call: Context menu of a device object in the "Devices" view.
Requirement: A project is open with a device and an I/O mapping configuration. The device matches the exported CSV file.

\section*{NOTICE!}

I/O mapping configurations are stored in CSV files with the semicolon separator.
These files can be edited manually. If the files are edited manually, then it is imperative that this format is retained in order to import successfully. Note: The entries of the file to the I/O mapping of the device are assigned by the device name and the parameter name. Parameter names that are not unique are numbered sequentially in this file ( \(@<n>\) ).
Fields without contents in the CSV file are ignored at import. To remove an existing entry in the I/O mapping by importing, you have to add a space in the respective field in the CSV file.

See also
- \& Chapter 6.4.1.21.3.5.37 "Command 'Export Mappings to CSV"' on page 2616
- Ehapter 6.4.1.8.2 "Configuring Devices and I/O Mapping" on page 1836

\section*{Command 'Export Mappings to CSV'}

Function: The command opens the default dialog for saving a file to the local file system. The filter is set to file format CSV. After specifying a name and a location, CODESYS stores the I/O mapping configuration in a CSV file with the semicolon separator.

Call: Context menu of a device object in the "Devices" view.
Requirement: A device object with an I/O mapping configuration is selected in the device tree.


See also
- Ela Chapter 6.4.1.21.3.5.36 "Command 'Import Mappings from CSV"' on page 2615
- © Chapter 6.4.1.8.2 "Configuring Devices and I/O Mapping" on page 1836

\section*{Command 'Read PLC Parameter File to Configuration'}

Function: This command reads the configuration file IoConfig.par of the PLC and stores the values in the project. Such a file is created if the parameters of the PLC have been changed by another device, for example via a visualization. Then these parameters are changed only in the memory of the PLC, but not in the configuration of the project.
Call: Context menu of the PLC device object
Requirement: You have made the command available using the dialog in "Tools \(\boldsymbol{\rightarrow}\) Customize".

\section*{Command 'Online Config Mode'}

Function: This command is for switching the online configuration mode on and off. At switch-on it establishes a connection to the PLC and loads an implicitly created application "HiddenOnlineConfigModeApp" to the PLC. Depending on the device, CODESYS goes into simple online configuration mode or a dialog box appears for selecting between simple and advanced online configuration mode.
Call: Context menu of the PLC object in the device tree
Requirement: The communication settings for the PLC device are correctly set.

Simple online configuration mode:
This command creates the implicit application HiddenOnlineConfigModeApp and loads it to the controller. The application automatically initializes all inputs and outputs of the controller once. After that you can access the I/Os as follows:
- Read I/Os
- Write outputs
- Diagnosis (in the device tree and on the "Status" tab of the device editor)
- Scan (of the current hardware)
- Interactive online functions, if supported (for example, writing asynchronous messages)

Advanced online configuration mode (parameter mode):
If there are already applications on the PLC and the controller supports it, the command first opens the dialog box "Devices", which displays the applications existing on the controller. From this dialog you can connect via the button "Parameter mode" to the PLC and then access the values of the device parameters without having to log in with a real application.

\section*{Writing and forcing in the I/O mapping}

In online configuration mode the writing and forcing of values on the "//O
Mapping" tab works differently to the way it works in real online mode. The outputs are written immediately after insertion into the table. There is no "Prepared Value" column; instead, the initial values can be changed directly after a doubleclick on the column "Current Value".

Dialog box This dialog box appears after the command "Online Config Mode" if the device supports the 'Config applica- advanced online configuration mode and there are already real applications on the controller. tion mode'
\begin{tabular}{|l|l|}
\hline "Parameter mode" & \begin{tabular}{l} 
The controller configuration in the project is compared with that on the device. \\
If they correspond, CODESYS establishes a connection to the PLC. Unlike the \\
simple online configuration mode it permits the reading and - if supported by the \\
driver - the writing of parameters in the generic device editor. The applications \\
already loaded to the device remain unchanged in this case!
\end{tabular} \\
\hline "Config application mode" & CODESYS switches to the 'simple online configuration mode'. \\
\hline
\end{tabular}

\section*{Command 'Runtime licensing'}

\section*{Symbol: \([\)}

Function: Management of runtime licenses on the PLC. The use of some libraries and devices require the PLC to have a runtime license.
Call: Main menu "Project", Context menu. Displayed only offline.
Requirement: A project is open. Log-in required for managing runtime licenses without need for memory card.
The license status of a PLC can be displayed at any time \({ }^{\Perp}\) Chapter 6.3.2.2.2.6 "View license information" on page 1453.

\section*{Menu 'Build'}
6.4.1.21.3.6.1 Command 'Generate Code' ..... 2618
6.4.1.21.3.6.2 Command 'Clean'. ..... 2618
6.4.1.21.3.6.3 Command 'Clean All'. ..... 2618
6.4.1.21.3.6.4 Command 'Build'. ..... 2619
6.4.1.21.3.6.5 Command 'Rebuild'. ..... 2619
6.4.1.21.3.6.6 Command 'Generate Runtime System Files'. ..... 2619
6.4.1.21.3.6.7 Command 'Check all Pool Objects'. ..... 2621
6.4.1.21.3.6. \(\quad\) Command 'Generate Code for Active Application'. ..... 2621
6.4.1.21.3.6.9 Command 'Check All Application Objects' ..... 2621
6.4.1.21.3.6.10 Command 'Check Library Compatibility'. ..... 2622
6.4.1.21.3.6.11 Command 'Generate Disassembly File'. ..... 2622

\section*{Command 'Generate Code'}

Symbol (\#; shortcut: [F11]
Function: The command starts the code generation for the active application.
Call: Menu bar: "Build"
When generating code with this command, code is generated as when downloading the application to the PLC, but the code is not transferred to the PLC. At this time, other source code tests are performed As a result, you can check the code for bugs that were not detected by the compiler and for fixing any bugs before the code is used in online mode.
See also
- ※y Chapter 6.4.1.11.5 "Generating Application Code" on page 1976

\section*{Command 'Clean'}

Function: This command deletes the build information for the active application.
Call: Main menu "Build".
During the last download, the build information was created and saved to a file (*.compileinfo).
After a cleaning process, an online change is no longer possible for the affected application. The application must be fully downloaded to the controller again.

See also
- \({ }^{4}\) Chapter 6.4.1.21.3.6.3 "Command 'Clean All"' on page 2618

\section*{Command 'Clean All'}

Function: This command deletes the build information for all applications in the project.
Call: Main menu "Build".
During the last download, the build information was created in the local file system and saved to a file ( \({ }^{*}\).compileinfo).
This command requires a download before another login An online change is no longer possible. As compared to the "Clean" command (only the active application), CODESYS regenerates the language model for all objects, which is very time-consuming.

\section*{NOTICE!}

Reconsider carefully whether or not executing this command is really necessary. If you only want to rebuild and download the active application, then execute the "Clean" command.

See also
- * \({ }^{*}\) Chapter 6.4.1.21.3.6.2 "Command 'Clean'" on page 2618

\section*{Command 'Build'}

Function: The command starts the build operation for the active application.
Call: The command is not in any menu by default. You can add it to a menu by using the dialog from "Tools \(\rightarrow\) Customize" (command category "Build").
During the build operation, CODESYS performs a syntactic validation of all objects in the application. However, code is not generated like at log in to the target system or download of the application. The build operation is always performed automatically when you log in with a changed program.

When the check is complete, CODESYS displays any error messages or warnings in the message view ("Build" category).
If the program has not been changed since it was compiled without errors the last time, then it is not recompiled. The message "The application is current" appears. If the syntactic validation is repeated, then you must execute the "Rebuild" command.
See also
- \(\quad\) \& Chapter 6.4.1.21.3.6.5 "Command 'Rebuild'" on page 2619

\section*{Command 'Rebuild'}

Function: The command starts the build operation for the active application, even if the last build contained errors.

Call: The command is not in any menu by default. You can add it to a menu by using the dialog from "Tools \(\rightarrow\) Customize" (command category "Build").

See also
- \(\quad \Leftrightarrow\) Chapter 6.4.1.21.3.6.4 "Command 'Build'" on page 2619

\section*{Command 'Generate Runtime System Files'}

Function: The command generates a C stub file and an M4 interface file from the current library project. These files are used as the basis for creating an external library file.
Call: Menu bar: "Build"
Requirement: A library project is open.
The command opens the "Generate Files for Runtime System" dialog.
\begin{tabular}{|l|l|}
\hline "Output directory" & \begin{tabular}{l} 
Directory where CODESYS creates the runtime system files. Click the ... button \\
to open the default dialog for browsing the file system.
\end{tabular} \\
\hline "Component names" & Name of the library project \\
\hline \multicolumn{2}{|l|}{ "Which files do you want to create?" } \\
\hline
\end{tabular}


\section*{Command 'Check all Pool Objects'}

Symbol i*; shortcut: [F11]
Function: The command starts a build operation (a syntax check) for all pool objects that are managed in the POU view and as a result are available throughout the project. First and foremost, this is useful when creating libraries.
Call: Menu bar: "Build"
Requirement: A library project is open.

\section*{NOTICE!}

The command does not result in code generation. In addition, no file is created in the project directory with information about the build operation.

See also
- \# Chapter 6.4.1.21.4.15.2 "Dialog 'Customize' - 'Menu"' on page 2801
- \(\xi^{*}\) Chapter 6.4.1.17.3 "Information for Library Developers" on page 2035

\section*{Command 'Generate Code for Active Application'}

Function: The command generates the code for the application of a library project.
Call: Menu bar: "Build"
Requirement: The project contains an application.
- A library project is open.
- The library project contains an application.

When generating code with this command, code is generated as when downloading the application to the PLC, but the code is not transferred to the PLC. At this time, other source code tests are performed As a result, you can check the code for bugs that were not detected by the compiler and for fixing any bugs before the code is used in online mode.
See also
- Chapter 6.4.1.21.3.6.6 "Command 'Generate Runtime System Files'" on page 2619

\section*{Command 'Check All Application Objects'}

Function: This command starts a build operation for all objects of the active application, even for the POUs that are not used by the application. After the build operation, the errors that were found in the unused objects are also displayed in the message window.
Call: The command is not in any menu by default. You can add it to a menu by using the dialog from "Tools \(\boldsymbol{\rightarrow}\) Customize" (command category "Build").
Requirement: An application of the open project is active.

\section*{NOTICE!}

The command does not result in code generation. In addition, no file is created in the project directory with information about the build operation.

See also
- © Chapter 6.4.1.21.4.15.2 "Dialog 'Customize' - 'Menu"' on page 2801

\section*{Command 'Check Library Compatibility'}

Function: The command triggers a check whether the currently opened library project is compatible with the last installed version of this library (next lower version number).

Call: By default the command is not available in any menu. You can add it to a menu by using the "Tools \(\rightarrow\) Customize" dialog, command category "Build".
Requirement: A library project is opened.
The check regards differences in the implemented interfaces of a method. So, after the check you will get displayed error messages in the messages window in the following cases:
- Adding or removing inputs or outputs of function blocks, functions or methods
- Changing the data type of inputs or outputs
- Modifying the implemented interfaces of a method

See also
- \(\quad \stackrel{y}{ }\) Chapter 6.4.1.21.4.15.2 "Dialog 'Customize' - 'Menu"' on page 2801
- \({ }^{*}\) Chapter 6.4.1.17.3 "Information for Library Developers" on page 2035

\section*{Command 'Generate Disassembly File'}

Function: This command generates a disassembly file <project name>. asm from the current project and saves it in the file directory in the project folder.
Call: The command is not in any menu by default. You can add it to a menu by using the dialog box from "Tools \(\rightarrow\) Customize" (command category "Build").
See also
- \({ }^{\mu}\) Chapter 6.4.1.21.4.15.2 "Dialog 'Customize' - 'Menu"' on page 2801

\section*{Menu 'Online'}
6.4.1.21.3.7.1 Command 'Choose Active Application'. ..... 2623
6.4.1.21.3.7.2 Command 'Login'. ..... 2624
6.4.1.21.3.7.3 Command 'Logout'. ..... 2627
6.4.1.21.3.7.4 Command 'Create Boot Application' ..... 2627
6.4.1.21.3.7.5 Command 'Load' ..... 2628
6.4.1.21.3.7.6 Command 'Online Change' ..... 2629
6.4.1.21.3.7.7 Command 'Source Download to Connected Device'... ..... 2631
6.4.1.21.3.7.8 Command 'Download Manager' ..... 2631
6.4.1.21.3.7.9 Command 'Multiple Download'. ..... 2631
6.4.1.21.3.7.10 Command 'Reset Cold' ..... 2633
6.4.1.21.3.7.11 Command 'Reset Warm' ..... 2634
6.4.1.21.3.7.12 Command 'Reset Origin' ..... 2635
6.4.1.21.3.7.13 Command 'Reset Origin Device' ..... 2636
6.4.1.21.3.7.14 Command 'Logoff Current Device User' ..... 2637
6.4.1.21.3.7.15 Command 'Download'... ..... 2637
6.4.1.21.3.7.16 Command 'Add Device User'. ..... 2637
6.4.1.21.3.7.17 Command 'Remove Device User' ..... 2638
6.4.1.21.3.7.18 Command 'Change Password Device User' ..... 2639
6.4.1.21.3.7.19 Command 'Stop Execution on Handled Exceptions' ..... 2639
6.4.1.21.3.7.20 Command 'Connect to Device' ..... 2640
6.4.1.21.3.7.21 Command 'Disconnect from Device'. ..... 2640
6.4.1.21.3.7.22 Command 'Wink' ..... 2640
6.4.1.21.3.7.23 Command 'Simulation. ..... 2640
6.4.1.21.3.7.24 Command 'Operating Mode'. ..... 2642
6.4.1.21.3.7.25 Command 'Virtual mode' ..... 2643
6.4.1.21.3.7.26 Command 'Virtual system testing'... ..... 2644

\section*{Command 'Choose Active Application'}

Symbol: \(\qquad\)
The command is implemented as a list box from which you can set an application active. By default, the list box is located on the toolbar.

Function: The list box displays the currently active application with its device path.
Call: The list box contains all applications that are organized in the "Devices" view. By clicking an entry in the list box, you activate the selected application.
Requirement: The project has multiple applications.
\begin{tabular}{|c|c|c|}
\hline \multirow[t]{5}{*}{Example} &  & \\
\hline & App [Device: PLC Logic] & - \(\square 4\) \\
\hline & App [Device: PLC Logic] & \\
\hline & App [Device2: PLC Logic] & \\
\hline & SubApp [Device: PLC Logic: App] SubSubApp [Device: PLC Logic: App: SubApp] & \\
\hline
\end{tabular}

When you call commands in the "Build" oder "Online" menus, these commands are applied to the active application. This is displayed in the list box, and also displayed in bold in the device tree. In particular, this applies to the "Build
\(\rightarrow\) Build" and "Online \(\rightarrow\) Login" commands.
You can also access these commands using the command icons on the toolbar where the list box is located. When the command icons are called, they are also applied to the active application.
However, if you call a command from the context menu of a device object in the device tree, then the command is applied to the corresponding object. For example, by calling you can establish a connection to an application on the device which is not active.

See also
- \& Chapter 6.4.1.21.3.5.12 "Command 'Set Active Application'" on page 2603
- \(\xi^{\sharp}\) Chapter 6.4.1.21.3.6.4 "Command 'Build'" on page 2619
- \({ }^{\circ}\) Chapter 6.4.1.21.3.7.2 "Command 'Login'" on page 2624

\section*{Command 'Login'}

Symbol: ©s; shortcut: [AIt]+[F8].
Function: The command connects the application to the target system (PLC to simulated device) and starts the online mode.

\section*{Call: Menu bar: "Online"; context menu of an "Application" object}

Requirement: The application contains no errors and the communication settings are configured.

A dialog prompt opens if the communication settings are incorrect. You can then switch directly to the "Communication Settings" of the PLC.
If you click "Login" from the online menu, then the currently active application is connected to the target system. If you choose this command from the context menu (right-click) while an application is selected in the device tree, then that application is logged in, even if it is not set as the active application.

If an online user management is configured on the target device, then you are prompted for user data when you log in. The "Device User Login" dialog opens for this.

\section*{CAUTION!}

\section*{Check controller accessibility}

For security reasons, controllers should not be accessible from the Internet or untrusted Networks under any circumstances! In particular, the TCP/IP programming ports (usually UDP-Ports 1740.. 1743 and TCP-Ports \(1217+11740\) or the controller specific ports) should not be accessible from the internet without protection. In case Internet access to the controller is needed, using a a safe mechanism is absolutely mandatory, such as VPN and password protection of the controller.
«̌ Chapter 6.4.1.11.4 "Handling of Device User Management" on page 1971

\section*{NOTICE!}

If a safety controller is inserted below a controller, then this command can interrupt the communication connections temporarily.
Connections of the safety controller to other safety controllers (via safety network variables), to field devices, and to the development system are affected. The safe field devices or other safety controller can enter the safe state as a reaction. The connection to the development system is affected only when a safety controller that is connected to the main controller via a fieldbus.
For more information, see the "Subordinate Safety Controllers" chapter.

Possible situations when logging in:
- A later version of the device description (than in the project) is on the PLC. A warning prompt is displayed with the option to cancel the process.
- The application does not exist on the PLC: You are prompted to confirm the download.
- The application is already on the PLC and has not been changed since the last download. The login continues without any more prompts.
- The application exists on the PLC, but it has been changed since the last download. You are prompted to select one of the following options:
- Login with online change (Note the information about online changes in the help page "Command 'Online Change' ".)
- Login with download
- Login without any change

The position also provides the option of updating the boot application on the PLC.
- An unknown version of the application exists on the PLC. CODESYS prompts you to replace it.
- A version of the application exists on the PLC and is running. CODESYS prompts you to log in anyway and overwrite the currently running application.
- The application on the PLC is currently halted at a breakpoint. You are logged out and the program has been changed: CODESYS prompts you with a warning that the PLC will be stopped completely if an online change or download occurs. This happens also if several tasks exist and the breakpoint affects only one of them.

Click "Details" in the dialogs above to open the "Application Information" dialog.


In CODESYS V3.5 SP17 and higher, only exactly one CODESYS instance can ever be logged in to an application of a controller. If a second CODESYS instance wants to log in to the same application of the same controller, then an error message is displayed.

See also
- « \({ }^{*}\) Chapter 6.4.1.21.3.7.6 "Command 'Online Change"' on page 2629
- \& Chapter 6.4.1.10.7 "Subordinate safety controller" on page 1964

Dialog 'Applica- The dialog provides two tabs with comparative information about the application changed in the tion Information' development system and its previous version currently located on the PLC. There are two tabs: (Details)
- "Application information": The application properties of the "Application in the IDE" (Integrated Development Environment) are compared with those of the "Application in the PLC": Project name, Last modification, IDE version, Author, Description. In addition, CODESYS shows the objects that have changed since the last download.
- "Application contents": When the "Download application info" is selected, the contents of the applications on both the (1) development system and (2) PLC can be compared. The "Download application info" option is located on the "Application Build Options" tab of the application properties.
If the code in the development system is not current, then (3) "Application not up to date. Generate code now?" appears at the bottom left of the dialog. Execute this command to update the application source code.
This detailed information can help you to better assess the effects of login in the current situation and to make a decision about downloading the new application.


See also
- \& Chapter 6.4.1.21.2.8.5 "Tab 'Applications"' on page 2434
- \& Chapter 6.4.1.5 "Comparing projects" on page 1817

Unknown appli- If one or more applications are already on the PLC, but are not in the project, then CODESYS

\section*{cations on the}

PLC opens a dialog with a list of these applications. You can then define whether an application should be deleted before loading the current application from the PLC. This also applies to child applications that are on the PLC, but have been deleted from the project in the meantime.

\section*{Compiling the project before login}

If an application program has not been compiled since the last change, then CODESYS compiles the project before login. This operation is the same as the "Generate Code" command when logged out.
If compile errors occur, then a dialog prompt opens. The errors are displayed in the message view in the "Build" category. You can then decide whether or not you log in without downloading the program to the PLC.
See also
- \(\boldsymbol{y}^{\text {® }}\) Chapter 6.4.1.21.3.6.4 "Command 'Build"' on page 2619

Error at login If an error occurs when logging in to the PLC, then CODESYS cancels the loading operation with an error message. The error dialog gives you the options of showing the error details. If an exception was thrown and the text *SOURCEPOSITION* is included in the log, then you can display the affected function in the editor by clicking "Show in Editor". The cursor jumps to the line containing the error.

Messages during the download operation

If CODESYS downloads the project to the PLC at login, then the following information is printed to the message view:
- Generated code size
- Size of the global data
- Resulting memory requirement on the PLC
- List of the affected blocks (for online change)

In online mode, you cannot change the settings of the devices or modules. You have to logout of the application for changing device parameters. Depending on the bus system, there may be some special parameters that you can also change in online mode.

CODESYS saves the view configuration separately in online and offline mode. In addition, views are closed that cannot be used in any operating mode. For this reason, the view can change automatically at login.

\section*{Command 'Logout'}

Symbol: keyboard shortcut: [Ctrl]+[F8].
Function: This command disconnects the application from the target system (controller or simulated device) and returns to offline mode.

Call: Main menu "Online", or context menu of the "Application" object.

\section*{Command 'Create Boot Application'}

Function: This command generates a boot application.
Call: Main menu "Online".

A boot application is the application that is started automatically when the controller is switched on or started.
In offline mode, you can save the boot application in any directory. In online mode, CODESYS save the boot application to the target device. The file name is <application name>.app.

See also
- ⓨ Chapter 6.4.1.11.7 "Generating boot applications" on page 1978
- \({ }^{4}\) Chapter 6.4.1.21.2.2 "Object 'Application'" on page 2410

\section*{Command 'Load'}

Function: This command causes a compilation of the active application with subsequent download to the controller.
Call: Menu bar: "Online".
Requirement: The application is in online mode.
When you execute this command, CODESYS performs a syntax check and generates the application code. This code is downloaded to the PLC. Furthermore, CODESYS generates the build log <project name>.<device name>.<application ID>.compile info in the project directory.

\section*{NOTICE!}

During loading all variables are re-initialized with the exception of persistent variables.

\section*{NOTICE!}

If a safety controller is inserted below a controller, then this command can interrupt the communication connections temporarily.
Connections of the safety controller to other safety controllers (via safety network variables), to field devices, and to the development system are affected. The safe field devices or other safety controller can enter the safe state as a reaction. The connection to the development system is affected only when a safety controller that is connected to the main controller via a fieldbus.
For more information, see the "Subordinate Safety Controllers" chapter.

The description of the "Login" command describes the possible situations when logging in and loading.
If you attempt to download an application when the same version of the application is already on the PLC, then you get the message: "Program is unchanged. Application was not downloaded". CODESYS downloads the application to the PLC.
During loading a record of the actions being executed (generation of code, execution of initialization, etc.) appears in the Message window in the message category "Compile". Furthermore, information is displayed regarding the memory ranges, the size of the code, the global data and the allocated memory. For the purpose of clarity, as opposed to the online change, the modified function blocks are no longer listed.
See also
- \({ }^{〔}\) Chapter 6.4.1.21.3.7.2 "Command 'Login'" on page 2624
- «y Chapter 6.4.1.10.7 "Subordinate safety controller" on page 1964

\section*{Command 'Online Change'}

Function: The command is used for initiating an online change on the current application. When this is done, CODESYS re-downloads only the changed parts of an application that is already running on the PLC.

Call: Menu bar: "Online"; context menu of an "Application" object
Requirement: The application is in online mode.
The command is available in the context menu if an application is selected in the device tree. In this way, you can perform an online change just for one application, even if that application is not currently active.

\section*{CAUTION!}

An online change modifies the running application program and does not cause a restart.
Make sure that the new application code still has the required effect on the controlled system.
Depending on the controlled plant, the plant and workpieces may be damaged or the health and life of persons could be endangered.

\section*{NOTICE!}
1. When an online change is performed, the application-specific initializations (example: homing) are not executed because the machine retains its status. For this reason, the new program code may not have the intended effect.
2. Pointer variables retain their value from the last cycle. When a pointer refers to a variable whose value was changed in an online change, the variable no longer yields the correct value. Make sure that the pointers are re-assigned in each cycle.
3. After the parent application has been changed, a child application is removed from the controller when an online change is performed.

\section*{NOTICE!}

For compiler version >=3.5.0.0, a fast online change is performed for minor changes. In this case, only the modified blocks are compiled and downloaded. In particular, no initialization code is generated. This means that also no code is generated when variables with the init_on_onlchange attribute are initialized. As a rule, this has no effect because the attribute is used primarily for initializing variables with addresses. However, it cannot happen that a variable changes its address during an online change.
To secure the effect of the init_on_onlchange attribute in the entire application code, you must deactivate the fast online change in general for the application by using the compiler definition no_fast_online_change. To do this, insert the definition in the application "Properties" ("Build" \(\overline{\text { tab }}\) ).

At the time of download, CODESYS also lists the changed interfaces, affected variables, and all blocks with new generated code in the "Build" category of the message view. If memory locations change, a dialog will inform you of possible problems in conjunction with pointers.

In the "Online Change Memory Reserve" view, memory reserves can be configured for the online change so that instance variables do not have to be moved in the memory when changing a function block in an online change.

\section*{What prevents an online change?}

There are actions in CODESYS after which an online change on a controller is no longer possible. Afterwards, the application always has to be completely recompiled. A typical case is the "Clean All" action which deletes the compile information stored at the last download. However, these kind of actions typically generate a warning which you have to acknowledge.

But there are also "normal" editing actions that result in an online change not being possible at the next login. Therefore, pay attention to the following symbol in the status bar when editing in the program POUs: 包. When this symbol turns red in color ( 0 the controller can be performed. Double-clicking the symbol opens the "Application Information" dialog with a list of differences to the last download. In the dialog, you also find information about which of the changes prevent an online change.

Actions and changes in different areas of an application that prevent an online change:
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
Check func- \\
tions
\end{tabular} & \begin{tabular}{l} 
Activation or removal of a check function (CheckBounds, CheckRange, \\
CheckDiv, etc.) \\
Change in an interface of a check function (also the insertion and deletion of \\
local variables)
\end{tabular} \\
\hline \begin{tabular}{l} 
Task configu- \\
ration
\end{tabular} & Change in the configuration settings \\
\hline \begin{tabular}{l} 
Project set- \\
tings
\end{tabular} & \begin{tabular}{l} 
Change of the "Compile Options" in the "Settings" section (Unicode, replace \\
constants, logging in, breakpoints) \\
Change in the "Compiler defines"
\end{tabular} \\
\hline \begin{tabular}{l} 
Application \\
properties
\end{tabular} & Change of the "Target system memory settings" ("Build" tab) \\
\hline \begin{tabular}{l} 
POU proper- \\
ties
\end{tabular} & Change of the "External implementation" option ("Build" tab) \\
\hline \begin{tabular}{l} 
Task-local \\
global vari- \\
able list
\end{tabular} & \begin{tabular}{l} 
All changes \\
\hline \begin{tabular}{l} 
Function \\
block
\end{tabular} \\
\begin{tabular}{l} 
Change of the base POU of a function block (EXTENDS FBbase), also the \\
insertion or deletion of such a base POU \\
Change in the interface list (IMPLEMENTS ITF). Exception: Adding a new \\
interface at the end of a list
\end{tabular} \\
\hline Data type \\
\begin{tabular}{l} 
Change of the data type of a variable from a user-defined data type to another \\
user-defined data type (for example, from TON to TOF) \\
Change of the data type from a user-defined data type to a base type (for \\
example, from TON to TIME) \\
Note: As a workaround, you should always change the name of the variable \\
together with the data type. Then the variable is initialized as a new variable \\
and the old one is removed. Then an online change is possible.
\end{tabular} \\
\hline \begin{tabular}{ll} 
Alarm config- \\
uration
\end{tabular} \\
\begin{tabular}{l} 
Change in the alarm database configuration \\
Change of the number of latch variables (also has an effect on the memory \\
format in the database) \\
Change to the configuration of the distributed alarms
\end{tabular} \\
\hline Data source
\end{tabular} \begin{tabular}{l} 
All changes in the configuration \\
\hline Device config- \\
uration
\end{tabular} \\
\begin{tabular}{l} 
Change in the device tree (also by the "Update Device" command) \\
Change in a device configuration: By default, changes to device parameters \\
are not capable of online change. However, exceptions can be configured in \\
the device description. \\
Note: I/O mapping to variables is possible by online change.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline Visualization & \begin{tabular}{l} 
Toggling of the overlay function \\
Before V3.5 SP6: Change in the configuration of the trace element \\
Note: In V3.5 SP6 and higher, the following applies: For online changes \\
that affect visualizations or affect the data of the application (for example, \\
a new variable is inserted), the visualization is completely reinitialized. For \\
TargetVisu, for example, this means that the visualization closes and reopens \\
with the start page. For WebVisu, the visualization also restarts with the start \\
visualization after a short waiting period.
\end{tabular} \\
\hline \begin{tabular}{l} 
Unit conver- \\
sion
\end{tabular} & \begin{tabular}{l} 
Insertion or removal of objects for unit conversion
\end{tabular} \\
\hline Trend & \begin{tabular}{l} 
Change of the number of variables or maximum number of variables. Change \\
of the number of variables with a description or special line settings
\end{tabular} \\
\hline
\end{tabular}

See also
- \(y^{4}\) Chapter 6.4.1.21.3.6.3 "Command 'Clean All"" on page 2618
- \({ }^{4}\) Chapter 6.4.1.11.5 "Generating Application Code" on page 1976
- \(\sum^{\Downarrow}\) Chapter 6.4.1.14.2 "Executing the online change" on page 2025
- « " Dialog 'Application Information' (Details)" on page 2626
- \({ }^{*}\) Chapter 6.4.1.20.6.3.21 "Attribute 'init_on_onlchange"' on page 2291
- \(\sum^{*}\) Chapter 6.4.1.21.2.30.2 "Tab 'Configuration"' on page 2538
- \& Chapter 6.4.1.21.4.12.4 "Dialog Box 'Project Settings' - 'Compileoptions'" on page 2769
- \(y^{*}\) Chapter 6.4.1.21.4.11.5 "Dialog 'Properties' - 'Build"' on page 2755

\section*{Command 'Source Download to Connected Device'}

Function: This command loads the project source code (as project archive) to the controller currently connected.
Call: Main menu "Online".
Requirement: The application is in online mode.
See also
- \(\quad\) \& Chapter 6.4.1.21.3.2.10 "Command 'Source Upload'" on page 2559
- \({ }^{\bullet}\), Chapter 6.4.1.21.3.2.11 "Command 'Source Download'" on page 2560

\section*{Command 'Download Manager'}

Symbol: 匋
Function: Download or create a boot project from the project devices or update the firmware oft he device.
Call: Main menu "Online", Context menu.
Requirement: A project is open.

\section*{Command 'Multiple Download'}

Function: The command causes the code generation of the applications contained in the project as well as the loading of the applications to the corresponding controllers.

Call: Menu bar: "Online"

The command opens a dialog with a list of the applications. In this dialog, select the applications that are to be loaded. Then, CODESYS performs the syntax check of these applications and generates the respective code. The code is then downloaded to the respective PLC. For each selected application, CODESYS generates a build log with the name <project name>.<device name>.<application ID>.compileinfo in the project directory.


For more information, see the "Subordinate Safety Controllers" chapter.

Dialog 'Multiple
Download'
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
"Please select the items to be \\
downloaded"
\end{tabular} & \begin{tabular}{l}
\(\boxed{\swarrow}\) : Selection of the applications. The applications are thereby also loaded to \\
different controllers.
\end{tabular} \\
\hline "Move Up", "Move Down" & \begin{tabular}{l} 
Change of the order of download of the applications. \\
The applications are downloaded to the PLCs in the order of this list. By default, \\
this list is alphabetically sorted. Parent-child relationships of applications are \\
thereby taken into account.
\end{tabular} \\
\hline "OK" & \begin{tabular}{l} 
Checks the syntax of all selected applications. Afterwards, the communication \\
with the associated controller is verified for each application before the download \\
takes place.
\end{tabular} \\
\hline
\end{tabular}

Table 495: "Online Change Options"
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
If an earlier version already exists on the PLC and is different from the current version, then the following options \\
are provided:
\end{tabular} \\
\hline \begin{tabular}{l} 
"Try to perform an online \\
change. If this is not possible, \\
perform a full download."
\end{tabular} & \begin{tabular}{l} 
Activated by default. If an online change cannot be executed for one of the \\
applications, then a download is performed.
\end{tabular} \\
\hline \begin{tabular}{l} 
"Force an online change. If \\
this is not possible, cancel the \\
operation."
\end{tabular} & \begin{tabular}{l} 
If an online change cannot be performed for (at least) one of the applica- \\
tions, then no download is performed and the online change is terminated (for \\
example, if you have executed the command "Clean All" beforehand).
\end{tabular} \\
\hline \begin{tabular}{l} 
"Always perform a full \\
download."
\end{tabular} & \begin{tabular}{l} 
Downloads all parts of the applications to the PLC, regardless of any existing \\
versions.
\end{tabular} \\
\hline
\end{tabular}

For selected applications that do not exist on the PLC yet, CODESYS performs a download automatically to the PLC.

Table 496: "Other Options"
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
"Delete all applications on the \\
PLC which are not part of the \\
project."
\end{tabular} & \(\boxed{\square}\) : Corresponding applications are deleted \\
\hline \begin{tabular}{l} 
"Start all applications after \\
download or online change"
\end{tabular} & \(\boxed{\square}\) : The applications are started after the download or online change. \\
\hline \begin{tabular}{l} 
"Do not release forced \\
variables"
\end{tabular} & \begin{tabular}{l} 
implementation of this application has been changed, then no download is per- \\
formed for this application. \\
The message "Error: Skipped because one or more variables have been forced" \\
appears for this application in the window "Multiple Download - Result".
\end{tabular} \\
\hline
\end{tabular}

Note that variables with the key attribute PERSISTENT RETAIN are not generally initialized. If you change the data layout, however, the persistent variables are automatically re-initialized.

After completion of the download a listing of all selected applications appears in the download order that you configured. In addition, you are shown information on the success of the download for each application in the "Multiple Download - Result" dialog:
- "Created": A new application has been created and downloaded to the controller.
- "Not changed": The application which exists on the controller has not been changed.
- "Online changed": The application which exists on the controller has been modified by an online change.
- "Downloaded": The application which exists on the controller has been replaced by a new created application.
- "Skipped due to impossible online change": An online change could not be performed for the application. The application was not changed.
- "Error": An error has occurred for this application during download. More details may be displayed.
- "Cancelled by user": The operation has been aborted by the user.

See also
- ② Chapter 6.4.1.10.7 "Subordinate safety controller" on page 1964
- \(\xi^{*}\) Chapter 6.4.1.9.19 "Data Persistence" on page 1920
- \(\boldsymbol{y}^{4}\) Chapter 6.4.1.11.5 "Generating Application Code" on page 1976
- ※y Chapter 6.4.1.9.19 "Data Persistence" on page 1920
- \(\quad \Leftrightarrow\) Chapter 6.4.1.21.3.7.5 "Command 'Load'" on page 2628

\section*{Command 'Reset Cold'}

Function: The command results in a cold start of the active application on the controller.
Call: Menu bar: "Online"
Requirement: The application is in online mode.

\section*{NOTICE!}

If a safety controller is inserted below a controller, then this command can interrupt the communication connections temporarily.
Connections of the safety controller to other safety controllers (via safety network variables), to field devices, and to the development system are affected. The safe field devices or other safety controller can enter the safe state as a reaction. The connection to the development system is affected only when a safety controller that is connected to the main controller via a fieldbus.

For more information, see the "Subordinate Safety Controllers" chapter.

After restarting with "Reset Cold", the following happens:
- Application code is retained on the controller.
- Variables are initialized (with the initialization value or the default initialization value 0 ), and the previous values are lost.
- Retain variables are initialized, and the previous values are lost.
- Persistent variables are retained with values.
- Breakpoints that were set in the code are retained with their status (for example, activated or deactivated).
- The application goes into the "STOP" state.

You can also select the command while debugging the application when it halts at a breakpoint in the "HALT ON BP" state. Then either the warm start is executed immediately, or the remaining statements of the current cycle are processed. Therefore, a message window opens for you to select the next action. However, the message window opens only if the runtime system is capable of restarting the cycle without terminating it first.

After the reset, you can run the application as usual and, for example, start the execution by clicking "Debug \(\rightarrow\) Start".

See also
- \# Chapter 6.4.1.12.6 "Resetting applications" on page 1990
- Chapter 6.4.1.9.19.2 "Preserving data with persistent variables" on page 1923
- \#Chapter 6.4.1.12.3 "Using Breakpoints" on page 1981
- \& Chapter 6.4.1.20.2.13 "Persistent Variable - PERSISTENT" on page 2122
- \& Chapter 6.4.1.10.7 "Subordinate safety controller" on page 1964
- \& Chapter 6.4.1.21.3.7.11 "Command 'Reset Warm"' on page 2634
- \(\triangleq\) Chapter 6.4.1.21.3.7.12 "Command 'Reset Origin"' on page 2635

\section*{Command 'Reset Warm'}

Function: The command results in a warm start of the active application on the controller.
Call: Menu bar: "Online"
Requirement: The application is in online mode.

\section*{NOTICE!}

If a safety controller is inserted below a controller, then this command can interrupt the communication connections temporarily.
Connections of the safety controller to other safety controllers (via safety network variables), to field devices, and to the development system are affected. The safe field devices or other safety controller can enter the safe state as a reaction. The connection to the development system is affected only when a safety controller that is connected to the main controller via a fieldbus.

For more information, see the "Subordinate Safety Controllers" chapter.

After restarting with "Reset Warm", the following happens:
- Application code remains loaded on the controller.
- Variables are initialized (with the initialization value or the default initialization value 0 ).
- Retain variables are retained with values.
- Persistent variables are retained with values.
- Breakpoints that were set in the code are retained with their status (for example, activated or deactivated).
- The application goes into the "STOP" state.

You can also select the command while debugging the application when it halts at a breakpoint in the "HALT ON BP" state. Then either the warm start is executed immediately, or the remaining statements of the current cycle are processed. Therefore, a message window opens for you to select the next action. However, the message window opens only if the runtime system is capable of restarting the cycle without terminating it first.
After the reset, you can run the application as usual and, for example, start the execution by clicking "Debug \(\rightarrow\) Start".

See also
- \(\quad\) ² Chapter 6.4.1.12.6 "Resetting applications" on page 1990
- \(\Rightarrow\) Chapter 6.4.1.9.19.2 "Preserving data with persistent variables" on page 1923
- \(\quad 4\) Chapter 6.4.1.12.3 "Using Breakpoints" on page 1981
- \({ }^{\bullet}>\) Chapter 6.4.1.20.2.13 "Persistent Variable - PERSISTENT" on page 2122
- \(\quad\), \({ }^{2}\) Chapter 6.4.1.10.7 "Subordinate safety controller" on page 1964
- 乡y Chapter 6.4.1.21.3.7.10 "Command 'Reset Cold'" on page 2633
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.7.12 "Command 'Reset Origin"" on page 2635

\section*{Command 'Reset Origin'}

Function: The command results in a reset origin of the active application on the controller.
Call: Menu bar: "Online"
Requirement: The application is in online mode.

\section*{NOTICE!}

If a safety controller is inserted below a controller, then this command can interrupt the communication connections permanently.
Connections of the safety controller to other safety controllers (via safety network variables), to field devices, and to the development system are affected. The safe field devices or other safety controller can enter the safe state as a reaction. The connection to the development system is affected only when a safety controller that is connected to the main controller via a fieldbus.

For more information, see the "Subordinate Safety Controllers" chapter.

After restarting with "Reset Origin", the following happens:
- The application code is deleted, and as a result the application has no state.
- Variables are deleted, and the values are lost.
- Retain variables are deleted, and the values are lost.
- Persistent variables are deleted, and the values are lost.
- Breakpoints that were set in the code are lost.

See also
- \(\#\) Chapter 6.4.1.12.6 "Resetting applications" on page 1990
- © Chapter 6.4.1.9.19.2 "Preserving data with persistent variables" on page 1923
- \& Chapter 6.4.1.12.3 "Using Breakpoints" on page 1981
- \& Chapter 6.4.1.20.2.13 "Persistent Variable - PERSISTENT" on page 2122
- \& Chapter 6.4.1.10.7 "Subordinate safety controller" on page 1964
- \(\#\) Chapter 6.4.1.21.3.7.11 "Command 'Reset Warm'" on page 2634
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.7.10 "Command 'Reset Cold'" on page 2633

\section*{Command 'Reset Origin Device'}

Function: The command opens a dialog to reset the device to its factory settings. All applications, boot applications, and remanent variables will be deleted from the device. Depending on the version of the device, a selection of the elements to be deleted can be made in this dialog. When these elements are unselected in the dialog, they are not deleted during the reset and remain on the controller. By default, all elements are selected and everything is deleted. Elements that are not available for selection are generally also deleted.
Call: Right-click a programmable device in the device tree.

\section*{NOTICE!}

If a safety controller is inserted below a controller, then this command can interrupt the communication connections permanently.
Connections of the safety controller to other safety controllers (via safety network variables), to field devices, and to the development system are affected. The safe field devices or other safety controller can enter the safe state as a reaction. The connection to the development system is affected only when a safety controller that is connected to the main controller via a fieldbus.
For more information, see the "Subordinate Safety Controllers" chapter.

After restarting with "Reset Origin Device", the following happens:
- All applications are reset as with the "Reset Origin" command.
- All files, which are not deleted by the "Reset Origin" command, are deleted (for example, files from visualization, alarms, and recipes).
- The user management is deleted.
- All certificates which are currently managed by the runtime system are deleted.

Note: When resetting the device, the objects selected in this dialog are also deleted. If not all displayed objects are selected in this dialog, then possibly other objects can no longer be used or they are also deleted. .
\begin{tabular}{|c|c|}
\hline "Delete" & \(\checkmark\) : The object is deleted when the "Reset Origin Device" command is executed. \\
\hline "Object" & \begin{tabular}{l}
Objects that can be excluded from "Delete". \\
The listed objects depend on the version of the controller. In version 3.5.16.20 and higher, the following objects can be excluded from the delete operation. \\
- "User Management" \\
- "PLC Logic" \\
- "Certificates"
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{\mu}\) Chapter 6.4.1.21.3.7.12 "Command 'Reset Origin"' on page 2635
- \({ }^{*} \boldsymbol{y}\) Chapter 6.4.1.21.3.7.11 "Command 'Reset Warm"' on page 2634
- ⓨ Chapter 6.4.1.21.3.7.10 "Command 'Reset Cold'" on page 2633
- ثy Chapter 6.4.1.10.7 "Subordinate safety controller" on page 1964

\section*{Command 'Logoff Current Device User'}

Symbol: ©
Function: This command logs out the user currently logged in to the controller (device). If CODESYS still has a connection to the controller, then it will be disconnected.

Call: Main menu "Online".
Requirement: The application is in online mode.

You can manage the device user management in the "Users and Groups"
tab and "Access control" of the device editor. The commands in the "Online
\(\rightarrow\) Security" menu provide another simple option for protecting access to the target device.

See also
- ② Chapter 6.4.1.21.2.8.14 "Tab 'Users and Groups'" on page 2450
- \({ }^{*}\) Chapter 6.4.1.11.4 "Handling of Device User Management" on page 1971

\section*{Command 'Download'}

Function: This command loads the compiled project in the PLC.
Call: Main menu "Online", Context menu.
Requirement: A project is open. Log-in required for download.

\section*{Command 'Add Device User'}

Symbol:
Function: This command configures a new device user and adds this user to the administrator group.

\section*{Call: Menu bar: "Online \(\rightarrow\) Security"}

Requirement: The device supports a device user management. You are logged in to the device as a user.

You can manage the device user management in the "Users and Groups" tab and "Access control" of the device editor. The commands in the "Online \(\rightarrow\) Security" menu provide another simple option for protecting access to the target device.

See also
- \({ }^{*} \Rightarrow\) Chapter 6.4.1.21.2.8.14 "Tab 'Users and Groups'" on page 2450
- \(\quad 4\) Chapter 6.4.1.11.4 "Handling of Device User Management" on page 1971

This command opens the "Add Device User" dialog. Here you define the access data of the new user.
The dialog corresponds to the dialog in the "Users and Groups" tab of the device editor for adding a new user.

Please use a strong password as follows:
- Password length >= 8 characters (best >=12)
- Use uppercase and lowercase
- Include numbers
- Use special characters
- Do not use existing names or sequence of characters that are easy to guess (for example, "123", "abc", "qwerty")

\section*{CAUTION!}

After performing this action, you can no longer use a blank username and password to log in. You must remember your password.

See also
- \({ }^{\circ} \gg\) Chapter 6.4.1.21.3.7.17 "Command 'Remove Device User"' on page 2638
- \#y Chapter 6.4.1.21.3.7.18 "Command 'Change Password Device User'" on page 2639

\section*{Command 'Remove Device User'}

Symbol:
Function: This command removes a user from the user management on the target system (device).
Call: Menu bar: "Online \(\rightarrow\) Security"
Requirement: You are logged in to the device as a user.

You can manage the device user management in the "Users and Groups"
tab and "Access control" of the device editor. The commands in the "Online \(\rightarrow\) Security" menu provide another simple option for protecting access to the target device.

See also
- \({ }^{4}\), Chapter 6.4.1.21.2.8.14 "Tab 'Users and Groups'" on page 2450
- \(\Rightarrow\) Chapter 6.4.1.11.4 "Handling of Device User Management" on page 1971

This command opens the "Remove Device User" dialog. Specify the user name and password of the user to be removed and click "OK" to confirm.

\section*{CAUTION!}

After performing this action, you can no longer use this removed user account to log in. If this user is the only one on the target system, then a dialog prompt notifies you that this user cannot be removed.

See also
- \(\sum^{\wedge}\) Chapter 6.4.1.21.3.7.16 "Command 'Add Device User'" on page 2637
- \(\boldsymbol{y y}^{2}\) Chapter 6.4.1.21.3.7.18 "Command 'Change Password Device User'" on page 2639

\section*{Command 'Change Password Device User'}

Symbol:
Function: The command changes the password for the user who is currently logged on the PLC.

Call: "Online \(\rightarrow\) Security" menu
Requirement: You are logged in to the device as a user.
The command opens the "Change Password for Device User" dialog for defining a new password. You have to specify the old password again.

\section*{NOTICE!}

After performing this action, you can no longer use the previous password to log in.

Make sure that you use a strong password. Note the following:
- Password length >= 8 characters (best >=12)
- Use uppercase and lowercase
- Include numbers
- Use special characters
- Do not use existing names or sequence of characters that are easy to guess (for example, "123", "abc", "qwerty")
See also
- \({ }^{\text {® }}\) Chapter 6.4.1.21.3.7.16 "Command 'Add Device User'" on page 2637
- \(\quad\), \(\quad\) Chapter 6.4.1.21.3.7.17 "Command 'Remove Device User'" on page 2638
- \(\Leftrightarrow\) Chapter 6.4.1.11.4 "Handling of Device User Management" on page 1971

\section*{Command 'Stop Execution on Handled Exceptions'}

Function: This command halts the application where the error is located despite a programmed exception handling.
Call: This command is not available by default, but it can be configured from the "Tools
\(\rightarrow\) Customize", "Add Command" dialog box ("Online" category).

Requirement: The application is in online mode and contains a programmed exception handling with the __TRY and __CATCH operators.
If you have configured this command from the "Online" menu and you call it from there, then the currently active application is affected. Furthermore, this command can help you to detect errors.
See also
- \({ }^{\#} \boldsymbol{y}\) Chapter 6.4.1.20.3.62 "Operators '__TRY', '__CATCH', '__FINALLY', '__ENDTRY"" on page 2206
- « "Adding commands" on page 1803
- \({ }^{\text {• }}\), Chapter 6.4.1.21.3.9.16 "Command 'Customize"' on page 2667

\section*{Command 'Connect to Device'}

Function: The command establishes a connection to the device currently selected in the device tree.
Call: Context menu of the device.
Requirements: A device is selected in the device tree. The communication settings are configured correctly.
See also
- « Chapter 6.4.1.21.3.7.21 "Command 'Disconnect from Device'" on page 2640

\section*{Command 'Disconnect from Device'}

Function: The command disconnects the connection from a device.
Call: Context menu of the device.
Requirements: A device is selected in the device tree.
See also
- \({ }^{*}\) Chapter 6.4.1.21.3.7.20 "Command 'Connect to Device"' on page 2640

\section*{Command 'Wink'}

Symbol:
Function: The command causes an LED of a connected controller to blink. As a result, the hardware can be identified clearly.

Call: The command is not in any menu by default. You can add it to a menu by means of the
"Tools \(\rightarrow\) Customize" dialog, in the "Online" command category.
Requirement: The controller supports this function and the connection parameters are configured correctly.

\section*{Command 'Simulation}

Function: The command switches the development system to simulation mode. Call: Menu bar: "Online"

In simulation mode, you can start and debug the active application an on a simulated target device. A physical target device is not necessary for testing the online behavior of an application. When logging in for the first time, you are prompted whether the application should be created or loaded. For a simulated device, you do not have to configure the communication settings. In CODESYS simulation mode, the entry of the controller in the device tree is displayed in italics.

\section*{NOTICE!}

\section*{No C code for simulation mode}

In simulation mode, C code is not generated and loaded to the runtime system. To simulate the code contained in the C modules anyway, you can implement it for this purpose in the respective IEC objects of the C code module.

After successful login, the red triangle symbol \((\boldsymbol{\Delta})\) in the device tree indicates simulation mode. You can use the corresponding online commands for testing the application.

To switch off simulation mode, log out of the controller and execute the "Simulation" command again.

The command affects the active application only.

Differences between simulation mode and operation with a physical controller
\begin{tabular}{|c|c|c|}
\hline & Simulation & Physical Controller \\
\hline Real-time behavior / multicore & \begin{tabular}{l}
- Runs in the CODESYS process with normal priority \\
- Single-core --> Worse real-time behavior
\end{tabular} & \begin{tabular}{l}
- Real-time operating system \\
- Single-core or multicore
\end{tabular} \\
\hline Architecture scope & - Simulation 64-bit (depends on the CODESYS installation): --> Possible compile error in the IEC application if the application has been previously run only as 32bit (for example, use of DWORD as POINTER) & - Controller 32-bit \\
\hline FPU (rounding error) & \begin{tabular}{l}
- Uses FPU of the PC \\
- Different configuration of the FPU exceptions
\end{tabular} & \begin{tabular}{l}
- Uses FPU of the controller or FPU emulation \\
- Different configuration of the FPU exceptions
\end{tabular} \\
\hline Handling of exceptions & - Exception handling of the Windows Runtime System & - Exception handling of the controller \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|}
\hline & Simulation & Physical Controller \\
\hline External libraries (Cmp/Sys/CAA/O EM/...) & \begin{tabular}{l}
- Only a few external Cmp/ SysLibs are physically available. As compared to embedded, more SysLibs could also be available. \\
- Other implementation/behavior of the SysLibs (Windows in contrast to the OS of the controller) \\
- "Unresolved Reference error" on download is ignored. The application can still be downloaded to the controller and started. If the missing functions are actually called, they return nonsense values. \\
Therefore, an IEC implementation can also be specified for external POUs. This substitute IEC code is then executed in the simulation.
\end{tabular} & - "Unresolved Reference error" on download when external libraries do not exist in the controller \\
\hline I/O drivers & \begin{tabular}{l}
- I/O configuration is generated but not evaluated. \\
- Fieldbus stacks are not evaluated. \\
- I/O channels are not updated and no bus telegrams are sent.
\end{tabular} & - Mostly no restriction, but depends on the capabilities of the controller \\
\hline SoftMotion drivers & - All SoftMotion axes are set to virtual and therefore simulated. & - Mostly no restriction, but depends on the capabilities of the controller \\
\hline
\end{tabular}

See also
- \({ }^{4}\) Chapter 6.4.1.12.2 "Testing in simulation mode" on page 1980

\section*{Command 'Operating Mode'}

Function: The commands set the controller to a state which prevents accidental change to the project.
Call: Menu bar: "Online \(\rightarrow\) Operating Mode"
You can use these commands, for example, to lock the state of a controller in order to prevent the controller from switching to another state while you program another controller.

When programming is complete, the controller should then be switched to a defined and externally visible state that is set exactly the same way after restarting.
The \%, 日, and \(\mathbf{8}\) symbols in the status bar indicate the current operating mode. Doubleclicking one of these symbols opens a help window.

If it supports the controller, then you can switch the controller to the following operating modes:
- Z "Debug": No restrictions
- 日 "Locked": The current state of debugging is locked on the application. No additional breakpoints can be set and no additional variables can be forced. Writing variables is still possible and breakpoints which are already set remain active.
Only the "RUN" state of an application is preserved in \(\mathbf{6}\) "Locked" operating mode even if the controller is restarted.
With this operating mode, a developer can prevent himself or another developer from changing the application on the controller, for example by setting or deleting a breakpoint, by forcing, or by making changes to the file system. This operating mode is helpful to prevent a download to an incorrect controller when, for example, multiple controllers of a plant are programmed.
- 日 "Operational":

This operating mode makes sure that the controller reloads the same applications after a restart and that no debug features are active anymore. The operating mode is set when a controller is completely programmed and should be accepted or already is.
Conditions for activating the "Operational" mode
- A boot application for each application has to exist on the controller.
- There must not be any active breakpoints set.
- All applications have to be running.
- There must not exist any forced values.
- Furthermore, the device can define more of its own restrictions.

The "Locked" and "Operational" operating modes are different in the use cases and in the requirements for activating the operating mode. However, for both operating modes the runtime system prevents the following actions:
- Regarding the application
- Download of an application
- Online change
- Force variables
- Set breakpoints
- Stop application
- Reset application
- Start application
- Delete application
- Regarding the file transfer of the controller
- Download of a file to the controller
- Delete a file on the controller
- Rename a file on the controller
- Create a directory on the controller
- Delete a directory on the controller
- Rename a directory on the controller


You cannot switch the operating mode between "Locked" and "Operational".

\section*{Command 'Virtual mode'}

Function: "Virtual Mode" option enables virtual mode for Automation Builder.
Call: Main menu "Online", Context menue
Requirement: A project is open. Command is only available, if a license for advanced simulation support is acitvated.

\section*{Command 'Virtual system testing'}

Function: The "Virtual system testing" editor contains settings for the virtual devices and the simulation set-up and control.

Call: Main menu "Online", Context menu
Requirement: A project is open. Command is only available, if a license for advanced simulation support is acitvated.

\section*{Menu 'Debug'}
6.4.1.21.3.8.1 Command 'Start' ..... 2644
6.4.1.21.3.8.2 Command 'Stop'. ..... 2645
6.4.1.21.3.8.3 Command 'Single Cycle' ..... 2645
6.4.1.21.3.8.4 Command 'New Breakpoint'. ..... 2645
6.4.1.21.3.8.5 Command 'New Data Breakpoint' ..... 2645
6.4.1.21.3.8.6 Command 'Edit Breakpoint'... ..... 2645
6.4.1.21.3.8.7 Command 'Enable Breakpoint' ..... 2646
6.4.1.21.3.8.8 Command 'Disable Breakpoint'. ..... 2646
6.4.1.21.3.8.9 Command 'Toggle Breakpoint' ..... 2646
6.4.1.21.3.8.10 Command 'Step Over'. ..... 2646
6.4.1.21.3.8.11 Command 'Step Into'...... ..... 2647
6.4.1.21.3.8.12 Command 'Step Out' ..... 2647
6.4.1.21.3.8.13 Command 'Run to Cursor' ..... 2648
6.4.1.21.3.8.14 Command 'Set Next Statement' ..... 2648
6.4.1.21.3.8.15 Command 'Show Next Statement'. ..... 2648
6.4.1.21.3.8.16 Command 'Force Values'. ..... 2649
6.4.1.21.3.8.17 Command 'Write Values' ..... 2649
6.4.1.21.3.8.18 Command 'Unforce Values' ..... 2650
6.4.1.21.3.8.19 Command 'Force All Values from <Device.Application>' ..... 2650
6.4.1.21.3.8.20 Command 'Write All Values from <Device.Application>' ..... 2651
6.4.1.21.3.8.21 Command 'Unforce All Values from <Device.Application>'. ..... 2652
6.4.1.21.3.8.22 Command 'Flow Control'. ..... 2652
6.4.1.21.3.8.23 Menu 'Core Dump' ..... 2653
6.4.1.21.3.8.24 Command 'Display Mode' - 'Binary', 'Decimal', 'Hexadeci- mal'. ..... 2654

\section*{Command 'Start'}

Symbol: ; keyboard shortcut: [F5]
Function: This command starts the application (status: "RUN").
Call: Menu bar: "Debug"; context menu of object: "Application"
Requirement: The application is in online mode and its status is "STOP".
Executing this command from the "Debug" menu will affect the application that is currently in focus.

\section*{See also}
- \(\Rightarrow\) Chapter 6.4.1.11.6 "Downloading the application code, logging in, and starting the PLC" on page 1977

\section*{Command＇Stop＇}

Symbol：■；keyboard shortcut：［Shift］＋［F8］
Function：This command stops the application（status：＂STOP＂）．
Call：Menu bar：＂Debug＂；context menu of object：＂Application＂
Requirement：The application is in offline mode and its status is＂RUN＂．
Executing this command from the＂Debug＂menu will affect the application that is currently in focus．

\section*{Command＇Single Cycle＇}

Keyboard shortcut［Ctrl］＋［F5］
Function：This command executes the active application for one cycle．
Call：Main menu＂Debug＂．
Requirement：The application is in online mode and the program is halted at a program step．

\section*{Command＇New Breakpoint＇}

Symbol：踢，keyboard shortcut［AIt］＋［F7］．
Function：This command opens the＂Breakpoint Properties＂dialog box．
Call：Main menu＂Debug＂．
Requirement：The application must be in online mode．


See also
－\({ }^{4}\) Chapter 6．4．1．12．3＂Using Breakpoints＂on page 1981

\section*{Command＇New Data Breakpoint＇}

Symbol：很
Function：The command opens the＂New breakpoint＂dialog．
Call：Menu bar：＂Debug＂

\section*{Requirement：}
－The application is in online mode．
－The device description file of the target device contains the entries for the＂data breakpoints＂ functionality．Currently，data breakpoints are possible only with the CODESYS Control Win V3．

See also
－\({ }^{\text {n }}\) Chapter 6．4．1．21．4．9＂Dialog＇New Breakpoint＂＇＂on page 2750
－\(\quad 4\) Chapter 6．4．1．12．3＂Using Breakpoints＂on page 1981

\section*{Command＇Edit Breakpoint＇}
```

Symbol: 覀

```

Function: This command opens the "Breakpoint Properties" dialog box.
Call: Main menu "Debug".
Requirement: The application is in online mode and the cursor is halted at a breakpoint.
See also
- \({ }^{4}\) Chapter 6.4.1.12.3 "Using Breakpoints" on page 1981

\section*{Command 'Enable Breakpoint'}

Function: This command enables a disabled breakpoint.
Call: Main menu "Debug".
Requirement: The application is in online mode and the cursor is halted at a disabled breakpoint.

See also
- \({ }^{4}\) Chapter 6.4.1.12.3 "Using Breakpoints" on page 1981

\section*{Command 'Disable Breakpoint'}

Function: This command disables an enabled breakpoint.
Call: Main menu "Debug".
Requirement: The application is in online mode and the cursor is halted at an enabled breakpoint.
See also
- ② Chapter 6.4.1.12.3 "Using Breakpoints" on page 1981

\section*{Command 'Toggle Breakpoint'}

Keyboard shortcut [F9]
Function: This command sets a breakpoint or clears an existing breakpoint.
Call: Main menu "Debug".
Requirement: The application is in online mode. The cursor is positioned at a breakpoint.
See also
- \({ }^{*}\) Chapter 6.4.1.12.3 "Using Breakpoints" on page 1981

\section*{Command 'Step Over'}

Symbol [注, shortcut [F10]
Function: The command executes the statement where the program is currently located and halts before the next statement in the POU.
Call: Menu bar: "Debug"
Requirement: The application is in online mode and the program is halted at the current break position (debug mode).

If the executed statement contains a call (from a program, function block instance, function, method, or action), then the subordinate POU is processed completely in one step and returned to the call. Then it halts before the next statement (in the next line of code).

Click "Step Into" to jump to a subordinate POU and process it in single steps.

See also
- \({ }^{\text {• }}\) Chapter 6.4.1.12.4 "Stepping Through a Program" on page 1985
- \(\quad \stackrel{y}{c}\) Chapter 6.4.1.21.3.8.11 "Command 'Step Into'" on page 2647

\section*{Command 'Step Into'}

Symbol \({ }^{\ddagger}\) 三, shortcut [F11]
Function: The command executes the statement where the program is currently located and halts before the next statement.
Call: Menu bar: "Debug"
Requirement: The application is in online mode and the program is halted at the current break position (debug mode).
If the executed statement contains a call (from a program, function block instance, function, method, or action), then the program execution jumps to this subordinate POU. Its code opens in a separate editor. The first statement there is executed and the program execution halts before the next statement. The new current breakpoint position is then in the called POU.

Click "Step Over" to remain in the currently active POU and execute the call in one step.

See also
- Es Chapter 6.4.1.12.4 "Stepping Through a Program" on page 1985
- \({ }^{4}\), Chapter 6.4.1.21.3.8.10 "Command 'Step Over"" on page 2646

\section*{Command 'Step Out'}

Symbol \({ }_{\text {E }}^{\text {三, }}\), shortcut [Ctrl]+[F11]
Function: The command executes the program until the next return and halts afterwards.
Call: Menu bar: "Debug"
Requirement: The application is in online mode and the program is halted at the current break position (debug mode).

If the current breakpoint position is in a subordinate POU, then this is run through to the end. Then the program execution jumps back to the calling point in the calling POU and halts there (in the line with the call).

If the current breakpoint position is in the main program, then the POU is run through to the end. Then the program execution jumps back to the beginning (to the program start at the first line of code in the POU) and halts there.

See also
- \({ }^{*}\) ) Chapter 6.4.1.12.4 "Stepping Through a Program" on page 1985

\section*{Command 'Run to Cursor'}

Symbol: \(+\overline{\text { ㅌ }}\)
Function: The command executes a program until a specified position as marked by the cursor.
Call: Menu bar: "Debug"
Requirement: The application is in online mode and the program is halted at the current break position (debug mode). Moreover, you have marked any line of code in any POU with the cursor.

The statements between the current breakpoint position and the cursor position are executed in one step. Then the execution halts at the cursor position, which then becomes the next breakpoint position. Remember that the line of code where you placed the cursor is reached but not executed.

See also
- \(\sum^{*}\) Chapter 6.4.1.12.4 "Stepping Through a Program" on page 1985

\section*{Command 'Set Next Statement'}

Symbol: \&
Function: The command determines which statement is executed next.
Call: Menu bar: "Debug"
Requirement: The application is in online mode and the program is halted at the current break position (debug mode). Moreover, you have marked any line of code in any POU with the cursor.

The line of code marked with the cursor becomes the current breakpoint position without executing the statements in between or the statement that jumped to it.

See also
- \(\sum^{4}\) Chapter 6.4.1.12.4 "Stepping Through a Program" on page 1985

\section*{Command 'Show Next Statement'}

Symbol: \(\Rightarrow\)
Function: The command displays the program statement that is processed in the next step.
Call: Menu bar: "Debug"
Requirement: The application is in online mode and the program is halted at the current break position (debug mode). The break position is in a line of code that you cannot see.
The command makes the window with the current breakpoint position active (in the code highlighted in yellow and marked with the \(\Rightarrow\) symbol) and makes the breakpoint position to become visible. This is useful if you have multiple editors open and the breakpoint position is hidden in an inactive editor.

See also
- \({ }^{*}\) ) Chapter 6.4.1.12.4 "Stepping Through a Program" on page 1985

\section*{Command 'Force Values'}

Keyboard shortcut: [F7]
Function: The command sets a permanent predefined value to a variable on the controller.
Call: Menu bar: "Debug"
Requirement: The application is in online mode.


\section*{CAUTION!}

Unusual changes to variable values in an application currently running on the controller can lead to undesired behavior of the controlled machinery.
Evaluate possible dangers before forcing variable values. Take the respective safety precautions. Depending on the controlled machinery, the result may lead to damage to machinery and equipment or injury to health and life of personnel.

With this command, CODESYS permanently sets one or more variables of the active application to defined values on the PLC.

A forced value is marked with the forced symbol ( \(\boldsymbol{E}\) ).
For more information about the functionality of forcing and the \(p\) of values, see the "Forcing and Writing of Variables" help page.

By default, the "Force Values [All Applications]" command, which applies to all application in the project, and is not included in a menu.

See also
- \({ }^{\text {• }}\) Chapter 6.4.1.12.5 "Forcing and Writing of Variables" on page 1987
- \({ }^{\bullet} \gg\) Chapter 6.4.1.21.4.8 "Dialog Box 'Prepare Value"' on page 2749
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.8.18 "Command 'Unforce Values"' on page 2650

\section*{Command 'Write Values'}

Keyboard shortcut [Ctrl]-[F7]
Function: This command sets a predefined value to a variable on the controller one time.
Call: Main menu "Debug".
Requirement: The application is in online mode.


\section*{CAUTION!}

Unusual changes to variable values in an application currently running on the controller can lead to undesired behavior of the controlled machinery.
Evaluate possible dangers before forcing variable values. Take the respective safety precautions. Depending on the controlled machinery, the result may lead to damage to machinery and equipment or injury to health and life of personnel.

With this command, one or more variables of the active application are set to defined values on the controller one time. Writing is done one time at the beginning of the next cycle.

Values are prepared by
- Clicking in the field "Prepared value" in the declaration section
- Clicking in the inline monitoring field in the implementation section
- Clicking in the field "Prepared value" in the watch window

The command "Write Values [All Applications]" affects all application in the project and is not included in a menu by default.

See also
- \& Chapter 6.4.1.21.3.8.16 "Command 'Force Values"' on page 2649
- «y Chapter 6.4.1.12.5 "Forcing and Writing of Variables" on page 1987

\section*{Command 'Unforce Values'}

Keyboard shortcut [Alt]+[F7]
Function: This command resets the forcing of all variables. The variables receive their current values from the PLC.
Call: "Debug".
Requirement: The application is in online mode.
The "Remove Force List" command has the same functionality as this command with one difference. If the "Remove Force List" command cannot be executed for all forced values, then no message is displayed.


\section*{CAUTION!}

Unusual changes to variable values in an application currently running on the PLC can lead to undesired behavior of the controlled machinery.

Evaluate possible dangers before forcing variable values. Take the respective safety precautions. Depending on the controlled system, the result may lead to damage to machinery and equipment or injury to health and life of personnel.

The command "Force Values [All Applications]" affects all application in the project and is not included in a menu by default.

See also
- © Chapter 6.4.1.21.3.8.16 "Command 'Force Values"' on page 2649
- « Chapter 6.4.1.12.5 "Forcing and Writing of Variables" on page 1987

\section*{Command 'Force All Values from <Device.Application>'}

Function: This command resets all values of variables from the selection <Device.Application> to predefined values permanently.

\section*{Call:}
- Context menu of the application in the device tree
- Context menu in the editor of a POU from the selected application

Requirement: The application is in online mode.

CAUTION!
Unusual changes to variable values in an application currently running on the controller can lead to undesired behavior of the controlled machinery.
Evaluate possible dangers before forcing variable values. Take the respective safety precautions. Depending on the controlled machinery, the result may lead to damage to machinery and equipment or injury to health and life of personnel.

With this command, CODESYS permanently sets one or more variables of the active application to defined values on the PLC. This is done at the beginning and end of a processing cycle. Order of processing: 1) read inputs, 2) force values, 3) process code, 4) force values, 5) write outputs.

You can prepare values as follows:
- Click in the "Prepared value" field in the declaration part and type in the value. For Boolean variables, you change the value by clicking the field.
- Click in the inline monitoring field in the implementation part of the FBD/LD/IL editor
- Click in the "Prepared value" field in the monitoring view and type in the value.

A forced value is marked with the forced symbol (E).
CODESYS forces the value until you explicitly lift it by
- Clicking "Unforce Values"
- Clicking "Unforce All Values from <Device.Application>"
- Lifting the force in the "Prepare Value" dialog
- Logging out of the application


The command "Force Values [All Applications]" affects all application in the project and is not included in a menu by default.

See also
- \(\boldsymbol{y y}^{4}\) Chapter 6.4.1.21.4.8 "Dialog Box 'Prepare Value'" on page 2749
- \({ }^{\bullet}\) Chapter 6.4.1.21.3.8.18 "Command 'Unforce Values'" on page 2650
- \({ }^{4}\) Chapter 6.4.1.12.5 "Forcing and Writing of Variables" on page 1987
- ② Chapter 6.4.1.21.3.8.21 "Command 'Unforce All Values from <Device.Application>'" on page 2652

\section*{Command 'Write All Values from <Device.Application>'}

Function: This command resets all values of variables from the selection <Device.Application> to predefined values one time.

\section*{Call:}
- Context menu of the application in the device tree
- Context menu in the editor of a POU from the selected application

Requirement: The application is in online mode.

\section*{CAUTION!}

Unusual changes to variable values in an application currently running on the controller can lead to undesired behavior of the controlled machinery. Evaluate possible dangers before forcing variable values. Take the respective safety precautions. Depending on the controlled machinery, the result may lead to damage to machinery and equipment or injury to health and life of personnel.

With this command, one or more variables of the selected <Device.Application> are set to defined values on the PLC one time. Writing is done one time at the beginning of the next cycle.
You can prepare values as follows:
- Click in the "Prepared value" field in the declaration part and type in the value. For Boolean variables, you change the value by clicking the field.
- Click in the inline monitoring field in the implementation part of the FBD/LD/IL editor
- Click in the "Prepared value" field in the monitoring view and type in the value.

See also
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.8.17 "Command 'Write Values"" on page 2649
- \({ }^{\circ}\) Chapter 6.4.1.21.3.8.19 "Command 'Force All Values from <Device.Application>"" on page 2650
- ⓨ Chapter 6.4.1.12.5 "Forcing and Writing of Variables" on page 1987

\section*{Command 'Unforce All Values from <Device.Application>'}

Function: This command resets the forcing of all values of the variables from the selected <Device.Application>. The variables receive their current values from the PLC.

\section*{Call:}
- Context menu of the application in the device tree
- Context menu in the editor of a POU from the selected application

Requirement: The application is in online mode.

\section*{CAUTION!}

Unusual changes to variable values in an application currently running on the controller can lead to undesired behavior of the controlled machinery.
Evaluate possible dangers before forcing variable values. Take the respective safety precautions. Depending on the controlled machinery, the result may lead to damage to machinery and equipment or injury to health and life of personnel.

See also
- \({ }^{\circ}\) Chapter 6.4.1.21.3.8.19 "Command 'Force All Values from <Device.Application>"" on page 2650
- \({ }^{4}\), Chapter 6.4.1.21.3.8.18 "Command 'Unforce Values'" on page 2650

\section*{Command 'Flow Control'}

Function: This command activates and deactivates the flow control.
Call: Menu"Debug"
Requirement: The application is in online mode.

\section*{NOTICE!}

An active flow control extends application runtime.
When "Confirmed Online Mode" is activated in the communication settings, a dialog prompt appears when switching on the flow control to cancel the process.
When flow control is activated, it is not possible to use breakpoints or step through the program.

\section*{See also}
- Chapter 6.4.1.12.7 "Flow Control" on page 1992

\section*{Menu 'Core Dump'}
6.4.1.21.3.8.23.1 Command 'Load Core Dump'. ..... 2653
6.4.1.21.3.8.23.2 Command 'Create Core Dump' ..... 2653
6.4.1.21.3.8.23.3 Command 'Close Core Dump'. ..... 2654
6.4.1.21.3.8.23.4 Command 'Load Device Log from Core Dump'. ..... 2654

\section*{Command 'Load Core Dump'}

Function: CODESYS scans the project directory for core dump files. When a new core dump is forced with the "Create Core Dump" command, the dump file is automatically loaded from the controller to the project directory. If multiple core dump files are available, then CODESYS prompts you to choose whether the latest file should be opened in the project. You can also select one of the other files.
When a file is loaded into the project, an online view of the application appears with state of the application at the time when the core dump was generated. You can then view the variable values afterwards. Finally, the call tree is also available.
Call: Main menu "Debug \(\rightarrow\) Core Dump".
Requirement: The application is in offline mode.

\section*{NOTICE!}

You can close the core dump view only by clicking "Close Core Dump". The
"Logout" command has no effect in this view.

See also
- \({ }^{4}\) Chapter 6.4.1.21.3.8.23.2 "Command 'Create Core Dump'" on page 2653
- \({ }^{4}\), Chapter 6.4.1.21.3.8.23.3 "Command 'Close Core Dump'" on page 2654

\section*{Command 'Create Core Dump'}

Function: This command causes CODESYS to check whether a core dump file is already available on the controller.
If a core dump file is available, then CODESYS prompts you to load this file to the project directory.
With the following requirements, CODESYS generates a new dump file with the current application data:
- A core dump file is still not available or CODESYS has rejected a core dump file from being loaded.
- The application is currently stopped at breakpoint or an exception has occurred.

The generated core dump file is saved directly to the project directory: <project name>.<device name>.<application name>.<application Guid>.core. You can cancel the file generation by clicking the button in the status bar.
The amount of detail in the dump depends on the support from the runtime system. Runtime systems that are appropriate for this purpose generate just one dump in the case of an exception error. The core dump output from clicking "Load Core Dump" can therefore be used for error analysis.

Call: Main menu "Debug \(\rightarrow\) Core Dump".
Requirement: The application is in online mode.
See also
- \({ }^{*}\) Chapter 6.4.1.21.3.8.23.1 "Command 'Load Core Dump'" on page 2653

\section*{Command 'Close Core Dump'}

Function: This command closes the core dump view of the application that is open in the project.
Call: Main menu "Debug \(\rightarrow\) Core Dump".
Requirement: The application is in offline mode and you have loaded a core dump file to the project from the controller.

\section*{Command 'Load Device Log from Core Dump'}

Function: This command imports the controller log list that was saved with the last generated core dump. The log list is displayed in the same view as in online mode in the "Log" tab of the device editor.

Call: Main menu "Debug \(\boldsymbol{\rightarrow}\) Core Dump".
Requirement: The application is in offline mode and a core dump is open in the project.
See also
- \& Chapter 6.4.1.21.2.8.9 "Tab 'Log"' on page 2437

\section*{Command 'Display Mode' - 'Binary', 'Decimal', 'Hexadecimal'}

Function: These commands in the "Display Mode" submenu are used for setting the format of values in the display mode when monitoring in online mode.

Call: Main menu "Debug".
Requirement: The project is in either online or offline mode.


The "Binary" and "Hexadecimal" display modes are unsigned, and "Decimal" is signed.

See also
- Chapter 6.4.1.13.2.2 "Calling of monitoring in programming objects " on page 1996

\section*{Menu 'Tools'}
6.4.1.21.3.9.1 Command 'IP-Configuration' ..... 2655
6.4.1.21.3.9.2 Command 'Install additional licence' ..... 2655
6.4.1.21.3.9.3 Command 'Migrate third party devices'. ..... 2655
6.4.1.21.3.9.4 Command 'Package Manager', ..... 2655
6.4.1.21.3.9.5 Command 'Library Repository' ..... 2657
6.4.1.21.3.9.6 Command 'License Manager'. ..... 2659
6.4.1.21.3.9.7 Command 'License Repository' ..... 2662
6.4.1.21.3.9.8 Command 'Device Repository'. ..... 2663
6.4.1.21.3.9.9 Command 'Create Device list CSV' ..... 2665
6.4.1.21.3.9.10 Command 'Multi Online Change'. ..... 2665
6.4.1.21.3.9.11 Command 'Device ECAD data'. ..... 2665
6.4.1.21.3.9.12 Command 'OPC UA Information Model Repository' ..... 2665
6.4.1.21.3.9.13 Command 'Scripting' - 'Execute Script File'.... ..... 2666
6.4.1.21.3.9.14 Command 'Scripting' - 'Enable Script Tracing' ..... 2667
6.4.1.21.3.9.15 Command 'Scripting' - 'Scripts' ..... 2667
6.4.1.21.3.9.16 Command 'Customize'. ..... 2667
6.4.1.21.3.9.17 Command 'Options'. ..... 2667
6.4.1.21.3.9.18 Command 'Import and Export Options'. ..... 2668
6.4.1.21.3.9.19 Command 'Device Reader' ..... 2668

\section*{Command 'IP-Configuration'}

Function: Scan the project for IP address, device ID and other Informations.
Call: Main menu "Tools", Context menu
Requirement: -
\({ }^{\text {² }}\) ) Chapter 6.3.2.9.2 "Configuration of the IP settings with the IP configuration tool" on page 1506

\section*{Command 'Install additional licence' \\ Function: Installs additional engineering license. \\ Call: Main menu "Tools", Context menu \\ Requirement: -}

\section*{Command 'Migrate third party devices'}

Function: After a selection of a previous version profile, all the third party devices which have been installed inside this version profile are listed and can migrated.
Call: Main menu "Tools", Context menu
Requirement: -
② Chapter 6.3.1.5 "Migration of third party devices" on page 1439

\section*{Command 'Package Manager'}

Symbol:
Function: The command opens the "Package Manager" dialog where you install, uninstall, and manage packages.

\section*{Call: Menu bar: "Tools"}

You can also call the Package Manager as a standalone application from the command line.

Table 497: "Currently Installed Packages"
List of installed packages with "Name", "Version","Installation date", "Update info", "License info"
If a package originates from the CODESYS Store, then CODESYS identifies it with the red package symbol instead of the yellow \({ }^{\text {Tl }}\) symbol.
When an update is available, CODESYS indicates this with an entry in the "Update info" column and with the symbol.
\begin{tabular}{|c|c|}
\hline "Refresh" & Refreshes the list \\
\hline "Install" & \begin{tabular}{l}
Opens the standard dialog for finding a file in the file system. By default, the file type is *.package. \\
You can also install two versions of a package. \\
After you select the package, the "Check package signatures" dialog opens. \\
- In the dialog, the package is displayed with the information about signing. Detailed information about signing is displayed in the tooltip and also in a dialog which opens when you double-click a package. \\
- "Allow unsigned and self-signed packages" \(\square\) : The package should be installed although it is unsigned or self-signed. \\
After the package is selected, the installation wizard opens with the dialogs: \\
- "Installation-License Agreement" \\
In this dialog, CODESYS also displays the "Checksum" of the package. \\
Displayed only when the package has a license agreement. \\
- "Choose Setup Type" \\
The options are package-dependent. \\
- "Complete setup": CODESYS installs all components \\
- "Typical setup": CODESYS installs a standard set of components as defined in the package \\
- "Custom setup": CODESYS installs those components which are selected in a dialog \\
- "Installation - Target Versions": You select which of the existing target versions should be updated by the package installation. You have to select at least one version profile. \\
When this dialog is successfully completed, the selected package is ready for installation. You have to close all CODESYS instances in order for the package installation to be automatically started and run.
\end{tabular} \\
\hline "Uninstall" & \begin{tabular}{l}
Uninstalls the selected package \\
- When the "Display versions" option is not selected, CODESYS uninstalls all versions of the selected package. \\
- When the "Display versions" option is selected and you have selected a package node on the top level, CODESYS uninstalls all versions of the selected package. \\
- When the "Display versions" option is selected and you have selected an individual package version, CODESYS uninstalls exactly this version. \\
When this dialog is closed, all CODESYS instances have to be closed in order for the package uninstallation to start.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Details" & \begin{tabular}{l}
Opens the "Details" dialog for the selected package with the following tabs: \\
- "Package Details" \\
- "Name": Package name \\
- "Version" \\
- "Checksum": SHA-1 checksum of the package \\
- "Vendor" \\
- "Copyright" \\
- "Description" \\
- "Installation date" \\
- "License Agreement" \\
- "Installations Log"
\end{tabular} \\
\hline "Search updates in background" & CODESYS automatically searches for updates every time the programming system is started and then one time every hour. \\
\hline "Display versions" & - Displays all versions of the installed package. \\
\hline
\end{tabular}

You can compare the "Checksum" with the package checksum from the package vendor. CODESYS displays this checksum in the "Details" dialog and in the "Installation - License Agreement" dialog of the installation wizard. You do this to make sure that you have installed an original package.

If you have installed a newer version of the programming system in the same installation directory as the previous version, the license information about the previously installed package remains unchanged and CODESYS displays the information in the "Package Manager" dialog.

Table 498: "Updates"
\begin{tabular}{|l|l|}
\hline "Search Updates" & \begin{tabular}{l} 
Searches for the selected package on your system and in the CODESYS Store \\
Updates. \\
CODESYS displays the found updates in the "Update Info" column of the \\
package list.
\end{tabular} \\
\hline "Download" & \begin{tabular}{l} 
Installs the update package from the "Download Package" dialog. In the \\
"Download Package" dialog, click the "Download and Installation" button for this.
\end{tabular} \\
\hline
\end{tabular}

Table 499: "CODESYS Store"
\begin{tabular}{|l|l|}
\hline "Rating" & Give an rating of the package \\
\hline "CODESYS Store" & Link to the homepage of the Store \\
\hline
\end{tabular}

\section*{Command 'Library Repository'}

Symbol: 血
Function: The command opens the "Library Repository" dialog. In this dialog you define which libraries are installed on the local system and are thus available for your application.
Call: Menu bar: "Tools"

\section*{Dialog 'Library \\ Repository'}

Table 500: "Location"
Display of the directory on the local system in which the library files are located. The libraries in this "Location" are listed in the "Installed libraries" area.
"Edit Locations" \(\quad\) Opens the "Edit Repository Locations" dialog.
\[
\begin{aligned}
& \text { 1. You can only use empty directories for new repositories. } \\
& \text { 2. You can also use existing repositories as locations. } \\
& \text { 3. The "System" repository is not editable; CODESYS indicates this by the italic } \\
& \text { lettering of the entry. }
\end{aligned}
\]

Table 501: Dialog "Edit Repository Locations"
\begin{tabular}{|l|l|}
\hline List of the repositories with "Location" and "Name" \\
\hline "Add" & \begin{tabular}{l} 
Creates a new repository. \\
Opens the "Repository Location" dialog. The selected directory ("Location" input \\
field) has to be either empty or an existing valid repository. "Name" is the input \\
field for a symbolic repository name.
\end{tabular} \\
\hline "Edit" & Opens the "Repository Location" dialog (see "Add") \\
\hline "Remove" " & \begin{tabular}{l} 
A dialog box open, asking whether only the entry should be removed from the \\
list of repositories, or whether the directory with the library files should be deleted \\
from the file system. If you want to delete the directory, you have to confirm this.
\end{tabular} \\
\hline
\end{tabular}

Table 502: "Installed libraries"
List of the libraries in a tree structure. Display of each library with category, name, company and version. The icon to the left of the name indicates whether the library is digitally signed or \(\square\) unsigned.
\begin{tabular}{|c|c|}
\hline "Company " & List box for filtering the displayed libraries. \\
\hline "Install" & \begin{tabular}{l}
Opens the "Select Library" dialog. Possible filters: \\
- "Compiled CODESYS library files (*.compiled-library)". \\
- "Compiled CODESYS library files (*.compiled-library-v3)" ab V3 SP15 \\
- "Library files (*.library)" for still uncompiled library projects \\
- "All files (*. *)"
\end{tabular} \\
\hline "Uninstall" & Uninstalls the selected library. \\
\hline "Export" & Opens the default dialog for saving the library project to the local file system. The file type is Library files (*.library), Compiled library files (*. compiled-library), or Compiled library files (*.compiled-library-v3). \\
\hline "Find" & \begin{tabular}{l}
Searches for libraries and function blocks. \\
Opens the "Find Library" dialog. When you enter a string in the input field, CODESYS displays the libraries that it finds with a corresponding string.
\end{tabular} \\
\hline "Details" & \begin{tabular}{l}
Opens the "Details" dialog with details from the project information of the library for the selected version of a library. You find the following information by clicking "More" in the "Details" dialog: \\
- "Size": Specified in bytes \\
- "Created": Creation date \\
- "Changed": Date of the last change \\
- "Last access": Date \\
- "Attributes" \\
- "Properties"
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Dependencies" & \begin{tabular}{l} 
For the selected library, the "Dependencies" dialog opens, showing the depend- \\
encies on other libraries. "Title", "Version" and "Company" are shown for \\
each library reference. References that function via placeholders are displayed \\
according to the syntax: \#<placeholder name>.
\end{tabular} \\
\hline "Group by category" & \(\bullet \square:\) Grouping by library category \\
& \begin{tabular}{l} 
- \(:\) Alphabetical sorting \\
The categories are defined by external description files '*.libcat.xml'.
\end{tabular} \\
\hline
\end{tabular}

Table 503: "Library Profiles"
A library profile defines the library version with which CODESYS resolves a library placeholder if a certain compiler version is set in the project.
\begin{tabular}{|l|l|}
\hline "Import" & \begin{tabular}{l} 
Imports a *. libraryprofile file. \\
If the import already contains existing placeholder entries, a query appears \\
asking whether CODESYS should overwrite it.
\end{tabular} \\
\hline "Export" & \begin{tabular}{l} 
Exports an xml file with the extension *. libraryprofile with the assign- \\
ments of the selected placeholder entries; you can only select a single entry of a \\
"Compiler version".
\end{tabular} \\
\hline
\end{tabular}

Placeholder resolutions can also be defined in the target device currently in use and even by a specific local specification in the Placeholders dialog in the Library Manager.

\section*{See also}
- \(\&\) Chapter 6.4.1.17.3 "Information for Library Developers" on page 2035
- \(\Leftrightarrow\) Chapter 6.4.1.21.2.24 "Object 'Project Information"" on page 2515
- \(\Rightarrow\) Chapter 6.4.1.21.2.16 "Object 'Library Manager'" on page 2469
- \(\Leftrightarrow\) Chapter 6.4.1.21.4.3 "Dialog 'Library Reference Conversion'" on page 2746
- * Chapter 6.4.1.17.6 "Exporting library files" on page 2037

\section*{Command 'License Manager'}

Symbol:
Function: This command opens the wizard for configuring licenses for CODESYS add-on products. The wizard starts with the "License Manager - Select target" dialog.
Call: Menu bar: "Tools".
The License Manager can handle licenses for CODESYS add-on products on the local computer, as well as licenses for RTS add-on products on devices. It supports both the installation in a soft container and on a dongle.

Dialog 'License This is the start dialog of the License Manager wizard. Here you decide where the license will Manager be installed.
Select Target'
\begin{tabular}{|l|l|}
\hline "Workstation" & Local computer \\
\hline "Device" & \begin{tabular}{l} 
Controller. The connection to this device must be configured correctly in order to \\
license ("Communication Settings" tab of the device editor).
\end{tabular} \\
\hline
\end{tabular}

After clicking "Next", you decide the container where you want to manage the licenses.

Dialog 'License
Manager -
Select con-
tainer'
\begin{tabular}{|l|l|}
\hline "Dongle" & \begin{tabular}{l} 
A corresponding dongle must be connected to the computer or device. Not all \\
devices support dongles.
\end{tabular} \\
\hline "Soft container" & \begin{tabular}{l} 
"CODESYS Security Key. A corresponding soft container must be registered in \\
the CodeMeter Control Center. The CODESYS installation provides an existing \\
soft container.
\end{tabular} \\
\hline
\end{tabular}

If you are installing a product on your local computer ("Workstation"), then the "License Manager" opens immediately for the specific selection of the dongle or soft container, and the next actions. This happens after you choose the container type and click "Next >".

If you are licensing the add-on product for a controller, then first the dialog opens for selecting the device in the network after you click "Next >". This dialog corresponds to the classic view of the "Communication Settings" tab of the device editor.

Dialog 'License
Manager'
\begin{tabular}{|l|l|}
\hline "Container" & \begin{tabular}{l} 
Depending on whether "Dongle" or "Soft container" was selected: Drop-down list \\
of all CODESYS dongles or soft containers that were found on the computer or \\
device.
\end{tabular} \\
\hline "Products" & \begin{tabular}{l} 
List of all installed products that are subject to licensing. A prepended symbol \\
indicates the existence and validity of the license. \\
On the right side of the window, the following information is displayed for the \\
selected product and corresponding licenses: \\
"Name" \\
"Nampany" \\
"Compl \\
"Unit counter" \\
"License quantity"
\end{tabular} \\
& \begin{tabular}{l} 
"Usage period" \\
"Feature map" \\
"Activation time" \\
"Expiration time" \\
"Firm code" \\
"Product code" \\
"Description"
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Install Licenses" & \begin{tabular}{l} 
Opens the dialog "Install licenses on <computer> - Select Operation": \\
\(\bullet\)\begin{tabular}{l} 
"Activate license": Opens the dialog "Install licenses on <computer> - \\
Activate License" (see more below) \\
"Request license": Opens the dialog "Install licenses on <computer> - \\
Request License" (see more below) \\
"Install license": Opens the dialog "Install licenses on <computer> - Install \\
License" (see more below)
\end{tabular} \\
\hline "Additional Functions" \\
\\
\hline \begin{tabular}{l} 
Opens the menu with the following actions: \\
"Return license": Opens the "Return Licenses" (see more below) \\
"Restore license": This function is available in the case of device licensing \\
only. Opens the "Restore Licenses" dialog (see more below)
\end{tabular} \\
\hline
\end{tabular} \\
\hline
\end{tabular}

Table 504: "Install Licenses on <computer> - Activate License"
This is the recommended way to activate a license available via the License Server when you have an Internet connection. Requirement: The computer has an Internet connection.
\begin{tabular}{|l|l|}
\hline "Ticket ID" & \begin{tabular}{l} 
Input field for the ticket ID that you received from the software vendor. The \\
ticket ID consists of five sets of five alphanumerical characters (for example: \\
LYSQ3-ZU93K-24LWC-XGW J8-5AY7H).
\end{tabular} \\
\hline "License server" & \begin{tabular}{l} 
Drop-down list of the license server that provides the license for activating the \\
product. You receive the server URL from the software vendor.
\end{tabular} \\
\hline "Select Ticket from Repository" & Opens the "License Repository" dialog. \\
\hline "Next" & \begin{tabular}{l} 
CODESYS connects to the license server. \\
- If the specified ticket contains only one license, then a dialog opens to \\
confirm the successful activation after completion of the server action. \\
If the specified ticket contains multiple licenses, then the dialog "Install \\
licenses - Select Licenses" opens with a list of these licenses (see descrip- \\
tion below).
\end{tabular} \\
\hline
\end{tabular}

Table 505: "Install Licenses - Select Licenses"
Selection of the licenses to be activated for the ticket which you specified in the dialog "Install Licenses - Activate License".
\begin{tabular}{|l|l|}
\hline "Name" & Product name \\
\hline "Available" & Number of available licenses \\
\hline "Used" & Number of used licenses \\
\hline Total & Sum of all used and available licenses \\
\hline Next & \begin{tabular}{l} 
CODESYS connects to the license server. After successful completion of the \\
server action, a dialog opens with the confirmation of the activation.
\end{tabular} \\
\hline
\end{tabular}

Table 506: "Install Licenses on <computer> - Request License"
If the computer does not have an Internet connection, then you can generate a context file from this dialog. The file "WibuCmRaC" is then transmitted to the license server via an Internet-enabled computer. When activation is complete, a license update file "WibuCmRaU" is provided for download.
\begin{tabular}{|l|l|}
\hline "Software vendor" & \begin{tabular}{l} 
Input field for firm codes from the software vendor that provided the license for \\
activating the product. As an alternative, you can select the software vendor from \\
the drop-down list.
\end{tabular} \\
\hline "Context file" & Location and name \\
\hline
\end{tabular}

Table 507: "Install Licenses on <computer> - Install License"
If you downloaded a license update file from the Internet during software activation, then you can use this dialog to install the license on your dongle. To do this, specify the path of the license update file in the input field.

Table 508: "Return License"
\begin{tabular}{|l|l|}
\hline If the license permits, you can return it in order to reactivate it later on another computer. \\
\hline "Ticket ID" & Field for specifying the ticket ID that was used for licensing. \\
\hline "License server" & \begin{tabular}{l} 
Drop-down list for selecting the license server that provides the license for acti- \\
vating the product. You receive the server URL from the software vendor.
\end{tabular} \\
\hline "Load License(s)" & \begin{tabular}{l} 
Button for showing all current licenses installed for the given ticket ID on the \\
server in the "Licenses" window.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Licenses" & \begin{tabular}{l} 
List of licenses available on the server for the given ticket ID. The following \\
information for the selected license is displayed next to the window on the right: \\
\\
\\
\\
\\
\\
\\
- "Name" \\
\\
- "Number of activations" \\
- "Return allowed" \\
- "Activation type" \\
- "Activation date" \\
- "Firm code" \\
- "Comment"
\end{tabular} \\
\hline "Return License(s)" & \begin{tabular}{l} 
Button for returning the selected license(s). These can be reactivated later on \\
another system.
\end{tabular} \\
\hline
\end{tabular}

Table 509: "Install Licenses - Restore Licenses"
When activated, device licenses are saved to a file (*. WibuCmRau) on the local computer and in the "CODESYS Central License Server". If lost, they can be restored from this file to the identical device.
\begin{tabular}{|l|l|}
\hline "Ticket ID" & \begin{tabular}{l} 
Field for specifying the ticket ID that was used for licensing that has already \\
occurred.
\end{tabular} \\
\hline "Restore" & \begin{tabular}{l} 
If a corresponding license backup file is found, then the license is reactivated in \\
the device.
\end{tabular} \\
\hline
\end{tabular}

See also
- Chapter 6.4.1.21.2.8.3 "Tab 'Communication Settings"' on page 2427
- \# Chapter 6.4.1.21.3.9.7 "Command 'License Repository"' on page 2662

\section*{Command 'License Repository’}

Symbol:
Function: This command opens the dialog box "License Repository" for viewing information about the individual licenses.
Call: Main menu "Tools"
Requirements: CODESYS is in offline or online mode.
In the license repository, after entering the ticket number, you can obtain information about the licenses concerned from the central license server.
To do this you can paste the ticket number(s) from the clipboard or import it/them from a text file.

Table 510: "Tickets"
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
<List of the ticket IDs imported \\
into the repository for compo- \\
nents requiring licenses>
\end{tabular} & \\
\hline "Licenses" & \begin{tabular}{l} 
If you select an entry in the list of tickets, the name and the status of the licensed \\
component are displayed here. \\
च: License available and valid \\
I: License found, but invalid \\
X: License not found
\end{tabular} \\
& \begin{tabular}{l} 
In the right-hand part of the dialog box you will then receive then the following \\
information about this license:
\end{tabular} \\
\hline & \begin{tabular}{l} 
"Name": name of the product to be licensed \\
"Item number": item number in the license server. \\
"Return allowed": It is possible to have this license deactivated so that it can be \\
re-activated on another system. \\
"Can be activated": you can have the license activated via the license manager. \\
"Activation quantitiy": number of activations that have taken place so far. \\
"Activation date": date of the current activation \\
"Container serial" "
\end{tabular} \\
\hline "Firm codes" \\
"Comment"
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Import Tickets" & \begin{tabular}{l} 
The standard dialog box for browsing the local file system appears. If you open a \\
text file containing one or more "tickets", i.e. license numbers, these are imported \\
into the repository. Alternatively you can also insert the numbers from the clip- \\
board into the list.
\end{tabular} \\
\hline
\end{tabular}

\section*{Command 'Device Repository'}

Symbol:
Function: This command opens the "Device Repository" dialog. This dialog is used for managing the devices that are installed on the local system and can be integrated into CODESYS projects.
Call: Menu bar: "Tools".

\section*{CAUTION!}

Do not change the internal device repository manually. Do not copy any files to or from the repository. Always use the device repository dialog to install or uninstall devices.

Table 511
\begin{tabular}{|l|l|}
\hline \multicolumn{2}{|c|}{} \\
\hline "Location" & \begin{tabular}{l} 
Shows the device repository directory on the local system. The list box shows \\
the currently set save locations. By default, CODESYS creates the system repo- \\
sitory during installation. The devices of the selected location are listed in the \\
"Installed device descriptions" field.
\end{tabular} \\
\hline "Edit Locations" & Opens the "Edit Repository Locations" dialog. \\
\hline
\end{tabular}

Table 512: Dialog 'Edit Repository Locations'
\begin{tabular}{|l|l|}
\hline List of the repositories with "Location" and "Name". \\
\hline "Add" & \begin{tabular}{l} 
Creates a new repository. \\
Opens the "Repository Location" dialog. The selected directory ("Location" input \\
field) must be empty or it must be a valid repository.
\end{tabular} \\
\hline "Edit" & Opens the "Repository Location" dialog (see "Add"). \\
\hline "Remove" " & \begin{tabular}{l} 
A dialog prompt opens for you to decide whether the respective directory should \\
also be deleted from the hard disk.
\end{tabular} \\
\hline
\end{tabular}

Table 513: "Installed Device Descriptions"
\(\left.\begin{array}{|l|l|}\hline \begin{array}{l}\text { List of device descriptions in multilevel tree structure. Shows all device descriptions with "Name", "Vendor", and } \\ \text { "Version". The top nodes represent device categories, for example PLCs, fieldbuses, and logical devices. }\end{array} \\ \hline \begin{array}{l}\text { "String for full-text search in all } \\ \text { devices" }\end{array} & \begin{array}{l}\text { This field is editable after clicking in it. For any character string entered, only } \\ \text { those devices that include the character string are displayed in the lower view. } \\ \text { The matched string is highlighted in yellow for these devices. }\end{array} \\ \hline \text { "Vendor" } & \text { Drop-down list with manufacturers whose available devices are displayed. } \\ \hline \text { "Install" } & \begin{array}{l}\text { Opens the "Install Device Description" dialog. } \\ \text { For the default devices with file type "*.devdesc.xml". You can also select } \\ \text { manufacturer-specific description files, such as "*.gsd" files for PROFIBUS DP } \\ \text { modules, "*.eds" and "*.dcf" files for CAN devices. } \\ \text { When you click "OK" to confirm the selection, CODESYS inserts the new device }\end{array} \\ \text { into the device repository. If an error occurs during installation (for example, } \\ \text { missing files that are referenced by the device description), then CODESYS } \\ \text { reports the error to the lower part of the device repository dialog. }\end{array}\right\}\)

\section*{NOTICE!}

During installation, CODESYS copies the device description files and all additional reference files to an internal location. Therefore, any changes to the original files no longer influence the installed devices. You must reinstall the devices to make any changes effective. We recommended that you change the internal version number of a device description after a modification.

See also
- ※2 Chapter 6.4.1.18.2 "Installing devices" on page 2038
- \({ }^{\bullet}\) Chapter 6.4.1.21.4.14.6 "Dialog 'Options' - 'Device Description Download"" on page 2785

\section*{Command 'Create Device list CSV'}

Symbol: 圈
Function: MS Excel template of device list for device import is opened.
Call: Main menu "Tools", Context menu
Requirement: -
y Chapter 6.6.1.3.3 "Creating CSV device list" on page 4198

\section*{Command 'Multi Online Change'}

Function: The MultiOnlineChange tool/plug-in for Automation Builder enables firmware update, download and online change of the same project to several AC500 V2 PLCs.
Call: Main menu "Tools", Context menu
Requirement: -

\section*{Command 'Device ECAD data'}

Function: Automation Builder provides an ECAD interface for exchanging the PLC configuration data with EPLAN Electric P8 and Zuken E3. This feature removes double data entry between electrical engineering in the ECAD tool and the control logic programming in Automation Builder by synchronizing the PLC hardware including topology and I/O signals between these tools.

Call: Main menu "Tools", Context menu
Requirement: A project is open.
Chapter 6.6.1.1 "Exporting and importing ECAD data (PBF)" on page 4191

\section*{Command 'OPC UA Information Model Repository'}

Function: The command opens the "OPC UA Information Model" dialog. The OPC UA information models, which are installed on the local system and can be integrated in CODESYS projects, are managed in the dialog.

Call: Menu bar: "Tools"

Table 514: Dialog 'OPC UA Information Model'
\begin{tabular}{|c|c|}
\hline "Location" & Displays the OPC UA information model directories on the local system. The list box shows the currently set locations. By default, CODESYS creates the system repository during installation. The information models of the selected location are listed in the "Installed OPC UA information models" area. \\
\hline "Edit Locations" & Opens the "Edit Repository Locations" dialog. \\
\hline \multicolumn{2}{|l|}{"Installed OPC UA information models"} \\
\hline \multicolumn{2}{|l|}{List of installed information models. Double-click to open installed information model documentation. Note: The information models of this repository can also be added to project archives.} \\
\hline "Install" & \begin{tabular}{l}
Opens the "Select Installed OPC UA Information Model(s)" dialog. \\
- File type: OPC UA Information Models *NodeSet2 . xml (example: "Informationmodel.NodeSet2.xml". When you click "Open", the selected information model is inserted in the repository. \\
- File type: All files * . *: You can select an OPC UA documentation, for example, in PDF or Word format. When you click "Open", the "Assign Documentation OPC UA Information Models" opens. For a description of the dialog, see below.
\end{tabular} \\
\hline "Uninstall" & Uninstalls the selected OPC UA information model. When you delete the information model from the repository, it is no longer available in the development system for use in the CODESYS Development System. \\
\hline "Details" & \begin{tabular}{l}
Opens the "Details" dialog for the selected information model. The dialog includes additional information about the information model. In "Alias", you can specify an alias name for the URI. Moreover, information is displayed as to whether or not a documentation for the information model is available. \\
- "Model URI" \\
- "Publication date" \\
- "Publisher" \\
- "Repository" \\
- "Alias" \\
- "Documentation available": \\
- "Yes": The "Uninstall documentation" button is available. \\
- "No": The "Install documentation" button is available. \\
- "Install documentation": Opens the "Select OPC UA Information Model Documentation" dialog. The data type OPC UA Information Model Documentation (*.pdf) is set as default in the dialog.
\end{tabular} \\
\hline "Documentation" & Opens the installed documentation for the selected information model. If no documentation is installed for the selected information model, then the command is disabled. \\
\hline "Display all versions" & All installed versions of the information model are displayed in a tree structure. \\
\hline
\end{tabular}

See also • \(\begin{aligned} & \text {, } \\ & \text { Chapter 6.4.1.21.2.17 "Object 'OPC UA Information Model'" on page } 2472\end{aligned}\)

\section*{Command 'Scripting' - 'Execute Script File'}

Symbol: 匃
Function: This command opens a dialog for selecting and then executing the script file ( *.py). Call: Menu bar: "Tools \(\rightarrow\) Scripting".

\section*{Command 'Scripting' - 'Enable Script Tracing'}

Symbol:䁶
Function: This command makes CODESYS print all commands from the script file to the message view. Use this command for monitoring and debugging scripts. A blue frame around the symbol indicates that the option is active.
Call: Main menu "Tools \(\rightarrow\) Scripting".

\section*{Command 'Scripting' - 'Scripts'}

Function: This command executes a script that is stored in the ScriptDir folder.
Call: Menu bar: "Tools \(\rightarrow\) Scripting \(\rightarrow\) Scripts".
Requirement: The ScriptDir folder exists in the CODESYS installation directory. Python scripts are stored in this folder with the file extension .py.

All scripts that are contained in the ScriptDir folder are executable as menu commands and are sorted alphabetically by file name.

\section*{Command 'Customize'}

Function: This command opens the "Customize" dialog box, where you can customize the menus, toolbars, and keyboard shortcuts according to your individual requirements.
Call: Main menu "Tools"

\section*{Command 'Options'}

Function: The command opens the dialog box "Options" for the configuration of the CODESYS options. These options define the behavior and appearance of the CODESYS user interface. CODESYS saves the settings in your current user profile on your local system. The current profile specifies the standard settings.

Call: "Tools" menu
See also
- \(\underbrace{*}\) Chapter 6.4.1.21.4.14.23 " Dialog 'Options' - 'SFC Editor'" on page 2795
- \({ }^{4}\), Chapter 6.4.1.21.4.14.4 "Dialog 'Options' - 'CFC Editor"' on page 2784
- \({ }^{\Perp}\) Chapter 6.4.1.21.4.14.5 "Dialog 'Options' - 'Declaration Editor"' on page 2785
- \(y^{\wedge}\) Chapter 6.4.1.21.4.14.7 "Dialog 'Options' - 'Device Editor"' on page 2786
- \(\sum_{\text {- }}\) Chapter 6.4.1.21.4.14.6 "Dialog 'Options' - 'Device Description Download"" on page 2785
- \(\rightleftarrows^{*}\) Chapter 6.4.1.21.4.14.10 "Dialog 'Options' - 'FBD, LD, and IL"' on page 2787
- \({ }^{4}\) Chapter 6.4.1.21.4.14.14 "Dialog 'Options' - 'International Settings'" on page 2790
- \({ }^{\bullet}\), Chapter 6.4.1.21.4.14.15 "Dialog 'Options' - 'Libraries'" on page 2790
- \(\sum^{*}\) Chapter 6.4.1.21.4.14.16 "Dialog 'Options' - 'Library Download"" on page 2790
- \({ }^{\circledR}\) Chapter 6.4.1.21.4.14.17 "Dialog 'Options' - 'Load and Save"' on page 2791
- \(\underbrace{\wedge}\) Chapter 6.4.1.21.4.14.20 "Dialog 'Options' - 'PLCopenXML'" on page 2793
- \(\Leftrightarrow\) Chapter 6.4.1.21.4.14.21 "Dialog 'Options' - 'Proxy Settings"" on page 2793
- \(\sum^{\bullet}\) Chapter 6.4.1.21.4.14.22 "Dialog 'Options' - 'Refactoring'" on page 2794
- \(\boldsymbol{y y}^{*}\) Chapter 6.4.1.21.4.14.24 "Dialog 'Options' - 'SmartCoding'" on page 2796
- « Chapter 6.4.1.21.4.14.26 "Dialog 'Options' - 'Text Editor'" on page 2798

\section*{Command 'Import and Export Options'}

Function: This command opens the "Import and Export Options" dialog. Here you can configure the export and import of selected settings of the CODESYS options. The settings are saved to an XML file with the default extension (options.xml).

Call: Menu bar: "Tools".

Dialog 'Import
and Export
Options'
\begin{tabular}{|l|l|}
\hline "Export selected options" & \begin{tabular}{l} 
"Select options": In the table, you can select the categories of options, either \\
user-specific or machine-specific (computer), whose current settings are to be \\
exported to the XML file.
\end{tabular} \\
& \begin{tabular}{l} 
"File": Path of the export file in the local file system. Example: \\
D: \(\backslash\) system1. options. xml. \\
Button ....: Opens the default dialog to search for an existing file in the local file \\
system, or to create one. The "File type" option export (*. options. xml) \\
is preset.
\end{tabular} \\
\hline "Import selected options" & \begin{tabular}{l} 
"File": Path of the options export file whose contents are to be imported. \\
Button ...: Opens the default dialog to search for an existing file of type option \\
export (*. options. xml) in the local file system. \\
After you click "OK" to close the dialog, the settings described in the file are \\
applied to the project.
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{\circ}\) Chapter 6.4.1.2.1.1 "Setting CODESYS options" on page 1802

\section*{Command 'Device Reader'}

Function: The command opens the standard "Select Device" dialog and reads the license and product information of the selected controller. This license and product information is displayed in the "Device Reader" dialog.
Call: Menu bar: "Tools"
Requirement: No applications exist on the controller.

If the command is selected although an application exists on the controller, then a dialog prompts the user whether or not all applications should be removed from the controller. When the user click "No" to this dialog, the "Device Reader" command is aborted.

Table 515: Dialog "Device Reader"
\begin{tabular}{|l|l|}
\hline "Status of Available Device Features" \\
\hline "Product" & CODESYS product (example: SoftMotion) \\
\hline "Feature" & \begin{tabular}{l} 
Feature of "Product" \\
Example: CNC is a "Feature" of SoftMotion.
\end{tabular} \\
\hline "License Active/Count" & \begin{tabular}{l} 
Yes: A license exists for the feature. \\
No: A license does not exist for the feature. \\
"Count": Number of licenses
\end{tabular} \\
\hline
\end{tabular}

\section*{Menu 'Window'}
6.4.1.21.3.10.1 Command 'Next Editor'. ..... 2669
6.4.1.21.3.10.2 Command 'Previous Editor' ..... 2669
6.4.1.21.3.10.3 Command 'Close All Editors' ..... 2669
6.4.1.21.3.10.4 Command 'Close All Editors of Inactive Applications' ..... 2670
6.4.1.21.3.10.5 Command 'Reset Window Layout'. ..... 2670
6.4.1.21.3.10.6 Command 'New Horizontal Tab Group' ..... 2670
6.4.1.21.3.10.7 Command 'New Vertical Tab Group' ..... 2670
6.4.1.21.3.10.8 Command 'Float'. ..... 2671
6.4.1.21.3.10.9 Command 'Dock'. ..... 2671
6.4.1.21.3.10.10 Command 'Auto Hide' ..... 2671
6.4.1.21.3.10.11 Command 'Next Pane'. ..... 2671
6.4.1.21.3.10.12 Command 'Previous Pane'. ..... 2671
6.4.1.21.3.10.13 Command 'Toggle First Pane'. ..... 2672
6.4.1.21.3.10.14 Command 'Toggle Second Pane'. ..... 2672
6.4.1.21.3.10.15 Command 'Windows' ..... 2672
6.4.1.21.3.10.16 Command 'Close All Editors But This'. ..... 2673
6.4.1.21.3.10.17 Command 'Select Object in Navigator'. ..... 2673
6.4.1.21.3.10.18 Command 'Select Parent Object in Navigator'. ..... 2673
6.4.1.21.3.10.19 Commands of the Submenu 'Window' ..... 2673

\section*{Command 'Next Editor'}

Keyboard shortcut: [Ctrl]+[F6]
Function: This command switches focus from the currently active view to the next view. The next view is identified by the tab to the right of the currently active tab.

Call: Main menu "Window"
Requirement: At least one object is open.
See also
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.10.2 "Command 'Previous Editor'" on page 2669

\section*{Command 'Previous Editor'}

Keyboard shortcut: [Shift]+[Ctrl]+[F6]
Function: This command switches focus from the currently active view to the previous view. The previous view is identified by the tab to the left of the currently active tab.
Call: Main menu "Window"
Requirement: At least one object is open.
See also
- ② Chapter 6.4.1.21.3.10.1 "Command 'Next Editor'" on page 2669

\section*{Command 'Close All Editors'}

\section*{Symbol:}

Function: This command closes all currently open editor views.
Call: Main menu "Window"
Requirement: At least one editor is open.

See also
- \({ }^{\circ} \Rightarrow\) Chapter 6.4.1.21.3.10.4 "Command 'Close All Editors of Inactive Applications'" on page 2670
- \(\quad \stackrel{y}{r}\) Chapter 6.4.1.21.3.10.16 "Command 'Close All Editors But This'" on page 2673

\section*{Command 'Close All Editors of Inactive Applications'}

Function: This command closes all editor views for objects that are located directly below a currently inactive application. Object editors in the POU view remain open.
Call: Main menu "Window"
Requirement: At least one object of an inactive application is open.
See also
- ③ Chapter 6.4.1.21.3.10.3 "Command 'Close All Editors'" on page 2669
- \({ }^{*}\), Chapter 6.4.1.21.3.10.16 "Command 'Close All Editors But This'" on page 2673

\section*{Command 'Reset Window Layout'}

Function: This command resets all currently open windows and views to their default positions. You are prompted for a confirmation before the command is executed.

Call: Main menu "Tools"

\section*{Command 'New Horizontal Tab Group'}

Symbol:
Function: This command moves the currently active view to a new, separate tab group below the existing one.
Call: Main menu "Window" or context menu of the tab
Requirement: Several editor views are open as tabs next to each other.
If you open another object in the editor, then this is automatically included in the tab group that is currently in focus.
See also
- « Chapter 6.4.1.21.3.10.7 "Command 'New Vertical Tab Group"' on page 2670

\section*{Command 'New Vertical Tab Group'}

Symbol:
Function: This command moves the currently active view to a new, separate tab group to the right of the existing one.
Call: Main menu "Window" or context menu of the tab
Requirement: Several editor views are open as tabs next to each other.
If you open another object in the editor, then this is automatically included in the tab group that is currently in focus.

See also
- \({ }^{*} \boldsymbol{y}\) Chapter 6.4.1.21.3.10.6 "Command 'New Horizontal Tab Group'" on page 2670

\section*{Command 'Float'}

Function: This command releases a docked view from its frame in the user interface and repositions it on the screen as a floating window.

Call: Main menu "Window"
Requirement: The application is in online mode.
This window can then be positioned outside of the user interface. Use the "Dock" command to return a floating window to the frame of the user interface.
See also
- ⓢ Chapter 6.4.1.21.3.10.9 "Command 'Dock"' on page 2671

\section*{Command 'Dock'}

Function: This command returns a floating window, which was released by the "Float" command, to the frame of the user interface.

Call: Main menu "Window"
See also
- \(y^{\prime 2}\) Chapter 6.4.1.21.3.10.8 "Command 'Float"' on page 2671

\section*{Command 'Auto Hide'}

Keyboard shortcut: [F7]
Function: This command shows or hides a view.
Call: Main menu "Window"
Hide simply means that CODESYS minimizes the view to a tab at the bottom of the user interface which is visible only when you move the mouse over the tab. The command functions like a check box. When a window is hidden, the check box is selected in the menu. When you click the command again, the checkbox is cleared and the window is shown.

\section*{Command 'Next Pane'}

Keyboard shortcut: [F6]
Function: This command sets the focus on the next pane.
Call: Main menu "Window"
Requirement: An object is open that contains two or more panes.
Example: If an object is open in the ST editor and the cursor is currently in the declaration section, then command sets the focus to implementation section.

See also
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.10.12 "Command 'Previous Pane"" on page 2671

\section*{Command 'Previous Pane'}

Keyboard shortcut: [Shift]+[F6]
Function: This command sets the focus on the previous pane.
Call: Main menu "Window"
Requirement: An object is open that contains two or more panes.

Example: If an object is open in the ST editor and the cursor is currently in the declaration section, then command sets the focus to implementation section.
See also
- \({ }^{4}\), Chapter 6.4.1.21.3.10.11 "Command 'Next Pane'" on page 2671

\section*{Command 'Toggle First Pane'}

Keyboard shortcut [Alt]+[F6]
Function: This command shows and hides the declaration view.
Call: "Window".
Requirement: The cursor is positioned in the editor of one of the following objects:
- POU
- Transition
- Method
- Get accessor method of a property
- Set accessor method of a property
- Visualization


You can also toggle the subviews by means of the buttons.

See also
- \(\sum^{4}\) Chapter 6.4.1.21.3.10.14 "Command 'Toggle Second Pane'" on page 2672

\section*{Command 'Toggle Second Pane'}

Function: This command shows and hides the implementation view.
Call: "Window".
Requirement: The cursor is positioned in the editor of one of the following objects:
- POU
- Transition
- Method
- Get accessor method of a property
- Set accessor method of a property
- visualization


See also
- ② Chapter 6.4.1.21.3.10.13 "Command 'Toggle First Pane"' on page 2672

\section*{Command 'Windows'}

Function: This command opens the "Windows" dialog box, which lists all open objects. You can then activate or close any of the listed views.

Call: Main menu "Window"

\section*{Command 'Close All Editors But This'}

Function: This command closes all editor views except the currently open one.
Call: Right-click the tab
Requirement: At least two objects are open.
See also
- \& Chapter 6.4.1.21.3.10.3 "Command 'Close All Editors"' on page 2669
- \& Chapter 6.4.1.21.3.10.4 "Command 'Close All Editors of Inactive Applications"' on page 2670

\section*{Command 'Select Object in Navigator'}

Function: This command selects the object of the active editor in the device tree.
Call: Right-click the tab
Requirement: At least one object is open.


This command is executed automatically when you select the "Track active editor" option for the device tree.

See also
- \({ }^{4}\) Chapter 6.4.1.21.3.10.18 "Command 'Select Parent Object in Navigator'" on page 2673

\section*{Command 'Select Parent Object in Navigator'}

Function: This command selects the parent object in the device tree.
Call: Right-click the tab
Requirement: At least one object is open.
See also
- \({ }^{\text {© }}\) Chapter 6.4.1.21.3.10.17 "Command 'Select Object in Navigator"' on page 2673

\section*{Commands of the Submenu 'Window'}

Function: The command activates the selected window.
Call: Main menu "Window"
For each opened editor window the menu "Window" contains a command "<n><object name>". Choosing this command activates the corresponding window. In offline mode CODESYS adds the extension "(Offline)". To differentiate between the implementation or the instances of a function block the extension "(Impl)" or "<instance path>" is added.

\section*{Menu 'Help'}
6.4.1.21.3.11.1 Command 'Contents' ..... 2674
6.4.1.21.3.11.2 Command 'Index'. ..... 2674
6.4.1.21.3.11.3 Command 'Find'. ..... 2674
6.4.1.21.3.11.4 Command 'About'. ..... 2675

\section*{Command 'Contents'}

Symbol: k; keyboard shortcut: [Ctrl]+[Shift]+[F1]
Function: This command opens the CODESYS help.
Call: Menu bar: "Help".

\section*{Command 'Index'}

Symbol: ?]; keyboard shortcut: [Ctrl]+[Shift]+[F2]
Function: This command opens the CODESYS help.
Call: Menu bar: "Help".
An index search is not possible in the online help. The "Index" tab opens in the offline help.
All index entries of the help are listed alphabetically in the index view.
\begin{tabular}{|l|l|}
\hline "Look for" " & \begin{tabular}{l} 
As you type letters into the input field, CODESYS searches automatically for \\
matches in the index list.
\end{tabular} \\
\hline "Display" & \begin{tabular}{l} 
Opens the help page for the highlighted index entry in the list and displays the \\
title of the help page and location of the help file (*.chm) in the "Index results for \\
<index entry>" view. When several pages are found and then displayed in this \\
view, then you view a specific help page by clicking its entry in the list. \\
Clicking an entry in the index list achieves the same result.
\end{tabular} \\
\hline
\end{tabular}

\section*{Command 'Find'}

\section*{Symbol:}

Function: This command opens the CODESYS help.
Call: Menu bar: "Help".
In the online help, you can run a full-text search from the input field on the top right of the help page. The "Find" tab opens in the offline help.

Table 516: Tab 'Search'
\begin{tabular}{|l|l|}
\hline "Search for" & \begin{tabular}{l} 
Combo box for defining the search term or for selecting the 25 most recent \\
search terms.
\end{tabular} \\
\hline "Search in titles only" & The search is performed only in the titles of the help pages. \\
\hline "Display partial matches" & Displays terms also as search results that include the search term. \\
\hline "Limit to .... matches" & \begin{tabular}{l} 
Limits the number of search results. \\
Maximum value: 1000
\end{tabular} \\
\hline "Find" & Starts the full-text search. \\
\hline
\end{tabular}

\section*{Command 'About'}

Function: This command opens a splash screen with information about the CODESYS version and copyright. In addition, buttons are available for detailed information about the version, license, and acknowledgments.

Call: Main menu "Help".
\begin{tabular}{|l|l|}
\hline "Version Info" & \begin{tabular}{l} 
Opens the "Detailed Version Information" dialog box with a list of CODESYS \\
components and information about the operating system. \\
"Export": Exports the detailed version information as a *.txt file or in any other \\
format.
\end{tabular} \\
\hline "License Info" & \begin{tabular}{l} 
Opens the "License Information" dialog box. \\
- "Plug-in": Drop-down list for the plug-in to display the license information \\
- "Software License": License information about selected "Plug-in"
\end{tabular} \\
\hline "Acknowledgments" & \\
\hline
\end{tabular}

\section*{Menu 'SFC'}
6.4.1.21.3.12.1 Command 'Init Step' ..... 2675
6.4.1.21.3.12.2 Command 'Insert Step' ..... 2676
6.4.1.21.3.12.3 Command 'Insert Step After'. ..... 2676
6.4.1.21.3.12.4 Command 'Insert Transition After' ..... 2676
6.4.1.21.3.12.5 Command 'Insert Transition'. ..... 2677
6.4.1.21.3.12.6 Command 'Insert Step-Transition' ..... 2677
6.4.1.21.3.12.7 Command 'Insert Step-Transition After' ..... 2677
6.4.1.21.3.12.8 Command 'Add Entry Action' ..... 2678
6.4.1.21.3.12.9 Command 'Add Exit Action'. ..... 2678
6.4.1.21.3.12.10 Command 'Parallel'. ..... 2678
6.4.1.21.3.12.11 Command 'Alternative'. ..... 2679
6.4.1.21.3.12.12 Command 'Insert Branch'. ..... 2679
6.4.1.21.3.12.13 Command 'Insert Branch Right'. ..... 2679
6.4.1.21.3.12.14 Command 'Insert Action Association'... ..... 2680
6.4.1.21.3.12.15 Command 'Insert Action Association After' ..... 2681
6.4.1.21.3.12.16 Command 'Insert Jump'. ..... 2681
6.4.1.21.3.12.17 Command 'Insert Jump After' ..... 2681
6.4.1.21.3.12.18 Command 'Insert Macro'.... ..... 2682
6.4.1.21.3.12.19 Command 'Insert Macro After'. ..... 2682
6.4.1.21.3.12.20 Command 'Zoom Into Macro' ..... 2682
6.4.1.21.3.12.21 Command 'Zoom Out of Macro'. ..... 2682
6.4.1.21.3.12.22 Command 'Paste After'. ..... 2683
6.4.1.21.3.12.23 Command 'Change Duplication' - 'Set' ..... 2683
6.4.1.21.3.12.24 Command 'Change Duplication' - 'Remove'. ..... 2683
6.4.1.21.3.12.25 Command 'Do Not Display Embedded Objects' ..... 2684

\section*{Command 'Init Step'}

Symbol: 追
Function: This command converts the selected step into an initial step.
Call: Main menu "SFC"

After you choose this command, the borders of the step element change to a double line. The previous initial step is automatically displayed as a normal step with a single-line border.
You can also activate and deactivate the property "Init step" in the properties dialog of a step. However, CODESYS does not automatically adjust the settings of other steps.

This command is useful if you want to convert a diagram. When you create a new SFC object, it automatically includes an initial step followed by a transition (TRUE) and a jump back to the initial step.

Please note: In online mode, it is possible to reset the diagram to the initial step using the SFCInit and SFCReset flags.

See also
- \& Chapter 6.4.1.20.1.5.6 "SFC Flags" on page 2067
- \(\#\) Chapter 6.4.1.20.1.5.8.6 "SFC element properties" on page 2080

\section*{Command 'Insert Step'}

Symbol: \(\square \uparrow\)
Function: This command inserts a step before the selected point.
Call: Menu bar "SFC"; context menu in SFC editor
The new step is named \(S t e p<n>\) by default, where \(n\) is an incremental number starting at 0 for the first step that is inserted in addition to the initial step. The name can be edited by clicking on it.

See also
- \({ }^{\wedge}\) Chapter 6.4.1.21.3.12.7 "Command 'Insert Step-Transition After'" on page 2677
- \(\quad>\) Chapter 6.4.1.21.3.12.1 "Command 'Init Step'" on page 2675
- \(\quad \stackrel{y}{c}\) Chapter 6.4.1.20.1.5.8.1 "SFC elements 'Step' and 'Transition"' on page 2073

\section*{Command 'Insert Step After'}

Symbol: \(\downarrow \downarrow\)
Function: This command inserts a step after the selected point.
Call: Menu bar "SFC"; context menu in SFC editor
The new step is named \(S t e p<n>\) by default, where \(n\) is an incremental number starting at 0 for the first step that is inserted in addition to the initial step. The name can be edited by clicking on it.

See also

- \(\boldsymbol{y}^{*}\) Chapter 6.4.1.21.3.12.1 "Command 'Init Step"" on page 2675
- \(\Downarrow\) Chapter 6.4.1.20.1.5.8.1 "SFC elements 'Step' and 'Transition"' on page 2073

\section*{Command 'Insert Transition After'}

Symbol: \(+\downarrow\)
Function: This command inserts a transition after the selected point.
Call: Menu bar "SFC"; context menu in SFC editor

The new transition is named Trans<n> by default, where n is an incremental number beginning at 0 for the first transition. The name can be edited by clicking on it.

See also
- \(\quad{ }^{\circ}\) Chapter 6.4.1.21.3.12.7 "Command 'Insert Step-Transition After"' on page 2677
- \({ }^{4}\), Chapter 6.4.1.20.1.5.8.1 "SFC elements 'Step' and 'Transition"' on page 2073

\section*{Command 'Insert Transition'}

Symbol: + \(\uparrow\)
Function: This command inserts a transition before the selected point.
Call: Menu bar "SFC"; context menu in SFC editor
The new transition is named Trans<n> by default, where \(n\) is an incremental number beginning at 0 for the first transition. The name can be edited by clicking on it.

See also
- \({ }^{\text {® }}\) Chapter 6.4.1.21.3.12.7 "Command 'Insert Step-Transition After"" on page 2677
- \(\sum^{\Perp}\) Chapter 6.4.1.20.1.5.8.1 "SFC elements 'Step' and 'Transition"' on page 2073

\section*{Command 'Insert Step-Transition'}

Symbol: \(甲 \uparrow\)
Function: This command inserts a step and a transition before the selected point.
Call: Main menu "SFC"
If you have selected a step, then CODESYS inserts a new step-transition combination. If you have selected a transition, then a new transition-step combination is inserted.

The new step is named \(S t e p<n>\) by default, where \(n\) is an incremental number beginning at 0 for the first step that was inserted in addition to the initial step. The new transition is named Trans<n> by default. You can edit the default names directly by clicking the names.

See also
- ② Chapter 6.4.1.21.3.12.7 "Command 'Insert Step-Transition After"" on page 2677
- \({ }^{4} \boldsymbol{y}\) Chapter 6.4.1.21.3.12.1 "Command 'Init Step"" on page 2675
- \({ }^{*}\) Chapter 6.4.1.20.1.5.8.1 "SFC elements 'Step' and 'Transition"' on page 2073

\section*{Command 'Insert Step-Transition After'}

Symbol: \(+\downarrow\)
Function: This command inserts a step and a transition after the selected point.
Call: Main menu "SFC"
If you have selected a step, then CODESYS inserts a new transition-step combination. If you have selected a transition, then a new step-transition combination is inserted.
The new step is named \(S t e p<n>\) by default, where \(n\) is an incremental number beginning at 0 for the first step that was inserted in addition to the initial step. The new transition is named Trans<n> by default. You can edit the default names directly by clicking the names.
See also
- \({ }^{\wedge} \boldsymbol{y}\) Chapter 6.4.1.21.3.12.6 "Command 'Insert Step-Transition"" on page 2677

\section*{Command 'Add Entry Action'}

\section*{Symbol:}

Function: This command opens the "Add Entry Action" dialog box where you define a new entry action. Depending on the SFC options, a dialog prompt may open for selecting the duplication mode for the new step action.
Call: Menu bar: "SFC"; context menu of the selected step element.
Requirement: A step element in SFC is selected.
The entry action is opened automatically in the ST editor. The step element contains an E in the lower left corner.

\section*{Confirmation prompt for selecting the duplication mode}

\section*{Options:}
- "Copy reference: A new step will call the same actions": If the step is copied in SFC, the link to the step action(s) is also copied. The steps copied by each other will therefore call the same actions.
- "Copy implementation: New action objects are created for a new step.": This means that the step actions for a copied step are embedded. By default, the generated action objects appear below an SFC box in the device tree or "POUs" view. These objects contains a copy of the original implementation code of the respective action.
The display of the embedded objects can be activated and deactivated in the tree by means of the "Show Embedded Objects" and "Hide Embedded Objects" commands in the context menu of an SFC object.
See also
- \(\xi^{\circ}\) Chapter 6.4.1.21.4.14.23 " Dialog 'Options' - 'SFC Editor"" on page 2795
- \# Chapter 6.4.1.9.5.5.2 "Programming in SFC" on page 1880
- \({ }^{\bullet}\) - Chapter 6.4.1.20.1.5.8.2 "SFC Element 'Action'" on page 2075
- \({ }^{*} \gg\) Chapter 6.4.1.21.3.12.25 "Command 'Do Not Display Embedded Objects'" on page 2684

\section*{Command 'Add Exit Action'}

Symbol:
Function: This command opens the "Add Exit Action" dialog box where you define a new exit action. Depending on the SFC options, a dialog prompt may open for selecting the duplication mode for the new step action. For more information, refer to the help page for the "Add Exit Action" command.

Call: Menu bar: "SFC"; context menu of the selected step element.
Requirement: A step element in SFC is selected.

\section*{See also}
- \(\quad>\) Chapter 6.4.1.21.3.12.8 "Command 'Add Entry Action'" on page 2678
- \({ }^{〔}\) Chapter 6.4.1.21.4.14.23 " Dialog 'Options' - 'SFC Editor'" on page 2795
- \({ }^{*}>\) Chapter 6.4.1.9.5.5.2 "Programming in SFC" on page 1880
- \({ }^{*}>\) Chapter 6.4.1.20.1.5.8.2 "SFC Element 'Action"" on page 2075
- \(\boldsymbol{y y}^{\prime}\) Chapter 6.4.1.21.3.12.25 "Command 'Do Not Display Embedded Objects'" on page 2684

\section*{Command 'Parallel'}

Symbol: 照
Function: This command converts the selected alternative branch into a parallel branch.
Call: Main menu "SFC"
Requirement: The horizontal connecting line of a branch is selected.

Please note that after you convert a branch，you must check and modify the layout of the steps and transitions before and after the branch．
See also
－ⓨ Chapter 6．4．1．21．3．12．11＂Command＇Alternative＇＂on page 2679

\section*{Command＇Alternative＇}

Symbol：卢
Function：This command converts the selected parallel branch into an alternative branch．
Call：Main menu＂SFC＂
Requirement：The horizontal connecting line of a branch is selected．
Please note that after you convert a branch，you must check and modify the layout of the steps and transitions before and after the branch．
See also
－\({ }^{4}\) Chapter 6．4．1．21．3．12．10＂Command＇Parallel＂＇on page 2678

\section*{Command＇Insert Branch＇}

Symbol：昂
Function：This command inserts a branch to the left of the selected point．
Call：Main menu＂SFC＂
This command functions similar to the＂Insert Branch Right＂command．
See also
－\({ }^{4}\) Chapter 6．4．1．20．1．5．8．3＂SFC element＇Branch＂＇on page 2078
－ \(\boldsymbol{y}^{2}\) Chapter 6．4．1．21．3．12．13＂Command＇Insert Branch Right＂＇on page 2679

\section*{Command＇Insert Branch Right＇}

Symbol：悹
Function：This command inserts a branch to the right of the selected point．
Call：Main menu＂SFC＂
The type of inserted branch depends on the selected element．
－If the uppermost element of the selected elements is a transition or an alternative branch， then CODESYS inserts an alternative branch．
－If the uppermost element of the selected elements is a step，a macro，a jump，or a parallel branch，then CODESYS inserts a parallel branch with the Branch＜x＞jump marker，where \(x\) is an incremental number．You can edit the default name of the jump marker or define the jump marker as a jump destination．
－If a common element of an existing branch（horizontal line）is selected，then CODESYS inserts the new branch line as a branch line on the far right．If an entire branch line of an existing branch is selected，then CODESYS inserts the new branch line directly to the right as a new branch line．

Please note：You can convert a branch into another type with the＂Alternative＂ and＂Parallel＂commands．


Example of
alternative branch

The following image shows a new inserted alternative branch generated by the "Insert Branch Right" command while the \(t 4\) transition was selected. CODESYS automatically inserts a step (Step32 in the example), a preceding transition, and a subsequent transition ( \(\mathrm{t} 41, \mathrm{t} 42\) ).

Processing in online mode: If Step 3 is active, then CODESYS passes the subsequent transitions ( \(t 4, t 41\) ) from left to right. The first branch line of the main branch with the first transition yielding TRUE is passed. Therefore, only one branch line is processed as opposed to with a parallel branch.


See also
- © Chapter 6.4.1.20.1.5.8.3 "SFC element 'Branch"' on page 2078
- \& Chapter 6.4.1.21.3.12.12 "Command 'Insert Branch"' on page 2679
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.12.10 "Command 'Parallel"' on page 2678
- \& Chapter 6.4.1.21.3.12.11 "Command 'Alternative"' on page 2679

\section*{Command 'Insert Action Association'}

Symbol: \(\mp\)
Function: This command assigns an IEC action to a step.
Call: Main menu "SFC"

Requirement: A step is selected.
CODESYS inserts the action element to the right of the selected step element.
If you have already assigned one or more actions to the step, they are displayed in an action list. The new action is then inserted as follows:
- If you selected the step element, the action is inserted as the first action of the step at first position of the action list.
- If you selected one of the available actions in the action list, then the action is inserted directly above the selected action.
The left section of the action element includes the qualifier ( N by default). You enter the action name in the right section. To set this value, click in the box to obtain an editing frame. You must have already created this action as a POU in the project.
You can also edit the qualifier. Valid qualifiers are described in the chapter "Qualifiers for Actions in SFC".

See also
- « Chapter 6.4.1.21.3.12.15 "Command 'Insert Action Association After"' on page 2681
- ② Chapter 6.4.1.20.1.5.4 "Qualifiers for Actions in SFC" on page 2065

\section*{Command 'Insert Action Association After'}

Symbol: \(\quad \downarrow\)
Function: This command assigns an IEC action to a step.
Call: Main menu "SFC"
Requirement: A step is selected.
This command functions similar to the "Insert Action Association" command. The difference between the two commands is that CODESYS inserts the new action in the last position of the action list, not the first position. If you select an action in the action list, then CODESYS inserts the new action at the bottom of the list, not at the top.

See also
- \({ }^{\text {® }}\) Chapter 6.4.1.21.3.12.14 "Command 'Insert Action Association'" on page 2680
- ⓨ Chapter 6.4.1.20.1.5.4 "Qualifiers for Actions in SFC" on page 2065

\section*{Command 'Insert Jump'}

Symbol: L \(\mathrm{L}_{\mathrm{L}} \uparrow\)
Function: This command inserts a jump element before the selected element.
Call: Main menu "SFC"
Requirement: A step is selected.
CODESYS automatically inserts the jump with the Step destination. Then, you still have to replace this jump destination with an actual destination by using the input assistant.

See also
- „y Chapter 6.4.1.20.1.5.8.4 "SFC element 'Jump"' on page 2079
- \(\quad\) Chapter 6.4.1.21.3.12.17 "Command 'Insert Jump After"' on page 2681

\section*{Command 'Insert Jump After'}

Symbol: Lo \(\downarrow\)
Function: This command inserts a jump element after the selected element.

Call: Main menu "SFC"
CODESYS automatically inserts the jump with the step destination. Then, you still have to replace this jump destination with an actual destination by using the input assistant.
See also
- „ Chapter 6.4.1.20.1.5.8.4 "SFC element 'Jump"" on page 2079
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.12.16 "Command 'Insert Jump"" on page 2681

\section*{Command 'Insert Macro'}

Symbol: 回 \(\uparrow\)
Function: This command inserts a macro element before the selected element.
Call: Main menu "SFC"
The new macro is named Macro \(\langle x\rangle\) by default, where \(x\) is an incremental number beginning at 0 for the first macro. You can edit the default name directly by clicking the name.

To edit the macro, click "Zoom Into Macro" in the macro editor.
See also
- \(\Leftarrow\) Chapter 6.4.1.21.3.12.20 "Command 'Zoom Into Macro'" on page 2682
- \(\quad \Rightarrow\) Chapter 6.4.1.21.3.12.19 "Command 'Insert Macro After'" on page 2682

\section*{Command 'Insert Macro After'}

\section*{Symbol: 审 \(\downarrow\)}

Function: This command inserts a macro element after the selected element.
Call: Main menu "SFC"
This command functions similar to the "Insert Macro" command.
See also
- \({ }^{\circledR}\) Chapter 6.4.1.21.3.12.20 "Command 'Zoom Into Macro'" on page 2682
- \(\quad>\) Chapter 6.4.1.21.3.12.18 "Command 'Insert Macro'" on page 2682

\section*{Command 'Zoom Into Macro'}

Symbol: \(\downarrow\)
Function: This command opens a macro for editing in the macro editor.
Call: Main menu "SFC"
Requirement: A macro is selected.
By choosing this command, CODESYS closes the main view of the SFC editor and opens the macro editor. This is also an SFC editor for editing the section of the SFC diagram that is displayed as a macro box in the main view.
Click "Zoom Out of Macro" to return to the main view.
See also
- \({ }^{\text {® }}\) Chapter 6.4.1.21.3.12.21 "Command 'Zoom Out of Macro'" on page 2682

\section*{Command 'Zoom Out of Macro'}

Symbol: \(\checkmark\)

Function: This command closes the macro editor and returns to the main view of the SFC editor.

Call: Main menu "SFC"
Requirement: A macro is open in the macro editor.

\section*{See also}
- \({ }^{4}\) ) Chapter 6.4.1.21.3.12.20 "Command 'Zoom Into Macro"' on page 2682

\section*{Command 'Paste After'}

Symbol: 圆 \(\downarrow\)
Function: This command pastes the elements from the clipboard after the selected position.
Call: Main menu "SFC".

\section*{Command 'Change Duplication' - 'Set'}

Function: This command embeds every step action or transition, which is called by a step or transition in the SFC box, with the caller. In this way, the action or transition object can be called only from exactly this caller (pseudo-embedding). The result is that copying step and transition elements that call cations or transitions automatically creates new action or transition objects. the implementation code is also copied.

Call: Menu bar: "SFC".
For more details about duplication mode, refer to the help page for the SFC element properties and the instructions for adding step actions.

Pseudo-embedded objects can be hidden in the "Devices" or "POUs" view by means of a command.

See also
- \& Chapter 6.4.1.20.1.5.8.6 "SFC element properties" on page 2080
- \({ }^{\bullet}>\) Chapter 6.4.1.9.5.5.2 "Programming in SFC" on page 1880

\section*{Command 'Change Duplication' - 'Remove'}

Function: This command removes the embedding of action, transition, and property objects by a step or transition that calls it for the entire SFC box. In this way, the pseudo-embedding of the action, transition, or property objects is removed. If step or transition elements are copied, which call actions, transitions, or properties, then the copying calls the same actions and transitions as the source.

Call: Menu bar: "SFC".
For more details about duplication mode, refer to the help page for the SFC element properties and the instructions for adding step actions.

Pseudo-embedded objects can be hidden in the "Devices" or "POUs" view by means of a command.

\section*{See also}
- \({ }^{*}\) Chapter 6.4.1.20.1.5.8.6 "SFC element properties" on page 2080
- \({ }^{\bullet}\), Chapter 6.4.1.9.5.5.2 "Programming in SFC" on page 1880
- \({ }^{\bullet}>\) Chapter 6.4.1.21.3.12.25 "Command 'Do Not Display Embedded Objects'" on page 2684

\section*{Command 'Do Not Display Embedded Objects'}

Function: This command causes action and transition objects, which are embedded in an SFC box by a step or transition, do not appear in the tree.

Call: Context menu of an SFC box in the "Devices" or "POUs" view.
See also
- \({ }^{\text {® }}\), Chapter 6.4.1.9.5.5.2 "Programming in SFC" on page 1880

\section*{Menu 'CFC'}
6.4.1.21.3.13.1 Command 'Edit Worksheet' ..... 2685
6.4.1.21.3.13.2 Command 'Edit Page Size' ..... 2686
6.4.1.21.3.13.3 Command 'Negate'. ..... 2686
6.4.1.21.3.13.4 Command 'EN/ENO'. ..... 2686
6.4.1.21.3.13.5 Command 'None' ..... 2687
6.4.1.21.3.13.6 Command 'R (Reset)' ..... 2687
6.4.1.21.3.13.7 Command 'S (Set)'. ..... 2687
6.4.1.21.3.13.8 Command 'REF= (Reference Assignment)'. ..... 2687
6.4.1.21.3.13.9 Command 'Display Execution Order' ..... 2688
6.4.1.21.3.13.10 Command 'Set Start of Feedback'. ..... 2688
6.4.1.21.3.13.11 Command 'Send to Front'. ..... 2688
6.4.1.21.3.13.12 Command 'Send to Back'. ..... 2689
6.4.1.21.3.13.13 Command 'Move Up'. ..... 2689
6.4.1.21.3.13.14 Command 'Move Down'. ..... 2690
6.4.1.21.3.13.15 Command 'Set Execution Order'.... ..... 2690
6.4.1.21.3.13.16 Command 'Order by Data Flow' ..... 2691
6.4.1.21.3.13.17 Command 'Order by Topology'. ..... 2691
6.4.1.21.3.13.18 Command 'Edit Parameters'. ..... 2692
6.4.1.21.3.13.19 Command 'Save Prepared Parameters to Project' ..... 2693
6.4.1.21.3.13.20 Command 'Connect Selected Pins'. ..... 2693
6.4.1.21.3.13.21 Command 'Unlock Connection'. ..... 2693
6.4.1.21.3.13.22 Command 'Show Next Collision'. ..... 2694
6.4.1.21.3.13.23 Command 'Select Connected Pins'. ..... 2694
6.4.1.21.3.13.24 Command 'Reset Pins' ..... 2694
6.4.1.21.3.13.25 Command 'Remove Unused Pins'. ..... 2694
6.4.1.21.3.13.26 Command 'Add Input Pin'. ..... 2695
6.4.1.21.3.13.27 Command 'Add Output Pin' ..... 2695
6.4.1.21.3.13.28 Command 'Route All Connections'. ..... 2695
6.4.1.21.3.13.29 Command 'Remove Control Point'. ..... 2695
6.4.1.21.3.13.30 Command 'Create Control Point'. ..... 2696
6.4.1.21.3.13.31 Command 'Connection Mark'. ..... 2696
6.4.1.21.3.13.32 Command 'Create group'. ..... 2696
6.4.1.21.3.13.33 Command 'Ungroup'. ..... 2697
6.4.1.21.3.13.34 Command 'Prepare Box for Forcing'. ..... 2697
6.4.1.21.3.13.35 Command 'Force Function Block Input'. ..... 2697
6.4.1.21.3.13.36 Command 'Use Attributed Member as Input'. ..... 2698

\section*{Command 'Edit Worksheet'}

Function: This command opens the "Edit Worksheet" dialog box in which you set the size of the worksheet.
Call: Main menu "CFC"
Requirements: A CFC editor is active.

\section*{Dialog box 'Edit worksheet'}
\begin{tabular}{|l|l|}
\hline "Use following dimensions" & \begin{tabular}{l} 
Here is where you set the size of the worksheet. Your change is only accepted if \\
the size is sufficient for the existing program.
\end{tabular} \\
\hline "Adapt the dimensions & Automatically adapts the size of the worksheet to the size of your program. \\
automatically" & \begin{tabular}{l} 
Shifts the worksheet along the x or y axis. The input of negative numbers is \\
permitted.
\end{tabular} \\
\hline \begin{tabular}{l} 
"Move the working sheet origin \\
relatively"
\end{tabular} \\
\hline
\end{tabular}

\section*{Command 'Edit Page Size'}

Function: This command opens the "Edit Page Size" dialog box, in which you change the size of the page-oriented CFC editor.
Call: Main menu "CFC"
Requirements: A page-oriented CFC editor is active.

\section*{Dialog box 'Edit \\ page size' \\ dialog box}
\begin{tabular}{|l|l|}
\hline "Width" & \begin{tabular}{l} 
Width of the page (minimum 24, maximum 1024). Elements outside of the \\
working area are marked red.
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Height of the page (minimum 24, maximum 1024). Elements outside of the \\
working area are marked red.
\end{tabular} \\
\hline "Margin width" & Width of the margin (minimum 6, maximum 25\% or page width). \\
\hline \begin{tabular}{l} 
"Set as standard for new CFC \\
objects"
\end{tabular} & \(\boxed{\square}\) : The current settings are selected as standard for new CFC objects. \\
\hline
\end{tabular}

See also
- \({ }^{4}\) Chapter 6.4.1.20.1.7.6.1 "CFC element 'Page'" on page 2109

\section*{Command 'Negate'}

Symbol: -d
Function: This command negates the selected function block input or function block output.
Call: Main menu "CFC", context menu
Requirements: A CFC editor is active. A function block input or function block output is selected.

\section*{Command 'EN/ENO'}

Symbol: \EM
Function: This command adds a boolean input "EN" (Enable) and a boolean output "ENO" (Enable Out) to the selected function block.

Call: Main menu "CFC", context menu
Requirements: A CFC editor is active. A function block is selected.
The added input "EN" activates the function block. The function block is executed only if the input is TRUE. The value of this signal is output at the "ENO" output.

\section*{Command 'None'}

Symbol: - - ; keyboard shortcut : [Ctrl]+[M] (to toggle between "S", "R", "REF", and None)
Function: The command removes a Reset (R), Set (S), or REF from the input of the "Output" element.

Call: Menu bar: "CFC \(\rightarrow\) Set/Reset"; context menu: "Set/Reset"
Requirement: A CFC editor is active. The input of an "Output" element is selected.

\section*{Command 'R (Reset)'}

Symbol: \(\mathbb{R}^{\text {; }}\); keyboard shortcut : [Ctrl]+[M] (to toggle between "S", "R", "REF", and None)
Function: The command adds a Reset to the input of a Boolean "Output" element.
Call: Menu bar: "CFC \(\rightarrow\) Set/Reset"; context menu: "Set/Reset"
Requirement: A CFC editor is active. The input of an "Output" element is selected.
If an "Output" element has a Reset input, then the Boolean output value is set to "FALSE" as soon as the value of the input is "TRUE". The "FALSE" value at the output is retained, even if the input value changes.
See also
- \(\left.\quad \begin{array}{l} \\ y\end{array}\right)\) Chapter 6.4.1.21.3.13.7 "Command 'S (Set)"' on page 2687

\section*{Command 'S (Set)'}

Symbol: si; keyboard shortcut : [Ctrl]+[M] (to toggle between "S", "R", "REF", and none)
Function: The command adds a Set (S) to the input of a Boolean "Output" element.
Call: Menu bar: "CFC \(\rightarrow\) Set/Reset"; context menu: "Set/Reset"
Requirement: A CFC editor is active. The input of an "Output" element is selected.
If an "Output" element has a Set input, then the Boolean output value is set to "TRUE" as soon as the value of the input is "TRUE". The "TRUE" value at the output is retained, even if the input value changes.

See also
- \(y^{\prime}>\) Chapter 6.4.1.21.3.13.6 "Command 'R (Reset)"' on page 2687

\section*{Command 'REF= (Reference Assignment)'}

Symbol: \({ }^{\text {REF }}\); keyboard shortcut : [Ctrl] \(]+[M]\) (to toggle between "S", " \(R\) ", "REF", and None)
Function: The command assigns a reference to an "Output" element.
Call: Menu bar: "CFC \(\rightarrow\) Set/Reset"; context menu: "Set/Reset"
Requirements: A CFC editor is active. The input of an "Output" element is selected.
```

Example: Declaration:
ref_int : REFERENCE TO INT;
a : INT;
CFC:

```


This corresponds to the ST code: ref_int REF= a;

For more information, see the description for the data type REFERENCE TO.
See also
- \({ }^{4}\) Chapter 6.4.1.9.5.3.3 " Programming in the CFC editor" on page 1871
- \(\Leftrightarrow_{\text {® }}\) Chapter 6.4.1.20.5.14 "Reference" on page 2245

\section*{Command 'Display Execution Order'}

Function: The command temporarily shows a numbered tag for all CFC elements of the programming object.

\section*{Call}
- Menu bar: "CFC \(\rightarrow\) Execution Order"
- Context menu in the CFC editor

Requirement: A CFC editor is active and the "Auto Data Flow Mode" property is selected.
The numbers represent the automatically determined execution order. The execution order is determined by data flow. In the case of multiple networks, it is determined by their topological position in the editor.
The tags are hidden as soon as you click in the CFC editor.

See also
- \({ }^{*} \Rightarrow\) Chapter 6.4.1.9.5.3.2 "Automatic Execution Order by Data Flow" on page 1867
- \(\quad\) Chapter 6.4.1.21.3.13.10 "Command 'Set Start of Feedback"' on page 2688


\section*{Command 'Set Start of Feedback'}

Symbol:
Function: The command defines the selected element as the starting point within a feedback.
Call:
- Menu bar: "CFC \(\rightarrow\) Execution Order"
- Context menu: "Execution Order"

Requirement: A CFC editor is active and the "Auto Data Flow Mode" property is selected. Moreover, a network of the CFC POU contains a feedback, and an element within the feedback is selected.

In the CFC editor, the starting point within the feedbacks is decorated with the symbol. Then the element has the lowest number in the execution order within the feedbacks. At runtime, the processing of the feedback begins with this element.

See also
- * \({ }^{*}\) Chapter 6.4.1.9.5.3.2 "Automatic Execution Order by Data Flow" on page 1867
- \({ }^{4}\), Chapter 6.4.1.21.3.13.9 "Command 'Display Execution Order"' on page 2688
- \(\quad\) Chapter 6.4.1.21.4.11.14 "Dialog 'Properties' - 'CFC Execution Order'" on page 2761

\section*{Command 'Send to Front'}

Symbol:

Function: The command numbers the elements so that the selected elements are located at the front of the execution order.

Call: Menu bar: "CFC \(\rightarrow\) Execution Order"; context menu: "Execution Order"
Requirements: A CFC editor is active and the "Explicit Execution Order Mode" property is selected. At least one element is selected.

The selected elements get the lowest numbers beginning at 0 while keeping the previous order. The remaining elements are numbered so that their execution order remains the same. The topological positions of the elements are retained anyway.

See also
- \(\xi^{*}\) Chapter 6.4.1.9.5.3.2 "Automatic Execution Order by Data Flow" on page 1867
- \(\quad \xi^{\circ}\) Chapter 6.4.1.21.4.11.14 "Dialog 'Properties' - 'CFC Execution Order'" on page 2761
- \(y^{2}\) Chapter 6.4.1.21.3.13.12 "Command 'Send to Back"" on page 2689
- \({ }^{〔}\) Chapter 6.4.1.21.3.13.13 "Command 'Move Up"' on page 2689
- \({ }^{4}\) Chapter 6.4.1.21.3.13.14 "Command 'Move Down"" on page 2690
- \({ }^{\wedge}\) Chapter 6.4.1.21.3.13.16 "Command 'Order by Data Flow'" on page 2691
- *y Chapter 6.4.1.21.3.13.17 "Command 'Order by Topology'" on page 2691

\section*{Command 'Send to Back'}

Symbol: 믈
Function: The command numbers the elements so that the selected elements are located at the end of the execution order.

Call: Menu bar: "CFC \(\rightarrow\) Execution Order"; context menu: "Execution Order"
Requirements: A CFC editor is active and the "Explicit Execution Order Mode" property is selected. At least one element is selected.

The selected elements get the highest numbers while keeping the previous order. The remaining elements are numbered so that their execution order remains the same. The topological positions of the elements are retained anyway.

See also
- \(\quad\) Chapter 6.4.1.9.5.3.2 "Automatic Execution Order by Data Flow" on page 1867
- \(\quad\) Chapter 6.4.1.21.4.11.14 "Dialog 'Properties' - 'CFC Execution Order'" on page 2761
- \({ }^{\circ}>\) Chapter 6.4.1.21.3.13.11 "Command 'Send to Front"' on page 2688
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.13.13 "Command 'Move Up"' on page 2689
- \({ }^{4}\) Chapter 6.4.1.21.3.13.14 "Command 'Move Down"" on page 2690
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.13.16 "Command 'Order by Data Flow'" on page 2691
- \({ }^{\Perp}\) Chapter 6.4.1.21.3.13.17 "Command 'Order by Topology'" on page 2691

\section*{Command 'Move Up'}

Symbol: \(\square\)
Function: The command numbers the elements so that the selected elements are located one position forward.
Call: Menu bar: "CFC \(\rightarrow\) Execution Order"; context menu: "Execution Order"
Requirements: A CFC editor is active and at least one element is selected. The "Explicit Execution Order Mode" property is selected.

The selected elements get a numbering decreased by one while keeping the previous order. The selected elements are processed one position earlier. The remaining elements are numbered so that their execution order remains the same. The topological positions of the elements are retained anyway.

See also
- \(y^{*}\) Chapter 6.4.1.9.5.3.2 "Automatic Execution Order by Data Flow" on page 1867
- \(\wedge^{\wedge}\) Chapter 6.4.1.21.4.11.14 "Dialog 'Properties' - 'CFC Execution Order'" on page 2761
- \({ }^{*}>\) Chapter 6.4.1.21.3.13.11 "Command 'Send to Front"" on page 2688
- \({ }^{\bullet}\) © Chapter 6.4.1.21.3.13.12 "Command 'Send to Back"' on page 2689
- \({ }^{4}\) Chapter 6.4.1.21.3.13.14 "Command 'Move Down"' on page 2690
- \(\quad 4\) Chapter 6.4.1.21.3.13.16 "Command 'Order by Data Flow'" on page 2691
- \(\quad\) \& Chapter 6.4.1.21.3.13.17 "Command 'Order by Topology'" on page 2691

\section*{Command 'Move Down'}

Symbol: -4
Function: The command numbers the elements so that the selected elements are located one position backward.

Call: Menu bar: "CFC \(\rightarrow\) Execution Order"; context menu: "Execution Order"
Requirements: A CFC editor is active and at least one element is selected. The "Explicit Execution Order Mode" property is selected.

The selected elements get a numbering increased by one while keeping the previous order. The elements are processed one position later. The remaining elements are numbered so that their execution order remains the same. The topological positions of the elements are retained anyway.

See also
- \(\sum^{*}\) Chapter 6.4.1.9.5.3.2 "Automatic Execution Order by Data Flow" on page 1867
- \({ }^{\Perp}\) Chapter 6.4.1.21.4.11.14 "Dialog 'Properties' - 'CFC Execution Order"" on page 2761
- 宏 Chapter 6.4.1.21.3.13.11 "Command 'Send to Front"' on page 2688

- \({ }^{\bullet}\) Chapter 6.4.1.21.3.13.13 "Command 'Move Up"" on page 2689
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.13.16 "Command 'Order by Data Flow"' on page 2691
- \(\quad\) \& Chapter 6.4.1.21.3.13.17 "Command 'Order by Topology"' on page 2691

\section*{Command 'Set Execution Order'}

Function: The command opens a dialog for setting the number of the selected element to any value.

Call: Menu bar: "CFC \(\rightarrow\) Execution Order"; context menu: "Execution Order"
Requirements: A CFC editor is active and the "Explicit Execution Order Mode" property is selected. Exactly one element is selected.

The selected element gets the number specified in the dialog. The remaining elements are numbered so that their execution order remains the same. The topological positions of the elements are retained anyway.

See also
- \(\boldsymbol{y}^{*}\) Chapter 6.4.1.9.5.3.2 "Automatic Execution Order by Data Flow" on page 1867
- \(\quad\) Chapter 6.4.1.21.4.11.14 "Dialog 'Properties' - 'CFC Execution Order'" on page 2761
- \({ }^{\text {• }}\) - Chapter 6.4.1.21.3.13.11 "Command 'Send to Front"" on page 2688
- \({ }^{\text {• }}\) - Chapter 6.4.1.21.3.13.12 "Command 'Send to Back'" on page 2689
- \({ }^{*}\) Chapter 6.4.1.21.3.13.13 "Command 'Move Up'" on page 2689
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.13.14 "Command 'Move Down"' on page 2690
- \(\&\) Chapter 6.4.1.21.3.13.16 "Command 'Order by Data Flow'" on page 2691
- 祭 Chapter 6.4.1.21.3.13.17 "Command 'Order by Topology'" on page 2691

\section*{Command 'Order by Data Flow'}

Function: The command numbers the elements in the program by data flow, or in the case of multiple networks by their topological position in the editor.

Call: Menu bar: "CFC \(\rightarrow\) Execution Order"; context menu: "Execution Order"
Requirements: A CFC editor is active and the "Explicit Execution Order Mode" property is selected.
The command is also available when no element is selected.

The execution order is determined by data flow. In the case of multiple networks, it is determined by their topological position of the networks. All numbered elements of the POU are set accordingly. Afterwards, the execution order is identical to that in auto data flow mode. The topological positions of the elements are retained anyway.

See also
- \(\quad\), Chapter 6.4.1.9.5.3.2 "Automatic Execution Order by Data Flow" on page 1867
- \({ }^{\circ}\), Chapter 6.4.1.21.4.11.14 "Dialog 'Properties' - 'CFC Execution Order'" on page 2761
- \({ }^{\bullet}>\) Chapter 6.4.1.21.3.13.11 "Command 'Send to Front"' on page 2688
- ② Chapter 6.4.1.21.3.13.12 "Command 'Send to Back'" on page 2689
- \& Chapter 6.4.1.21.3.13.13 "Command 'Move Up"" on page 2689
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.13.14 "Command 'Move Down"' on page 2690
- \(\quad\) © Chapter 6.4.1.21.3.13.17 "Command 'Order by Topology'" on page 2691

\section*{Command 'Order by Topology'}

Function: The command orders the execution order of the elements by their topological position from right to left and from top to bottom.
Call: Menu bar: "CFC \(\rightarrow\) Execution Order"; context menu: "Execution Order"
Requirements: A CFC editor is active and the "Explicit Execution Order Mode" property is selected. At least one element is selected.

The command applies to all elements in the program, even if not all elements are selected when the command is executed. The topological positions of the elements are retained anyway.

See also
- \#y Chapter 6.4.1.9.5.3.2 "Automatic Execution Order by Data Flow" on page 1867
- \({ }^{\circ}\), Chapter 6.4.1.21.4.11.14 "Dialog 'Properties' - 'CFC Execution Order'" on page 2761
- \({ }^{\bullet}>\) Chapter 6.4.1.21.3.13.11 "Command 'Send to Front"' on page 2688
- ② Chapter 6.4.1.21.3.13.12 "Command 'Send to Back"' on page 2689
- \({ }^{〔}\) Chapter 6.4.1.21.3.13.13 "Command 'Move Up"' on page 2689
- \({ }^{4}\) Chapter 6.4.1.21.3.13.14 "Command 'Move Down"" on page 2690
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.13.16 "Command 'Order by Data Flow'" on page 2691

\section*{Command 'Edit Parameters'}

Function: This command opens the "Edit Parameters" dialog box, where you change the constant input parameters of a function block.
Call: Main menu "CFC \(\rightarrow\) Edit Parameters", or "Right-Click \(\rightarrow\) Edit Parameters", click the "Parameter" function block.

Requirements: A CFC editor is active. An instantiated function block has VAR_INPUT CONSTANT variables in its declaration.


CODESYS displays blocks with VAR_INPUT CONSTANT variables by the word "Parameter" in the lower left corner of the block.

Dialog Box 'Edit
Parameters'
\begin{tabular}{|l|l|}
\hline "Parameters" & Name of the variable \\
\hline "Type" & Data type of the variables \\
\hline "Value" & Click into the field to type a value. \\
\hline "Initial Value" & Initialization Value \\
\hline "Category" & \begin{tabular}{l} 
Additional information about the parameters; these values are defined by attrib- \\
utes and cannot be changed in this dialog box.
\end{tabular} \\
\hline "Unit" & \begin{tabular}{l} 
- parameterCategory \\
"Min"
\end{tabular} \\
\hline "Max" & \begin{tabular}{l} 
- parameterUnit \\
\(\bullet\) \\
parameterMinValue
\end{tabular} \\
\hline "Delete Prepared Parameters" & \begin{tabular}{l} 
This command is active when you write a prepared value ("Debug \\
\(\rightarrow\) Write Value").
\end{tabular} \\
\hline
\end{tabular}

When you exit the field and the dialog box by clicking "OK", the value changes are applied to the project.
```

Example of a
block with con-
stant inputs

```
```

FUNCTION_BLOCK FB1

```
FUNCTION_BLOCK FB1
VAR_INPUT CONSTANT
VAR_INPUT CONSTANT
    {attribute 'parameterCategory':='General'}
    {attribute 'parameterCategory':='General'}
    {attribute 'parameterUnit':= 'm/s'}
    {attribute 'parameterUnit':= 'm/s'}
    {attribute 'parameterMinValue':= '0'}
    {attribute 'parameterMinValue':= '0'}
    {attribute 'parameterMaxValue':= '100'}
    {attribute 'parameterMaxValue':= '100'}
    fbin1:INT;
    fbin1:INT;
    fbin2:DWORD:=24354333;
    fbin2:DWORD:=24354333;
    fbin3:STRING:='abc';
    fbin3:STRING:='abc';
END_VAR
```

END_VAR

```

\title{
This functionality and the declaration of variables with keyword VAR INPUT CONSTANT applies only to the CFC editor. In the FBD editor, CODESYS always shows all input parameters on the block, regardless of whether or not they are declared as VAR_INPUT or VAR_INPUT CONSTANT. CODESYS also does not make a distinction about this in \(\overline{\text { text editors. }}\)
}

See also
- \({ }^{\star}\) Chapter 6.4.1.20.1.7.5 "CFC Editor in Online Mode" on page 2103
- \& Chapter 6.4.1.21.3.13.19 "Command 'Save Prepared Parameters to Project'" on page 2693

\section*{Command 'Save Prepared Parameters to Project'}

Function: This command saves the prepared parameter values to the project.
Call: Main menu "CFC".
Requirements: A CFC editor is active. Parameter values of function block instances are changed in online mode. You are in offline mode.
If the values of constants on the controller are different from the values in the application, then this is indicated by a red asterisk next to the parameter field. Clicking "Incur Prepared Parameters" saves the controller values to the application.

See also
- \(\mathrm{m}_{\text {" "Changing of constant input parameters of function block instances" on page } 2105}\)
- \({ }^{\text {• }}\) Chapter 6.4.1.21.3.13.18 "Command 'Edit Parameters"" on page 2692

\section*{Command 'Connect Selected Pins'}

Symbol: \(\sigma\)
Function: The command establishes a connection between the selected pins.
Call: Main menu "CFC", context menu
Requirements: A CFC editor is active. Precisely one output and several inputs are selected.
In order to select the pins you must keep the [CTRL] key pressed while clicking on the pins. Then you execute the command.

See also
- \({ }^{*}>\) Chapter 6.4.1.21.3.13.23 "Command 'Select Connected Pins'" on page 2694

\section*{Command 'Unlock Connection'}

Symbol:
Function: This command unlocks a disabled connection.
Call: Main menu "CFC \(\rightarrow\) Routing", context menu "Routing"
Requirements: A CFC editor is active. A connection or a connection mark is selected.
You obtain a disabled connection if you change the connections of the automatic routing. If you wish to carry out automatic routing again, you must first unlock a disabled connection.


See also
- \({ }^{4}\) Chapter 6.4.1.9.5.3.3 " Programming in the CFC editor" on page 1871
- 鼠 Chapter 6.4.1.20.1.7.6.12 "CFC element 'Connection Mark - Source/Sink"' on page 2112

\section*{Command 'Show Next Collision'}

Function: This command displays the next collision in the editor and marks the place concerned.

Call: Menu menu "CFC \(\rightarrow\) Routing", context menu "Routing"
Requirements: A CFC editor is active and at least one connection with a collision is present.
This function is very useful if you operate with large networks and see only one sub-area. A collision is additionally indicated to you by the red bordered symbol in the top right corner of the editor.

\section*{Command 'Select Connected Pins'}

Symbol: -E; shortcut: [Ctrl]+[Left Arrow], or [Ctrl]+[Right Arrow]
Function: The command selects all pins that are connected to the currently selected line, or connected to the currently selected connection mark in page-oriented CFC.
Call: "CFC" menu; context menu
Requirements: A CFC editor or a page-oriented CFC editor is active. One line and therefore exactly one connection or exactly one connection mark is selected.

See also
- \({ }^{4}\) Chapter 6.4.1.20.1.7.2 "CFC Editor" on page 2098
- \({ }^{4}\) Chapter 6.4.1.20.1.7.3 "CFC editor, page-oriented" on page 2101
- \({ }^{\circ} \boldsymbol{y}\) Chapter 6.4.1.20.1.7.6.12 "CFC element 'Connection Mark - Source/Sink'" on page 2112

\section*{Command 'Reset Pins'}

Symbol: 汘, [Ctrl]+[U]
Function: The command restores the deleted pins of a box.
Call: "CFC \(\rightarrow\) Pins" menu; "Pins" in the context menu
Requirements: A CFC editor is active and a box is selected.
The command restores all inputs and outputs of the box as they are defined in their implementation.
See also
- \({ }^{\circledR}\), Chapter 6.4.1.21.3.13.25 "Command 'Remove Unused Pins'" on page 2694

\section*{Command 'Remove Unused Pins'}

Symbol:

Function: The command removes all unused pins of the selected element.
Call: Menu "CFC \(\rightarrow\) Pins", context menu "Pins"
Requirements: A CFC editor is active. An element is selected.
See also
- \({ }^{\circ} \boldsymbol{y}\) Chapter 6.4.1.21.3.13.24 "Command 'Reset Pins'" on page 2694

\section*{Command 'Add Input Pin'}

Symbol:
Function: The command adds a further input to the selected function block.
Call: Main menu "CFC \(\rightarrow\) Pins", context menu "Pins"
Requirements: A CFC editor is active. A function block is selected.
See also
- \(\sum^{\wedge}\) Chapter 6.4.1.21.3.13.27 "Command 'Add Output Pin'" on page 2695

\section*{Command 'Add Output Pin'}

Symbol: 洊
Function: The command adds a further output to the selected function block.
Call: Main menu "CFC \(\rightarrow\) Pins", context menu "Pins"
Requirements: A CFC editor is active. A suitable function block is selected.
See also
- \({ }^{\text {² }}\), Chapter 6.4.1.21.3.13.26 "Command 'Add Input Pin'" on page 2695

\section*{Command 'Route All Connections'}

Symbol: "
Function: This command cancels all manual changes to the connections in the program and re-establishes the original state.
Call: Main menu "CFC \(\rightarrow\) Routing", context menu "Routing"
Requirements: A CFC editor is active.
CODESYS cannot automatically route connections that are fixed by control points. You must remove the control points before executing the command. Use the "Remove Control Point" command to do this. Furthermore you must disconnect connections that have been changed manually and are marked by the \(\underline{\text {, icon. Use the "Disconnect Connection" command to do this. }}\)
See also
- \(\Rightarrow\) Chapter 6.4.1.21.3.13.29 "Command 'Remove Control Point'" on page 2695
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.13.21 "Command 'Unlock Connection'" on page 2693

\section*{Command 'Remove Control Point'}

Function: This command removes a control point.
Call: Context menu "Routing"
Requirements: A CFC editor is active. You have selected a connecting line.

If you move the mouse pointer over a selected connecting line, the existing control points are displayed with yellow circle symbols. Set the cursor on the control point to be deleted and execute the command from the context menu.

See also
- \(\quad 4\) Chapter 6.4.1.20.1.7.6.2 "CFC element 'Control Point"' on page 2109
- \({ }^{\bullet} \boldsymbol{y}\) Chapter 6.4.1.21.3.13.30 "Command 'Create Control Point"' on page 2696

\section*{Command 'Create Control Point'}

Symbol: +
Function: The command creates a control point on a connecting line.
Call: Context menu "Routing"
Requirements: A CFC editor is active. The cursor is over a connection.
The control point is created in the position on the connection at which the cursor is located when calling the command. The command corresponds to the "Control Point" element in the "Tools" window.

See also
- \({ }^{4} \boldsymbol{y}\) Chapter 6.4.1.20.1.7.6.2 "CFC element 'Control Point"" on page 2109
- \({ }^{*}\) Chapter 6.4.1.21.3.13.29 "Command 'Remove Control Point"' on page 2695

\section*{Command 'Connection Mark'}

Symbol:
Function: This command switches the display of the connection between two elements back and forth between a connecting line and the use of connection marks.

Call: Main menu "CFC", context menu
Requirements: A CFC editor is active. A connection or a connection mark is selected.
If you have selected a connecting line, the command removes this line and adds a "Connection Mark - Source" at the output of one element and a "Connection Mark - Sink" at the input of the other. Both are given the same name by default, " \(\mathrm{C}-<\mathrm{n}>\) ", where n is a sequential number.

If you select a pair of connection marks, the command converts these marks into a connecting line.

See also
- \(\boldsymbol{y y}^{\prime}\) Chapter 6.4.1.20.1.7.6.12 "CFC element 'Connection Mark - Source/Sink'" on page 2112

\section*{Command 'Create group'}

Symbol:向
Function: This command groups the selected elements.
Call: Main menu "CFC \(\rightarrow\) Group", context menu "Group"
Requirements: A CFC editor is active. Several elements are selected.

Grouped elements can only be moved together. The position of the elements is not affected by the grouping.

See also
- \({ }^{*}\), Chapter 6.4.1.21.3.13.33 "Command 'Ungroup'" on page 2697

\section*{Command 'Ungroup'}

Symbol: 面
Function: The command undoes a previous grouping.
Call: Main menu "CFC \(\rightarrow\) Group", context menu "Group"
Requirements: A CFC editor is active. A grouping is selected.
See also
- *y Chapter 6.4.1.21.3.13.32 "Command 'Create group'" on page 2696

\section*{Command 'Prepare Box for Forcing'}

This command is required when using compiler versions 3.5.11.x and 3.5.12.x.
The command is no longer required for compiler versions \(>=3.5 .13 .0\).

Function: The command activates and deactivates the forceability of the inputs for a function block element.

\section*{Call:}
- CFC
- Context menu

Requirements: The CFC editor is in offline mode and a function block element is selected.
After executing the command, the "Force Function Block Input" command is available in online mode to open a dialog for forcing the box input values.

See also
- \({ }^{4}\) Chapter 6.4.1.21.3.13.35 "Command 'Force Function Block Input'" on page 2697
- \(\Leftrightarrow\) Chapter 6.4.1.20.1.7.2 "CFC Editor" on page 2098
- \({ }^{4}\) Chapter 6.4.1.12.5 "Forcing and Writing of Variables" on page 1987

\section*{Command 'Force Function Block Input'}

\section*{NOTICE!}

This kind of forcing uses a data breakpoint internally and is therefore different from forcing with the "Force Values" command or [F7].
Values that were forced by the command "Force FB Input" do not respond to the commands "Show All Forces" or "Unforce Values".

Function: The command opens the "Force Value" dialog to force the selected input of a function block. Forcing can be canceled with the same command and dialog.
Call:
- CFC
- Context menu

\section*{Requirements:}
- The CFC editor is in online mode and the input of the function block is selected.
- For compiler versions 3.5.11.x and 3.5.12.x, the "forceability" of the function block is enabled by the "Prepare Box for Forcing" command.
In the "Force Value" dialog, you can either specify a value that the input of the function block should be forced, or remove the currently forced value.

After forcing, the input is highlighted in green again. Boolean inputs get a small monitoring view with the forced value. The forced value is displayed in the "Value" column of monitoring views (in the declaration part of the POU or in a watch list).

\section*{Dialog 'Force Value'}
\begin{tabular}{|l|l|}
\hline "Expression" & Name of the function block input. Example: TON_1. IN. \\
\hline "Type" & Data type of the input \\
\hline
\end{tabular}

Table 517: "What do you want to do?"
\begin{tabular}{|l|l|}
\hline "Set a new value to force" & \begin{tabular}{l} 
©: You can specify a new value in the input field. The format has to correspond \\
to the data type.
\end{tabular} \\
\hline "Remove value" & O: Forcing at the input is canceled. \\
\hline
\end{tabular}

See also
- \({ }^{4}\) Chapter 6.4.1.20.1.7.2 "CFC Editor" on page 2098
- \({ }^{\bullet}\) Chapter 6.4.1.12.5 "Forcing and Writing of Variables" on page 1987
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.13.34 "Command 'Prepare Box for Forcing'" on page 2697

\section*{Command 'Use Attributed Member as Input'}

Symbol: \(\Downarrow\)
Function: This command allows for connecting a structure member to a scalar type input.
Call: Menu bar: "CFC \(\rightarrow\) Pins"; context menu: "Pins"
Requirements: A CFC editor is active and a function block input is selected.
The member of the structure that is connected to the input of the subsequent function block must be provided with the pragma \{attribute 'ProcessValue'\}. The data type of the structure member has to be compatible with the data type of the subsequent input. Inputs connected in this way are flagged with the " V " symbol.


If you do not execute the command "Use attributed member as input" for this link, then a compiler error is issued.

\section*{See also}
- \(\boldsymbol{y y}^{4}\) Chapter 6.4.1.20.6.3.39 "Attribute 'ProcessValue'" on page 2312

\section*{Menu 'FBD/LD/IL'}
6.4.1.21.3.14.1 Command 'Insert Network'. ..... 2700
6.4.1.21.3.14.2 Command 'Insert Network (Below)' ..... 2701
6.4.1.21.3.14.3 Command 'Toggle Network Comment State' ..... 2701
6.4.1.21.3.14.4 Command 'Insert Assignment'. ..... 2701
6.4.1.21.3.14.5 Command 'Insert Box'. ..... 2701
6.4.1.21.3.14.6 Command 'Insert Box with EN/ENO'. ..... 2702
6.4.1.21.3.14.7 Command 'Insert Empty Box' ..... 2702
6.4.1.21.3.14.8 Command 'Insert Empty Box with EN/ENO'. ..... 2702
6.4.1.21.3.14.9 Command 'Insert Box Parallel (Below)'. ..... 2702
6.4.1.21.3.14.10 Command 'Insert Jump'. ..... 2703
6.4.1.21.3.14.11 Command 'Insert Label'. ..... 2703
6.4.1.21.3.14.12 Command 'Insert Return'. ..... 2703
6.4.1.21.3.14.13 Command 'Insert Input' ..... 2703
6.4.1.21.3.14.14 Command 'Insert Coil'. ..... 2704
6.4.1.21.3.14.15 Command 'Insert Set Coil'. ..... 2704
6.4.1.21.3.14.16 Command 'Insert Reset Coil'. ..... 2704
6.4.1.21.3.14.17 Command 'Insert Contact'. ..... 2704
6.4.1.21.3.14.18 Command 'Insert Contact (Right)' ..... 2705
6.4.1.21.3.14.19 Command 'Insert Contact in Parallel (Below)'. ..... 2705
6.4.1.21.3.14.20 Command 'Insert Contact in Parallel (Above)' ..... 2705
6.4.1.21.3.14.21 Command 'Toggle Parallel Mode' ..... 2706
6.4.1.21.3.14.22 Command 'Insert Negated Contact'. ..... 2706
6.4.1.21.3.14.23 Command 'Insert Negated Contact Parallel (Below)'. ..... 2706
6.4.1.21.3.14.24 Command 'Paste Contacts: Paste Below'. ..... 2707
6.4.1.21.3.14.25 Command 'Paste Contacts: Paste Above'. ..... 2707
6.4.1.21.3.14.26 Command 'Paste Contacts: Paste Right (After)'....... ..... 2707
6.4.1.21.3.14.27 Command 'Insert IL Line Below'. ..... 2707
6.4.1.21.3.14.28 Command 'Delete IL Line'. ..... 2707
6.4.1.21.3.14.29 Command 'Negation'. ..... 2708
6.4.1.21.3.14.30 Command 'Edge Detection'. ..... 2708
6.4.1.21.3.14.31 Command 'Set/Reset'. ..... 2708
6.4.1.21.3.14.32 Command 'Set Output Connection'. ..... 2708
6.4.1.21.3.14.33 Command 'Insert Branch'. ..... 2709
6.4.1.21.3.14.34 Command 'Insert Branch Above'. ..... 2709
6.4.1.21.3.14.35 Command 'Insert Branch Below'. ..... 2709
6.4.1.21.3.14.36 Command 'Set Branch Start Point'. ..... 2709
6.4.1.21.3.14.37 Command 'Set Branch End Point'. ..... 2710
6.4.1.21.3.14.38 Command 'Update Parameters' ..... 2710
6.4.1.21.3.14.39 Command 'Remove Unused FB Call Parameters'. ..... 2710
6.4.1.21.3.14.40 Command 'Repair POU' ..... 2710
6.4.1.21.3.14.41 Command 'View as Function Block Diagram'. ..... 2711
6.4.1.21.3.14.42 Command 'View as Ladder Logic' ..... 2711
6.4.1.21.3.14.43 Command 'View as Instruction List'. ..... 2711
6.4.1.21.3.14.44 Command 'Go to'. ..... 2712

\section*{Command 'Insert Network'}

Function: This command inserts a further network in the FBD/LD/IL editor.
Call: Main menu "FBD, \(L D\), \(I L\) ", context menu

Requirements: The FBD, LD or IL editor is active. No box is selected.
See also
- \({ }^{\mu}\) Chapter 6.4.1.20.1.6.4.1 "FBD/LD/IL element 'Network'" on page 2091
- \({ }^{*}\) Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082
- \(\quad 4\) Chapter 6.4.1.21.3.14.2 "Command 'Insert Network (Below)"' on page 2701

\section*{Command 'Insert Network (Below)'}

Symbol: [通, shortcut: [Ctrl]+ [T]
Function: This command inserts a further network in the FBD/LD/IL editor below the selected network.

Call: Main menu "FBD, LD, IL", context menu
Requirements: The FBD, LD or IL editor is active. A network is selected. No box is selected.
See also
- \(\quad\) \& Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082

\section*{Command 'Toggle Network Comment State'}

Symbol: \((\times x)^{\prime}\), shortcut: [Ctrl] + [O]
Function: The command comments the selected network in or out.
Call: Main menu "FBD, \(L D\), \(I L\) ", context menu
Requirements: The FBD, LD or IL editor is active. A network is selected, but no box is selected.
See also
- \({ }^{\circ}\) Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082

\section*{Command 'Insert Assignment'}

Symbol -vne, shortcut: [Ctrl] + [A]
Function: This command inserts an assignment in the FBD or LD editor.
Call: Main menu "FBD/LD/IL", context menu
Requirements: The FBD, LD or IL editor is active. A network is selected, but no box is selected.


See also
- \({ }^{\mu}\) Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082

\section*{Command 'Insert Box'}

Symbol: 洉, shortcut: [Ctrl] + [B]
Function: This command inserts a box that is available in the project at the end of the selected network.

Call: Main menu "FBD, LD, IL", context menu

Requirements：The FBD，LD or IL editor is active．A network is selected，but no box is selected． If you select this command the input assistant opens，where you can select the desired box．
See also
－＊Chapter 6．4．1．20．1．6．4．2＂FBD／LD／IL element＇Box＂＇on page 2092
－\＆Chapter 6．4．1．20．1．6．1＂FBD／LD／IL Editor＂on page 2082
－\＆Chapter 6．4．1．9．7＂Using input assistance＂on page 1885

\section*{Command＇Insert Box with EN／ENO＇}

Symbol：：苃，shortcut：［Ctrr］＋［Shift ］＋［E］
Function：This command inserts a box with a boolean input＂Enable＂and a boolean output ＂Enable Out＂at the end of the selected network．
Call：Main menu＂FBD，LD，IL＂，context menu
Requirements：the FBD，LD or IL editor is active．A network is selected，but no box is selected．
See also
－\＆Chapter 6．4．1．20．1．6．1＂FBD／LD／IL Editor＂on page 2082
－\(\&\) Chapter 6．4．1．9．7＂Using input assistance＂on page 1885

\section*{Command＇Insert Empty Box＇}

Symbol：洋，shortcut：［Ctrr］＋［Shift］＋［B］
Function：This command inserts an empty function block at the end of the currently selected network．

Call：Main menu＂FBD／LD／LL＂，context menu
Requirements：The FBD，LD or IL editor is active．A network is selected，but no box is selected．
See also
－＊Chapter 6．4．1．20．1．6．1＂FBD／LD／IL Editor＂on page 2082

\section*{Command＇Insert Empty Box with EN／ENO＇}

Symbol：們
Function：The command inserts an empty box with a Boolean input＂Enable＂and a Boolean output＂Enable Out＂at the end of the selected network．
Call：Main menu＂FBD／LD／IL＂，context menu
Requirements：The FBD editor，the IL editor or the LD editor is active．A network must be selected．No other box may be selected．
If＂Enable＂has the value FALSE at the time of the function block call，then the operations defined in the FB are not executed．Otherwise，if＂Enable＂has the value true，these operations are executed．The ENO output acts as a repeater of the EN input．
See also
－\＆Chapter 6．4．1．20．1．6．1＂FBD／LD／IL Editor＂on page 2082

\section*{Command＇Insert Box Parallel（Below）＇}

Function：This command inserts an empty box parallel below the selected function block． Call：Menu bar：＂FBD／LD／L＂；context menu．

Requirements: A box is selected in the LD editor.
See also
- 鼠 Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082

\section*{Command 'Insert Jump'}

Symbol \(\rightarrow\), shortcut: [Ctrl]+[L]
Function: This command inserts a jump element before the selected element.
Call: Main menu "FBD/LD/IL", context menu
Requirements: The FBD, LD or IL editor is active. A connecting line is selected.
See also
- \({ }^{\Perp}\) Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082
- \(\&\) Chapter 6.4.1.20.1.6.4.7 "FBD/LD/IL element 'Jump'" on page 2093

\section*{Command 'Insert Label'}

Symbol: …
Function: This command inserts a jump label into the currently selected network.
Call: Main menu "FBD, LD, IL", context menu
Requirements: The FBD, LD or IL editor is active. A network is selected. No jump label is selected.

See also
- « Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082
- \(\boldsymbol{y}^{2}\) Chapter 6.4.1.20.1.6.4.6 "FBD/LD/IL element 'Label"' on page 2093

\section*{Command 'Insert Return'}

Symbol: बहет I
Function: This command inserts an element "Return" in the selected place.
Call: Main menu "FBD/LD/IL", context menu
Requirements: The FBD, LD or IL editor is active. A box output is selected.
See also
- \(\Rightarrow\) Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082
- \(\Leftrightarrow\) Chapter 6.4.1.20.1.6.4.8 "FBD/LD/IL element 'Return'" on page 2093

\section*{Command 'Insert Input'}

Symbol: 纸, shortcut: [Ctrl]+[Q]
Function: This command adds a further input to an extendable box (ADD, OR, ADD, MUL, SEL) above the selected input.
Call: "FBD/LD/IL" menu
Requirements: The FBD or LD editor is active. An input of a box is selected.
If a box is selected, the command "Append Input" is available in the context menu. The input is inserted at the lower end of the box.

See also
- \(\quad 4\) Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082

\section*{Command 'Insert Coil'}

Symbol: «ァ, shortcut: [Ctrl] + [A]
Function: This command inserts a coil into the network.
Call: Main menu "FBD, LD, IL", context menu
Requirements: The LD editor is active. A network, a coil or a connecting line is selected, but no box is selected.

See also
- \(\quad\) Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082
- « \({ }^{*}\) Chapter 6.4.1.20.1.6.4.12 "LD element 'Coil"' on page 2095

\section*{Command 'Insert Set Coil'}

Symbol: «
Function: This command inserts a set coil into the network.
Call: Main menu "FBD, \(L D\), \(I L\) ", context menu
Requirements: The LD editor is active. A network, a coil or a line is selected, but no box is selected.

See also
- \({ }^{\mu}\) Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082

\section*{Command 'Insert Reset Coil'}

Symbol: 的
Function: This command inserts a reset coil into the network.
Call: Main menu "FBD, \(L D\), \(I L\) ", context menu
Requirements: The LD editor is active. A network, a coil or a line is selected, but no box is selected.

See also
- \(\&\) Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082
- \({ }^{\bullet}>\) Chapter 6.4.1.20.1.6.4.12 "LD element 'Coil"' on page 2095
- ⓨ "Ladder diagram (LD)" on page 1860

\section*{Command 'Insert Contact'}

Symbol \(\sqrt{1}\), shortcut: [Ctrl] + [K]
Function: This command inserts a contact to the left of the selected element.
Call: Main menu "FBD/LD/IL", context menu
Requirements: The LD editor is active. A line or a contact is selected.
See also
- \({ }^{*}\) Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082
- «y Chapter 6.4.1.20.1.6.4.11 "LD element 'Contact'" on page 2094

\section*{Command 'Insert Contact (Right)'}

Symbol: \(\overrightarrow{\text { s. }}\), shortcut: [Ctrl] + [D]
Function: This command inserts a contact to the right of the selected element.
Call: Main menu "FBD/LD/IL", context menu
Requirements: The LD editor is active. A line, a contact or a box is selected.
See also
- \(\quad 4\) Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082
- \(\quad>\) Chapter 6.4.1.20.1.6.4.11 "LD element 'Contact'" on page 2094

\section*{Command 'Insert Contact in Parallel (Below)'}

Symbol: ww ; keyboard shortcut: [Ctrl]+[R]
Function: This command inserts a contact with lines in parallel with and below the selected element.

Call: Menu bar: "FBD/LD/IL"; context menu.
Requirements: The LD editor is active. A line or a contact or a box is selected.

You can program closed parallel branches in a LD network as short circuit evaluation (SCE) or OR constructs. SCE branches are displayed with double vertical lines, and OR branches with single lines. Refer to the help page for "Closed branches".

See also
- \(\Leftrightarrow\) Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082
- \(\Longleftrightarrow\) Chapter 6.4.1.21.3.14.20 "Command 'Insert Contact in Parallel (Above)"' on page 2705
- \(\sum^{*}\) Chapter 6.4.1.20.1.6.4.14 "Closed branch" on page 2096

\section*{Command 'Insert Contact in Parallel (Above)'}

Symbol: 』1; keyboard shortcut: [Ctrl]+[P]
Function: This command inserts a contact with lines in parallel with and above the selected element.

Call: Menu bar: "FBD/LD/IL"; context menu.
Requirements: The LD editor is active. A line, a contact or a box is selected.


You can program closed parallel branches in a LD network as short circuit evaluation (SCE) or OR constructs. SCE branches are displayed with double vertical lines, and OR branches with single lines. Refer to the help page for "Closed branches".

See also
- \(\Rightarrow\) Chapter 6.4.1.20.1.6.4.11 "LD element 'Contact'" on page 2094
- \(\&\) Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082
- \(\quad\) • Chapter 6.4.1.20.1.6.4.14 "Closed branch" on page 2096

\section*{Command 'Toggle Parallel Mode'}

Function: This command toggles a parallel branch between an OR construct and the Short Circuit Evaluation (SCE) .

Call: Menu bar: "FBD/LD/IL"; context menu.
Requirements: The LD editor is active. A vertical line of a parallel branch is selected.

You can program closed parallel branches in a LD network as short circuit evaluation (SCE) or OR constructs. SCE branches are displayed with double vertical lines, and OR branches with single lines. Refer to the help page for "Closed branches".

See also
- \(\quad\) \& Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082
- ⓨ Chapter 6.4.1.20.1.6.4.14 "Closed branch" on page 2096

\section*{Command 'Insert Negated Contact'}

Symbol: 4 , shortcut: [Ctrl] + [K]
Function: This command inserts a negated contact to the left of the selected element.
Call: Main menu "FBD/LD/IL", context menu
Requirements: The LD editor is active. A line or a contact is selected.
See also
- \({ }^{*}\) Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082
- \(\Leftrightarrow\) Chapter 6.4.1.20.1.6.4.11 "LD element 'Contact'" on page 2094

\section*{Command 'Insert Negated Contact Parallel (Below)'}

Symbol: Lud
Function: The command inserts a negated contact with lines in parallel with and below the selected element.
Call: Main menu "FBD/LD/IL", context menu
Requirements: The LD editor is active. A line, a contact or a box is selected.
See also
- \({ }^{4}\) Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082
- \(\sum^{*}\) Chapter 6.4.1.9.5.2.3 "Programming ladder diagrams (LD)" on page 1864

\section*{Command 'Paste Contacts: Paste Below'}

Shortcut: [Ctrl] + [F]
Function: This command inserts a previously copied contact with lines below the selected element.

Call: Main menu "FBD/LD/IL \(\rightarrow\) Paste", context menu
Requirements: the LD editor is active.
See also
- « Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082
- \(y^{4}\) Chapter 6.4.1.20.1.6.4.11 "LD element 'Contact'" on page 2094

\section*{Command 'Paste Contacts: Paste Above'}

Shortcut: [Ctrl] + [F]
Function: This command inserts a previously copied contact with lines above the selected element.

Call: Main menu "FBD/LD/IL \(\rightarrow\) Paste Contacts", context menu
Requirements: the LD editor is active. A line or a contact is selected.
See also
- \(\Rightarrow\) Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082
- \({ }^{\text {• }}\) Chapter 6.4.1.20.1.6.4.11 "LD element 'Contact'" on page 2094

\section*{Command 'Paste Contacts: Paste Right (After)'}

Shortcut: [Ctrl] + [G]
Function: this command inserts a previously copied contact to the right of the selected element.
Call: Main menu "FBD/LD/IL \(\rightarrow\) Paste Contacts", context menu
Requirements: The LD editor is active. A line or a contact is selected.
See also
- \({ }^{4}\) Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082
- \(y^{4}\) Chapter 6.4.1.20.1.6.4.11 "LD element 'Contact"' on page 2094

\section*{Command 'Insert IL Line Below'}

Symbol:
Function: The command inserts an instruction line below the selected line.
Call: Main menu "FBD/LD/IL", context menu
Requirements: The IL editor is active. A line is selected.
See also
- \(\Leftrightarrow\) Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082

\section*{Command 'Delete IL Line'}

Symbol: : shortcut: [Ctrl]+[Del]
Function: This command deletes the selected instruction line.

Call：Main menu＂FBD／LD／IL＂，context menu
Requirements：The IL editor is active．A line is selected．
See also
－\(\quad\) \＆Chapter 6．4．1．20．1．6．1＂FBD／LD／IL Editor＂on page 2082

\section*{Command＇Negation＇}

Symbol：目，shortcut：［Ctrl］＋［N］
Function：This command negates the following elements：
－Input／output of a box
－Jump
－Return
－Coil
Call：Main menu＂FBD／LD／IL＂，context menu
Requirements：the FBD or LD editor is active．The corresponding element is selected．
See also
－ \(\boldsymbol{H}^{4}\) Chapter 6．4．1．20．1．6．1＂FBD／LD／IL Editor＂on page 2082

\section*{Command＇Edge Detection＇}

Symbol FBD：-4 ，symbol LD：园，shortcut：［Ctrl］＋［N］
Function：This command inserts an edge detector before the selected box input or box output．
Call：Main menu＂FBD／LD／IL＂，context menu
Requirements：The FBD or LD editor is active．A box input or box output is selected．
See also
－\(\quad\) \＆Chapter 6．4．1．20．1．6．1＂FBD／LD／IL Editor＂on page 2082

\section*{Command＇Set／Reset＇}

Symbol：－图，shortcut［Ctrl］＋［M］
Function：In the case of an element with a boolean output，this command switches between reset，set and no mark．
Call：Main menu＂FBD／LD／IL＂，context menu
Requirements：The FBD or LD editor is active．An element with a boolean output is selected． See also
－\({ }^{*}\) Chapter 6．4．1．20．1．6．1＂FBD／LD／IL Editor＂on page 2082

\section*{Command＇Set Output Connection＇}

Symbol：파，shortcut［Ctrl］＋［W］
Function：This command turns the selected box output into the forwarding box output．
Call：Main menu＂FBD／LD／IL＂，context menu
Requirements：The FBD or LD editor is active．One of several box outputs is selected．

See also
- \(\boldsymbol{y y}^{4}\) Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082

\section*{Command 'Insert Branch'}

Symbol: 七, shortcut [Ctrl] + [Shift] + [V]
Function: This command creates an open line branch on the selected line.
Call: Main menu "FBD/LD/IL", context menu
Requirements: The FBD or LD editor is active. An input or an output of a box is selected.
See also
- ⓨ Chapter 6.4.1.20.1.6.4.9 "FBD/LD/IL element 'Branch'" on page 2093
- \(\sum_{\text {• }}\) Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082

\section*{Command 'Insert Branch Above'}

Symbol:荤
Function: This command inserts a line branch above the selected open line branch.
Call: Main menu "FBD/LD/IL", context menu
Requirements: The FBD or LD editor is active. An open line branch is selected.
See also
- \(\underbrace{}_{\text {• }}\) Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082
- \(\boldsymbol{y y}^{2}\) Chapter 6.4.1.20.1.6.4.9 "FBD/LD/IL element 'Branch'" on page 2093

\section*{Command 'Insert Branch Below'}

Symbol: \(\overline{5 \times 1}\)
Function: This command inserts a line branch below the selected open line branch.
Call: Main menu "FBD/LD/IL", context menu
Requirements: The FBD or LD editor is active. An open line branch is selected.
See also
- \(\quad\) Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082
- \(\quad\) Chapter 6.4.1.20.1.6.4.9 "FBD/LD/IL element 'Branch'" on page 2093

\section*{Command 'Set Branch Start Point'}

Symbol *
Function: This command sets the starting point of a line branch on the selected line.
Call: Main menu "FBD/LD/IL", context menu
Requirements: The LD editor is active. A line is selected.
See also
- ② Chapter 6.4.1.20.1.6.4.14 "Closed branch" on page 2096
- \({ }^{\star}\) ) Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082

\section*{Command 'Set Branch End Point'}

Symbol *
Function: This command sets the end point of a line branch on the selected line.
Call: Main menu "FBD/LD/IL", context menu
Requirements: The LD editor is active. A line is selected. A starting point of the line branch has been set.

See also
- \({ }^{4}\) Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082
- \({ }^{*}\) Chapter 6.4.1.20.1.6.4.14 "Closed branch" on page 2096

\section*{Command 'Update Parameters'}

Function: This command enters changes to the declaration of the selected element in the diagram.
Call: Main menu "FBD/LD/IL", context menu
Requirements: The FBD, LD or CFC editor is active. A box is selected. An extending change has been made to the declaration.

The command checks whether a box and its declaration in the declaration editor correspond. The change is accepted for the box only if the declaration was extended. Deletions and overwrites are not updated.

See also
- \(\quad \stackrel{y}{c}\) Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082

\section*{Command 'Remove Unused FB Call Parameters'}

\section*{Symbol:}

Function: This command deletes inputs and outputs of the selected box to which no variable and no value were assigned. However, the default inputs and outputs are always retained.
Call: Main menu "FBD/LD/IL", context menu
Requirements: The FBD or LD editor is active. A box is selected. The box has interfaces to which no value is assigned.

See also
- \(\quad\) ² Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082

\section*{Command 'Repair POU'}

Symbol:
Function: This command repairs internal inconsistencies in the selected box.
Call: Main menu "FBD/LD/IL", context menu
Requirements: The FBD or LD editor is active. The defective box is selected. The editor has found internal inconsistencies in the programming module that can possibly be resolved automatically. CODESYS reports the inconsistencies in the Message window.
This situation is conceivable when editing a project that was created with an older programming system version that did not yet handle the inconsistency concerned as an error.
See also
- Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082

\section*{Command 'View as Function Block Diagram'}

\section*{CAUTION!}

Loss of data! An error-free conversion requires syntactically correct code. Otherwise parts of the implementation can be lost.

Shortcut: [Ctrl] + [1]
Function: This command converts the active instruction list or the active ladder diagram into the function block diagram.
Call: Menu "FBD/LD/LL \(\rightarrow\) View"
Requirements: The LD or IL editor is active.
See also
- \& Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082

\section*{Command 'View as Ladder Logic'}

\section*{CAUTION!}

Loss of data! An error-free conversion requires syntactically correct code. Otherwise parts of the implementation can be lost.

Shortcut: [Ctrl] + [2]
Function: This command converts the current function block code or the active instruction list into a ladder diagram.
Call: Menu "FBD/LD/LL \(\rightarrow\) View"
Requirements: The FBD or IL editor is active.
See also
- \({ }^{4}\) Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082

\section*{Command 'View as Instruction List'}

If necessary, IL can be activated in the CODESYS options.

\section*{CAUTION!}

Loss of data! An error-free conversion requires syntactically correct code. Otherwise parts of the implementation can be lost.

Shortcut: [Ctrl] + [3]
Function: This command converts the active function block code or the active ladder diagram into an instruction list.
Call: Menu "FBD/LD/LL \(\rightarrow\) View"
Requirements: The LD or FBD editor is active.

See also
- \(\quad 4\) Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082

\section*{Command 'Go to'}

Symbol:
Function: This command allows you to jump to any network.
Call: Main menu "FBD/LD/IL"
Requirements: The LD, FBD or IL editor is active. A network is selected.
This command opens a dialog box with an input field. Enter the number of the desired network in the input field.
See also
- \(\quad 4\) Chapter 6.4.1.20.1.6.1 "FBD/LD/IL Editor" on page 2082

\section*{Menu 'Library'}
\[
\text { 6.4.1.21.3.15.1 Command 'Add Library'........................................................... } 2712
\]
6.4.1.21.3.15.2 Command 'Try to Reload Library'.......................................... 2713
6.4.1.21.3.15.3 Command 'Properties'........................................................... 2714
6.4.1.21.3.15.4 Command 'Placeholders'...................................................... 2716
6.4.1.21.3.15.5 Command 'Export Library'...................................................... 2716

\section*{Command 'Add Library'}

Function: The command opens the "Add Library" dialog. In this dialog, you can add libraries to the Library Manager and then integrate them in your application.
Call: Menu bar: "Libraries"
Requirement: The Library Manager is open in the editor.

Dialog 'Add
Library'
In the line above the library list, you can search for library names or library modules by typing an appropriate string.
\begin{tabular}{|l|l|}
\hline "Library" & \begin{tabular}{l} 
Suitable libraries that are installed in the library repository. For example, the \\
selection of libraries is defined in the device description or by the system inte- \\
grator. \\
By default, the displayed libraries are grouped into categories.
\end{tabular} \\
\hline "Company" & Vendor of the library \\
\hline "Advanced" & Opens the advanced "Add Library" dialog \\
\hline W & The displayed libraries are grouped into categories. \\
\hline The displayed libraries are listed in alphabetical order. \\
\hline
\end{tabular}

Specific libraries can be blacklisted in a device description. These libraries cannot be added below this device in the Library Manager.

See also
- \(\quad \begin{gathered} \\ \text { - }\end{gathered}\) Chapter 6.4.1.21.3.9.5 "Command 'Library Repository'" on page 2657

Dialog 'Add You should use this dialog only if you have expert knowledge of library referencing. Using this Library' - 'Addialog, you can link special versions or change placeholder definitions. vanced'

We recommend seriously that you follow the guidelines for the creation of libraries when developing and referencing libraries.

Table 518: Tab 'Library'
\begin{tabular}{|c|c|}
\hline "Company" & Filtering the list according to vendor \\
\hline "Group by category" & Display of the libraries in a tree structure grouped in categories.
: Display of the libraries in alphabetical order in a flat structure. \\
\hline "Display all versions" & Display of all versions of the libraries. Version specification '*' means the latest version available in the repository.
\(\qquad\) Display of the latest versions of the libraries only. A multiple selection of libraries is possible in this display. To do this, hold down the [Shift] key and select the entries. \\
\hline "Details" & Opens a detailed view with the library modules. \\
\hline "Library Repository" & Opens the "Library Repository" dialog. There you can install more libraries to your local system. \\
\hline
\end{tabular}

Table 519: Tab 'Placeholder'
\begin{tabular}{|l|l|}
\hline "Placeholder name" & \begin{tabular}{l} 
The input field provides a combo box for entering the valid placeholder names \\
that are read from the currently accessible device descriptions. You can also \\
enter a new placeholder name in order to define a free placeholder, which is not \\
resolved by the device or by the library profile.
\end{tabular} \\
\hline "Default library" & \begin{tabular}{l} 
CODESYS uses this library when for any reason no device is available that the \\
resolution defines. In this way it is possible to compile the current project without \\
errors.
\end{tabular} \\
\hline
\end{tabular}

\section*{Note about placeholder resolution}

For compiler version V3.5.8.0 and later, the following statement applies in the case of library placeholders with a resolution in the device description that are located in the Library Manager of the POU pool. This placeholder is always resolved automatically according to the description of the device that compiles the application.

\section*{See also}
- \({ }^{4}\) ) Chapter 6.4.1.17.3 "Information for Library Developers" on page 2035
- \(y^{4}\) Chapter 6.4.1.21.3.15.4 "Command 'Placeholders"" on page 2716

\section*{Command 'Try to Reload Library' \\ Function: This command tries to reload the selected library. \\ Call: Main menu "Library". \\ Requirement: A library is selected that failed to load.}

If for any reason a library is not available in the defined repository location when a project is opened, CODESYS displays a corresponding error message. Once you have rectified the error, i.e. when the library is properly available again, you can reload the library with this command without having to leave the project.

See also
- \# Chapter 6.4.1.21.3.9.5 "Command 'Library Repository"' on page 2657

\section*{Command 'Properties'}

Function: The command opens the "Properties" dialog for the library selected in the Library Manager.

\section*{Call}
- Menu bar: "Library"
- Context menu of the selected library
- Symbol \(\boldsymbol{\sim}^{-1}\) in the toolbar of the Library Manager

Requirement: A library is selected.

\section*{NOTICE!}

This dialog is intended for library developers. Use this only if you have profound knowledge of library referencing. In addition, follow the guidelines for library developers.

See also
- \({ }^{\text {• }}\) Chapter 6.4.1.17.3 "Information for Library Developers" on page 2035

\section*{Dialog 'Proper- \\ ties'}

Table 520: "General"
\begin{tabular}{|l|l|}
\hline "Namespace" & \begin{tabular}{l} 
Namespace of the selected library. By default, this is identical to the library \\
name, unless it was defined explicitly in the project information when the library \\
was created. You can change the namespace for the open project. \\
Example: LA
\end{tabular} \\
\hline "Default": & \begin{tabular}{l} 
Library that triggers the placeholder when no other trigger is defined or is pos- \\
sible. \\
Requirement: The selected library is a library placeholder, and therefore the \\
setting is available.
\end{tabular} \\
\begin{tabular}{l} 
Note: For compiler version 3.5.8.0 and higher, the following statement applies \\
in the case of library placeholders with a resolution in the device description \\
that are located in the Library Manager of the "POUs" view. This placeholder \\
is always resolved automatically according to the description of the device that \\
compiles the application.
\end{tabular} \\
\hline
\end{tabular}


If the selected library is developed in compliance with the "Guidelines for Developing Libraries", then we do not recommend that you change the following settings.

Table 521: "Version"
Selection of version constraint
Requirement: The settings are available only if the selected library is not a library placeholder.
Note: Container and interface libraries are created automatically with library references with version constraint. As far as possible, do not create libraries that include library references with version constraint. Otherwise, you reference the libraries by placeholders. Edit a placeholder resolution in the "Placeholders" dialog.
\begin{tabular}{|l|l|}
\hline "Exact version" & \begin{tabular}{l} 
O: (selected from list box) Version is integrated into the project. \\
Note: This option is strongly recommended for container libraries, and it is usu- \\
ally preset for this library type.
\end{tabular} \\
\hline "Always newest version" & \begin{tabular}{l} 
O: The library repository is scanned and the latest detected version is inte- \\
grated. \\
Note: If a newer library version is available, then the library POUs that are \\
actually used can change. This option is strongly recommended for interface \\
libraries, and it is usually preset for this library type.
\end{tabular} \\
\hline
\end{tabular}

Table 522: "Visibility"
\(\left.\begin{array}{|l|l|}\hline \text { "Allow only qualified access to } \\
\text { all identifiers" }\end{array} \begin{array}{l}\text { "Wibrary POUs (and variables) are called in the project only with prepended } \\
\text { namespace paths. } \\
\text { "When the current project is } \\
\text { referenced as a library in } \\
\text { another project" }\end{array} \quad \begin{array}{l}\text { Note: Changing the following settings makes sense only if you created a library } \\
\text { with your project and therefore opened a library project. In this way, the selected } \\
\text { library is referenced in the new library. }\end{array}\right]\)\begin{tabular}{ll} 
"Make visible all IEC symbols \\
in the project if is this reference \\
were directly integrated here." As a container library, the selected library makes the contents of the refer- \\
enced library visible at the top level (later in a project). \\
Requirement: A container project is created with a library project. A container \\
library does not implement its own POUs, but references other libraries exclu- \\
sively. It bundles libraries. A container library can be employed sensibly to \\
bundle multiple libraries (in a reference) in a project. This option must be acti- \\
vated for each library reference. \\
Symbolic access to library POUs: <namespace of container
\end{tabular}
"Optional (if the library is missing, no error will be reported)."
: The selected library is treated as optional. When downloading the project that references the library, no error is reported, even if the library is not available in the library repository.

See also
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.15.4 "Command 'Placeholders"' on page 2716

\section*{Command 'Placeholders'}

Function: This command opens the "Placeholders" dialog box. The dialog shows information on the currently selected placeholder library and allows to assign a project-specific resolution.

Call:
- Menu "Libraries"
- Symbol - \(_{\text {- }}\) in the symbol bar in the upper part of the Library Manager window.

Requirement: A placeholder library is selected in the Library Manager.
A placeholder library, which is included in the "Devices" view, will be resolved as follows:
- If you have assigned a specific resolution to the placeholder library via the dialog "Placeholder", this will be applied.
- If no specific resolution is defined, it will be checked, whether there is one specified in the device description of the application..
- Afterwards the library profile will be checked for a resolution definition.
- The result is displayed in the Library Manager below the "Effective Version".

A placeholder library, which is included in the "POUs" view, gets resolved as follows:
- A specific resolution defined in the dialog "Placeholder" will be ignored.
- For the application it will be checked whether there is a resolution defined in the device description.
- Afterwards the library profile will be checked.
- The result is displayed in the tooltip of the symbol .

See also
- \({ }^{4}\) Chapter 6.4.1.17 "Using Libraries" on page 2034
- \(\Leftrightarrow\) Chapter 6.4.1.21.2.16 "Object 'Library Manager'" on page 2469
- \({ }^{*}\) Chapter 6.4.1.17.3 "Information for Library Developers" on page 2035

Dialog box 'Pla-
ceholders'
\begin{tabular}{|l|l|}
\hline "Name" & Name of the placeholder. \\
\hline "Library" & \begin{tabular}{l} 
Current resolution, valid for the project \\
Double-click on the entry in order to edit the placeholder resolution. A selection \\
list with the available library versions appears. Additionally the command "Other \\
Library" is available.
\end{tabular} \\
\hline Command "Other Library" & \begin{tabular}{l} 
The command opens the dialog box "Bibliothek durchsuchen" for searching \\
and installing libraries. Choose this command, if you do not want to redirect \\
to another version, but on a specific libray.
\end{tabular} \\
\hline "Info" & \begin{tabular}{l} 
Type of placeholder resolution: \\
\(\bullet\) \\
- Resolved by device description
\end{tabular} \\
\hline & Resolved by library profile \\
- Resolved by <specific library>
\end{tabular}

\section*{Command 'Export Library'}

Function: This command is used for saving the library file to the hard disk.
Call: Context menu of the Library Manager
Requirement: A library is selected in the Library Manager.
The command opens the standard dialog for saving a file in the local file system. The library file can have the file type Library files (*.library), Compiled library files (*.compiled-library), or Compiled library files (*.compiled-library-v3).

See also
- \(\boldsymbol{y}^{*}\) Chapter 6.4.1.21.2.16 "Object 'Library Manager'" on page 2469
- \(\quad\) Chapter 6.4.1.17.6 "Exporting library files" on page 2037

\section*{Menu 'Image Pool'}
\[
\text { 6.4.1.21.3.16.1 Command 'Insert Image'......................................................... } 2717
\]

\section*{Command 'Insert Image'}

Symbol: 函
Function: This command inserts a new line into an image pool.
Call: Main menu "Imagepool", or right-click.
Requirements: An image pool is active and a line is selected in the image pool.
See also
- ⓨ Chapter 6.4.1.21.2.15 "Object 'Image Pool"' on page 2468

\section*{Menu 'Declarations'}
6.4.1.21.3.17.1 Command 'Insert'. ..... 2717
6.4.1.21.3.17.2 Command 'Edit Declaration Header' ..... 2717
6.4.1.21.3.17.3 Command 'Move Down' ..... 2718
6.4.1.21.3.17.4 Command 'Move Up' ..... 2718

\section*{Command 'Insert'}

Symbol
Function: This command inserts a new line for a variable declaration in the declaration editor and the input field for the variable name opens.
Call: Context menu in the tabular declaration editor; button in the declaration heading.
To edit the other fields of the declaration lines, double-click the fields and select the data from the drop-down lists or by means of the respective dialogs.

See also
- \(\quad\) Chapter 6.4.1.9.4.2 "Using the declaration editor" on page 1851

\section*{Command 'Edit Declaration Header'}

Function: The command opens the dialog "Edit Declaration Header", which serves in the declaration editor for the configuration of a POU header.

Call: Context menu of the tabular declaration editor
Requirements: The tabular declaration editor is the active editor.
See also
- (y) "Declaring in the tabular declaration editor" on page 1852

Dialog 'Edit Dec- Function: The dialog is for configuring the declaration part of a POU.
laration Header'
Call: Click on the header bar of the tabular declaration editor, or context menu in the tabular declaration editor.
\begin{tabular}{|c|c|}
\hline "Declaration" & \begin{tabular}{l}
Selection list for changing the POU type \\
- "PROGRAM" \\
- "FUNCTION_BLOCK" \\
- "EXTENDS": \\
Input field for a basic function block \\
- "IMPLEMENTS": Input field for an interface \\
- "FUNCTION" \\
- "Return type" \\
Input field with current POU name: you can change the name of the POU
\end{tabular} \\
\hline "Automatically adapt all references on rename" & Dialog box "Refactoring" opens.
: Renaming is only effective in the declaration header of the POU. \\
\hline "Attributes" & The dialog box "Attribute "opens for the input of attributes and pragmas. \\
\hline
\end{tabular}

See also
- \({ }^{4}\) Chapter 6.4.1.9.4.2 "Using the declaration editor" on page 1851
- \(\xi^{\star}\) Chapter 6.4.1.20.6 "Pragmas" on page 2269
- \({ }^{*}\) Chapter 6.4.1.9.16 "Refactoring" on page 1910

\section*{Command 'Move Down'}

Symbol:
Function: This command shifts a variable declaration downwards by one row.
Call: Context menu
Requirement: A row with a variable declaration is selected in the tabular declaration editor.
See also
- \(\quad\) Chapter 6.4.1.9.4.2 "Using the declaration editor" on page 1851

\section*{Command 'Move Up'}

Symbol:
Function: This command shifts a variable declaration upwards by one row.
Call: Context menu
Requirement: A row with a variable declaration is selected in the tabular declaration editor.
See also
- \(\Leftrightarrow\) Chapter 6.4.1.9.4.2 "Using the declaration editor" on page 1851

\section*{Menu 'Declarations' (Persistence)}
6.4.1.21.3.18.1 Command 'Reorder List and Clean Gaps'............................. 2719
6.4.1.21.3.18.2 Command 'Save Current Values to Recipe'........................... 2719
6.4.1.21.3.18.3 Command 'Restore Values from Recipe'............................... 2719
6.4.1.21.3.18.4 Command 'Add all instance paths'....................................... 2720

\section*{Command 'Reorder List and Clean Gaps'}

Function: This command cleans the gaps that can result when you make changes to the declaration of persistent variables. The memory requirement is reduced by this cleaning. When the command is executed, CODESYS displays a warning informing the user about the possible loss of data.
Call: Main menu "Declarations", context menu
Requirement: The persistence editor (persistent variable list) is active.
Before cleaning you should consider saving the current values of the persistent variables to a recipe (command "Save Current Values to Recipe"). Then you can load the values to the controller again after the next download.
See also
- \({ }^{*}\) Chapter 6.4.1.9.19 "Data Persistence" on page 1920
- ② Chapter 6.4.1.21.3.18.3 "Command 'Restore Values from Recipe"' on page 2719
- ② Chapter 6.4.1.21.3.18.2 "Command 'Save Current Values to Recipe'" on page 2719

\section*{Command 'Save Current Values to Recipe'}

Function: This command creates a new recipe definition in the recipe manager and stores the current values of the persistent variables in it. You should execute this command before the command "Reorder List and Clear Gaps" in order to avoid a possible loss of data. You can subsequently restore the data with the command "Restore Values from Recipe".
Call: Main menu "Deklarationen"
Requirement: The application is in online mode and the persistence editor (persistent variable list) is active.

> If a list already exists in the recipe manager with the corresponding names when saving a persistent variable list, then the current persistent variables are sorted into the list:
> - New persistent variables are added to the list
> - Variables, that are not in the list, will be deleted
> Therefore, it is possible to add more recipes to the list in the recipe manager and these will be retained. However, if new variables are added to the list, then these are deleted the next time the command "Save Current Values to Recipe" is executed.

See also
- \(\sum^{*}\) Chapter 6.4.1.9.19 "Data Persistence" on page 1920
- \({ }^{*}>\) Chapter 6.4.1.21.3.18.1 "Command 'Reorder List and Clean Gaps"" on page 2719
- \(\boldsymbol{y y}^{*}\) Chapter 6.4.1.21.3.18.3 "Command 'Restore Values from Recipe"' on page 2719
- \(\quad\) を Chapter 6.4.1.21.2.20 "Object 'Persistent variable list"' on page 2476

\section*{Command 'Restore Values from Recipe'}

Function: This command restores the values of the persistent variables that you have stored in a recipe using the command "Save Current Values to Recipe". You would normally select this command after executing the command "Reorder List and Clear Gaps".
Call: Main menu "Declarations"
Requirement: The persistence editor (persistent variable list) is active, the application is in online mode

\section*{See also}
- \(\underbrace{*}\) Chapter 6.4.1.9.19 "Data Persistence" on page 1920
- y Chapter 6.4.1.21.3.18.2 "Command 'Save Current Values to Recipe"' on page 2719
- \(\&\) Chapter 6.4.1.21.3.18.1 "Command 'Reorder List and Clean Gaps'" on page 2719
- \(\quad\) © Chapter 6.4.1.21.2.20 "Object 'Persistent variable list"' on page 2476
- \({ }^{\text {• }}\), Chapter 6.4.1.21.3.18.2 "Command 'Save Current Values to Recipe"' on page 2719

\section*{Command 'Add all instance paths'}

\section*{Function:}
- When you execute the command in the persistence editor, the application is searched for declarations of persistent variables with the PERSISTENT keyword which are outside of the persistence editor. For each declaration found, an instance path of this variable is added in the persistence editor.
- When you execute the command in a variable configuration, an instance path is added for each variable with an incomplete address. All function blocks of the application are considered in this case.

Call: Menu bar: "Declarations", right-click.
Requirement
- the persistence editor (global persistent variable list) is active or a variable configuration (global variable list with VAR_CONFIG declarations) is opened.
- The application was compiled successfully.

See also
- « Chapter 6.4.1.20.2.13 "Persistent Variable - PERSISTENT" on page 2122
- \(\quad\) Chapter 6.4.1.20.2.14 "Retain Variable - RETAIN" on page 2124

Menu 'Device Communication', Gateway
6.4.1.21.3.19.1 Command 'Add New Gateway'.............................................. 2720
6.4.1.21.3.19.2 Command 'Configure the Local Gateway'............................. 2721

\section*{Command 'Add New Gateway'}

Function: This command opens the "Gateway" dialog where you can define a gateway channel and add it to the current device configuration.

Call: Menu bar: "Gateway" in the "Communication Settings" dialog of the device editor.

Dialog 'Gateway'
\begin{tabular}{|l|l|}
\hline "Name" & Name of the gateway. \\
\hline "Driver" & Driver type from a drop-down list. \\
\hline \begin{tabular}{l} 
Driver-specific settings, for \\
example: \\
IP address, port
\end{tabular} & \begin{tabular}{l} 
Editable after double-clicking the predefined value. A short description for each \\
parameter is displayed in the lower part of the dialog. \\
Note: You can also specify the address of a DNS domain. This has to begin with \\
dns: (example: dns:MyDynDNSAdress).
\end{tabular} \\
\hline
\end{tabular}

The dialog is also used for later editing of the gateway entries of your project.
See also
- \({ }^{\star}\), Chapter 6.4.1.21.2.8.3 "Tab 'Communication Settings'" on page 2427

\section*{Command 'Configure the Local Gateway'}

Function: The command opens the "Gateway Configuration" dialog where you can configure the block drivers for the local gateway. This is an alternative to manually editing the configuration file Gateway.cfg.

Call: Context menu when a gateway entry is selected in the device editor in the "Communication Settings" dialog.

\section*{NOTICE!}

A correct configuration of the gateway requires detailed knowledge. In case you have any doubts, do not change the default configuration settings.

Dialog 'Gateway The configuration tree displayed in the dialog corresponds to the description currently valid Configuration' configuration file gateway.cfg. It displays the parameters with the current settings for the interfaces involved. Changes to the configuration in the dialog, confirmed by clicking "OK" result in the direct update of the configuration file.

After the gateway configuration file gateway.cfg has been changed, the gateway has to be restarted in order for the changes to be applied.
\begin{tabular}{|l|l|}
\hline "Add" & \begin{tabular}{l} 
Menu with commands for adding interfaces and settings. The commands are \\
also available in the context menu of the dialog. The selection depends on which \\
entry is selected and which settings have already been added:
\end{tabular} \\
& \begin{tabular}{l} 
"Add Interface": Select an interface for communication via the gateway. It is \\
inserted at the top level of the tree. See the table below for the possible block \\
driver interfaces. \\
"Add Configuration Setting": Select a setting for the selected interface. It is \\
inserted below the interface in the tree. To edit the value of the setting, double- \\
click in the "Setting" column to open an editing field. See the table below for the \\
possible settings per block driver interface.
\end{tabular} \\
\hline "Delete" & Deletes the selected configuration setting \\
\hline "Up", "Down" & Moves the selected configuration entry one position up or down. \\
\hline
\end{tabular}

Table 523: Possible block driver interfaces
\begin{tabular}{|c|c|}
\hline "COM Port" & Serial port on the device, for example for data exchange according to the RS-232 standard on a COM port intended for this purpose. \\
\hline & \begin{tabular}{l}
Possible configuration settings: \\
"Name": Symbolic only \\
"Port": Physical serial port which is used for this interface, for example COM 5 on a Windows computer \\
"Baudrate": \(2400,4800,9600,19200,38400,57600,115200\) \\
"Activate auto addressing": \(\square\) (default \(=\square\) ) The setting Local address is evaluated. Both devices, which communicate via the serial port, will negotiate their addresses independently before they begin exchanging messages. If the addresses of both devices are the same, then they are negotiated again. This setting is useful when the local addresses cannot be set explicitly, for example for physically separated devices. \\
"Local address": Evaluated only when "Enable auto addressing" is activated. Default = actual value for port
\end{tabular} \\
\hline "Shared Memory": & Shared memory driver \\
\hline & \begin{tabular}{l}
Possible settings: \\
"Name": Symbolic only \\
"Forced address": Default = -1 (= no forced address); example: 42 means that the driver has to use the fixed address defined here and that addresses are assigned freely in the range \(0-255\). This setting can be useful when more than one shared memory driver is activated in the configuration.
\end{tabular} \\
\hline "Ethernet UDP/IP": & Ethernet interface for data exchange according to the "user datagram protocol". \\
\hline & \begin{tabular}{l}
Possible settings: \\
"Name": Symbolic only \\
"Port index": Port number for the communication. Port indices are in the range \(0-3\). They are mapped to the following Ethernet port: 1740 to 1743. \\
"IP address": Default = 127.0.0.1. This setting can be useful to explicit set an interface when the device has several network interfaces. Example: 127.0.0.1 stands for some local network interface, also known as localhost. Every other address (example: 10.27.7.72) represents a real IP address which has to be available on the device. \\
"Network mask": Default = 255.255.255.0; example: \\
255.255.252.0. This setting can be useful to explicitly set an interface when there are multiple network interfaces on the device. \\
"PPP remote address": Default = 127.0.0.1; example: \\
10.13.42.240; establishes a logical point-to-point connection between the UDP interface and the node named with the address specified here; has the effect that the UDP interface communicates exclusively with this node and that no broadcasts are sent in the network
\end{tabular} \\
\hline "Ethernet TCP/IP": & Ethernet interface for data exchange according to the "Transmission Control Protocol". \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline & \begin{tabular}{l} 
Possible settings: \\
"Name", "Port", "IP address": See Ethernet UDP/IP above. \\
\\
"Inactivity timeout": Default = 0. This setting defines the time \\
span (in seconds) after which the TCP connections are closed \\
when data is no longer exchanged.
\end{tabular} \\
\hline "CAN Client" & \begin{tabular}{l} 
"Name": Symbolic only \\
A description for the other settings can be found directly in the \\
dialog.
\end{tabular} \\
\hline "USB Port" & \begin{tabular}{l} 
"Name": Symbolic only \\
A description for the other settings can be found directly in the \\
dialog.
\end{tabular} \\
\hline
\end{tabular}

See also:
- \(\quad\), Chapter 6.4.1.21.2.8.3 "Tab 'Communication Settings'" on page 2427

\section*{Menu 'Recipes'}
6.4.1.21.3.20.1 Command 'Insert Variable'. ..... 2723
6.4.1.21.3.20.2 Command 'Add a New Recipe' ..... 2723
6.4.1.21.3.20.3 Command 'Remove Recipe'. ..... 2724
6.4.1.21.3.20.4 Command 'Load Recipe' ..... 2724
6.4.1.21.3.20.5 Command 'Save Recipe' ..... 2724
6.4.1.21.3.20.6 Command 'Read Recipe'. ..... 2725
6.4.1.21.3.20.7 Command 'Write Recipe'. ..... 2725
6.4.1.21.3.20.8 Command 'Load and Write Recipe' ..... 2725
6.4.1.21.3.20.9 Command 'Read and Save Recipe' ..... 2726
6.4.1.21.3.20.10 Command 'Remove Variables'. ..... 2726
6.4.1.21.3.20.11 Command 'Load Recipes from Device' ..... 2727
6.4.1.21.3.20.12 Command 'Update Structured Variables'. ..... 2727

\section*{Command 'Insert Variable'}

Symbol: \(\#\)
Function: This command inserts a variable into the currently opened recipe definition before the selected position.
Call: Main menu "Recipes".
Requirement: You have opened a recipe definition in the editor and selected the normal view.
CODESYS inserts the default text "NewVariable" in the column "Variable". You must replace this name with the respective variable name. To do this, open the input assistant by clicking \(\ldots\) or enter the variable name directly into the table element.

See also
- \({ }^{4}\) Chapter 6.4.1.13.2.4 "Changing Values with Recipes" on page 2003
- \({ }^{2}\) Chapter 6.4.1.21.3.20 "Menu 'Recipes'" on page 2723

\section*{Command 'Add a New Recipe'}

Symbol: \&
Function: This command opens a dialog box for adding a new recipe (new column) to the recipe definition.

\section*{Call: Main menu "Recipes".}

Requirement: You have opened a recipe definition in the editor.
After choosing the command, a dialog box opens for you to define the name of the new recipe. The dialog box also provides the capability of copying existing recipes into the new recipe.
See also
- \({ }^{4}\) Chapter 6.4.1.13.2.4 "Changing Values with Recipes" on page 2003
- \({ }^{4}\) Chapter 6.4.1.21.3.20 "Menu 'Recipes'" on page 2723

\section*{Command 'Remove Recipe'}

Symbol: \&
Function: This command removes a recipe from the currently opened recipe definition.
Call: Main menu "Recipes".
Requirement: You have selected a field in the recipe column of a recipe definition.
See also
- \(\Rightarrow\) Chapter 6.4.1.13.2.4 "Changing Values with Recipes" on page 2003
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.20 "Menu 'Recipes"' on page 2723

\section*{Command 'Load Recipe'}

Symbol: \({ }^{x}\)
Function: The command loads a recipe from a file.
Call: Menu bar: "Recipes".
Requirement: You have selected a field in the recipe column of a recipe definition.
This command overwrites the values of the selected recipe of the recipe definition.
If you have selected the option "Recipe Management in the PLC", please note
the following.
If you change recipes in the project by choosing the command "Load Recipe" or
"Read Recipe", then an online change is required when logging in again.

If you want to overwrite only individual recipe variables with new values, then remove the values for the other variables before loading to the recipe file. Entries without value definitions are not read, and therefore updating leaves these variables unchanged on the PLC and in the project.
For values of the data type REAL/LREAL, the hexadecimal value is also written to the recipe file in some cases. This is necessary so that the exact identical value is restored when converting back. In this case, change the decimal value and delete the hexadecimal value.

\section*{See also}
- \(\Rightarrow\) Chapter 6.4.1.13.2.4 "Changing Values with Recipes" on page 2003
- \(\quad\) © Chapter 6.4.1.21.3.20 "Menu 'Recipes'" on page 2723

\section*{Command 'Save Recipe'}

Symbol:

Function: This command saves the variable values of a recipe to a file.
Call: Main menu "Recipes".
Requirement: You have selected the value of a recipe in the recipe definition.
When you choose this command, CODESYS saves the values of the selected recipe to a file. You can define the format in the settings for the recipe manager in the tab "Storage".
See also
- \(\Rightarrow\) Chapter 6.4.1.13.2.4 "Changing Values with Recipes" on page 2003
- \(\Rightarrow\) Chapter 6.4.1.21.3.20 "Menu 'Recipes'" on page 2723

\section*{Command 'Read Recipe'}

Symbol: \({ }^{\top}\)
Function: This command reads the variable values of a recipe from the controller.
Call: Main menu "Recipes".
Requirement: The application is in online mode and you have selected the value of a recipe in the recipe definition.

When you choose this command, CODESYS overwrites the values of the selected recipe with the read values from the controller.

If you have selected the option "Recipe Management in the PLC", please note the following.
If you change recipes in the project by choosing the command "Load Recipe" or "Read Recipe", then an online change is required when logging in again.

See also
- \(\quad\) Chapter 6.4.1.13.2.4 "Changing Values with Recipes" on page 2003
- \(\quad \Rightarrow\) Chapter 6.4.1.21.3 "Menu Commands" on page 2551

\section*{Command 'Write Recipe'}

\section*{Symbol:}

Function: This command writes the values of a recipe to the variables in the controller.
Call: Main menu "Recipes".
Requirement: The application is in online mode and you have selected the value of a recipe in the recipe definition.

When you choose this command, CODESYS overwrites the values in the controller with the values of the selected recipe.

See also
- \({ }^{*}\) Chapter 6.4.1.13.2.4 "Changing Values with Recipes" on page 2003
- \({ }^{4}\) Chapter 6.4.1.21.3.20 "Menu 'Recipes'" on page 2723

\section*{Command 'Load and Write Recipe'}

Symbol:
Function: This command loads a recipe from a file and writes the values to the variables in the PLC.

Call: Menu bar: "Recipes".

Requirement: The application is in online mode. You have selected the value of a recipe in the recipe definition.
After choosing the command, you are prompted either to write the variable values also to the recipe in the project or only to write them to the PLC. Updating the values in the recipe could require an online change when logging in again.
When you choose this command, CODESYS overwrites the values of the selected recipe of the recipe definition. In addition, these recipe values overwrite the variable values in the PLC.

If you want to overwrite only individual recipe variables with new values, then remove the values for the other variables before loading to the recipe file. Entries without value definitions are not read, and therefore updating leaves these variables unchanged on the PLC and in the project.
For values of the data type REAL/LREAL, the hexadecimal value is also written to the recipe file in some cases. This is necessary so that the exact identical value is restored when converting back. In this case, change the decimal value and delete the hexadecimal value.

See also
- \({ }^{\Perp}\) Chapter 6.4.1.13.2.4 "Changing Values with Recipes" on page 2003
- 党 Chapter 6.4.1.21.3.20 "Menu 'Recipes'" on page 2723

\section*{Command 'Read and Save Recipe'}

Symbol:
Function: This command reads the variable values of a recipe from the controller and saves them to a file.

Call: Main menu "Recipes".
Requirement: The application is in online mode and you have selected the value of a recipe in the recipe definition.

After choosing the command, you are prompted either to read the variable values to the recipe or only to save them. Updating the values in the recipe could require an online change when logging in again.
The values are saved with the default name for recipe files according to the settings for the recipe manager (tab "Storage").
See also
- \(\Leftrightarrow\) Chapter 6.4.1.13.2.4 "Changing Values with Recipes" on page 2003
- ※ Chapter 6.4.1.21.3.20 "Menu 'Recipes'" on page 2723

\section*{Command 'Remove Variables'}

\section*{Symbol -}

Function: This command removes the selected variables from a "Recipe Definition".
Call: The command is not in any menu by default. You can add it to a menu by using the dialog box from "Tools \(\rightarrow\) Customize" (command category "Recipe").
See also
- \&y Chapter 6.4.1.13.2.4 "Changing Values with Recipes" on page 2003
- 学 Chapter 6.4.1.21.3.20 "Menu 'Recipes'" on page 2723

\section*{Command 'Load Recipes from Device'}

\section*{Symbol: B.}

Function: This command initiates the synchronization of the recipes from the open recipe definition in the project and the recipes located on the device in the form of recipe files.

Call: Menu bar: "Recipes".
Requirement: The application is in online mode and a recipe definition is open in the editor.
In detail, the synchronization is described as follows:
- The current values for the recipe variables located in the project are overwritten by the values from the recipes on the controller. As a result, there is likely an online change at the next login.
- If recipe variables are defined in the recipe files on the controller, and the recipe variables are missing in the recipe definition of the project, then these variables are ignored when at the time of download. Before that, a message appears for each recipe file regarding the variables in question.
- If recipe variables are missing in the recipe files on the controller, and these recipe variables are included in the recipe definition of the project, then a message appears for each recipe file with the variables in question.
- If more recipes for these variables have been created on the controller, then they are added to the recipe definition in the project.

\section*{Command 'Update Structured Variables'}

\section*{Symbol:}

Function: This command opens the "Update Structured Variables" dialog box.
Call: Main menu "Recipes".
In this dialog box, you can update recipe definitions if the declaration of a structured variable or a block has changed. For example, if the dimension of an array is changed, then you can automatically add or remove the entries in the recipe definition.

Table 524: Dialog Box 'Update Structured Variables'
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
"Remove not existing \\
variables"
\end{tabular} & \begin{tabular}{l}
\(\checkmark\) \\
the pariables are removed from the recipe definition when they no longer exist in \\
the change to a structured element.
\end{tabular} \\
\hline \begin{tabular}{l} 
"Update instances of structures \\
and function blocks"
\end{tabular} & \begin{tabular}{l} 
V: If the declaration of a structure or function block is extended and available in \\
the recipe definition with an instance, then the respective variables are added to \\
the recipe definition.
\end{tabular} \\
\hline "Update array dimensions of \\
array instances" & \begin{tabular}{l}
\(\checkmark:\) If the dimension of an array is extended and available in the recipe definition \\
with an instance, then the respective variables are added to the recipe definition.
\end{tabular} \\
\hline \begin{tabular}{l} 
"Update contained global \\
variable lists"
\end{tabular} & \begin{tabular}{l}
\(\checkmark:\) If the declaration of a global variable list is extended and available in the \\
recipe definition with an instance, then the respective variables are added to the \\
recipe definition.
\end{tabular} \\
\hline "Update contained programs" & \begin{tabular}{l}
\(\checkmark:\) If the declaration of a program is extended and instanced in the recipe \\
definition, then the respective variables are added to the recipe definition.
\end{tabular} \\
\hline
\end{tabular}

See also
- « Chapter 6.4.1.13.2.4 "Changing Values with Recipes" on page 2003
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.20 "Menu 'Recipes'" on page 2723

\section*{Menu 'Text List'}
6.4.1.21.3.21.1 Command 'Add Language'. ..... 2728
6.4.1.21.3.21.2 Command 'Create Global Text List'. ..... 2728
6.4.1.21.3.21.3 Command 'Export Everything as Text' ..... 2728
6.4.1.21.3.21.4 Command 'Export All Unicode .txt Text List Files'. ..... 2729
6.4.1.21.3.21.5 Command 'Insert Text'. ..... 2729
6.4.1.21.3.21.6 Command 'Import/Export Text Lists'. ..... 2729
6.4.1.21.3.21.7 Command 'Remove Language'. ..... 2730
6.4.1.21.3.21.8 Command 'Rename Language'. ..... 2730
6.4.1.21.3.21.9 Command 'Remove Unused Text List Entries'. ..... 2731
6.4.1.21.3.21.10 Command 'Check Visualization Text IDs' ..... 2731
6.4.1.21.3.21.11 Command 'Update Visualization Text IDs'. ..... 2731
6.4.1.21.3.21.12 Command 'Add Text List Support'. ..... 2732
6.4.1.21.3.21.13 Command 'Remove Text List Support'. ..... 2732

\section*{Command 'Add Language'}

Symbol: 재
Function: This command adds a further language column to the text list.
Call: Main menu "Textlist", context menu
Requirement: A text list or a global text list is open and active.
In the dialog box "Enter Language", enter a code for the new language, for example "en-US". CODESYS inserts the code as column header.

\section*{Command 'Create Global Text List'}

Symbol:
Function: This command creates the global text list in the "POUs" view.
Call: "Visualization", context menu.
Requirements: A visualization is open.
See also
- \({ }^{\star}\) Chapter 6.4.1.21.2.11 "Object 'GlobalTextList'" on page 2465
- \(\quad\) Chapter 6.4.1.9.10.7 "Managing static text in global text lists" on page 1894

\section*{Command 'Export Everything as Text'}

Symbol:
Function: This command exports all the text lists of the project.
Call: Main menu "Textlist", context menu

\section*{Requirement}
- A text list or a global text list is open and active.
- The visualization does not code the characters of the texts in Unicode.

CODESYS creates a file as plain text in the format . txt for each text list. The name of the text list becomes the name of the file. The directory into which the files are exported is set in "Project \(\rightarrow\) Project Settings \(\rightarrow\) Visualization", category "General" in "Text list files".
A controller can read and use this format. For example, you can copy the file to a controller and, by means of a configuration in the visualization manager, prevent the text lists from being transmitted again when loading the application.

\section*{Command 'Export All Unicode .txt Text List Files'}

\section*{Symbol: 画}

Function: This command exports all the text lists of the project.
Call: Main menu "Textlist", context menu

\section*{Requirement}
- A text list or a global text list is open and active.
- The visualization codes the characters of the texts in Unicode.
- The option "Use Unicode strings" in the visualization manager is activated.
- The compiler instruction VISU_USEWSTRING in the application is set. Check this by selecting the command "Properties" in the context menu of the application. Then select the "Compile" tab. VISU_USEWSTRING must be entered in the input field for "Compiler defines".

CODESYS creates a file as plain text in the format . txt for each text list. The name of the text list becomes the name of the file. The directory into which the files are exported is set in "Project \(\rightarrow\) Project Settings \(\rightarrow\) Visualization", category "General" in "Text list files".

A controller can read and use this format. For example, you can copy the file to a controller and, by means of a configuration in the visualization manager, prevent the text lists from being transmitted again when loading the application.

See also
- \({ }^{4}\) Chapter 6.4.1.9.10 "Managing text in text lists" on page 1891

\section*{Command 'Insert Text'}

Symbol:
Function: This command inserts a new line into the text list above the selected line. An input field under "Standard" opens, in which you input the source text.

Call: Main menu "Textlist \(\rightarrow\) Insert Text", context menu
Requirement: A text list, not a "GlobalTextList", is open and active. A field in the table is selected.

See also
- \({ }^{4}\) Chapter 6.4.1.9.10 "Managing text in text lists" on page 1891

\section*{Command 'Import/Export Text Lists'}

Symbol: 函
Function: This command exports an active text list, imports a file, or matches a text list with a file. The file has the CSV format. The "Import/Export" dialog provides options for this.

Call: Menu bar: "Text List \(\rightarrow\) Import/Export Text Lists"; context menu
Requirement: A text list or global text list is active.

\section*{Dialog 'Import/ \\ Export'}
\begin{tabular}{|l|l|}
\hline "Select File for Import" & \begin{tabular}{l} 
File that CODESYS reads. \\
\\
\\
\hline "... opens the dialog "Select Text List File" for you to select a file.
\end{tabular} \\
\hline & \begin{tabular}{l} 
File that CODESYS writes to. \\
\\
\hline
\end{tabular}\(\quad ..\)\begin{tabular}{l} 
opens the dialog "Select Text List File" for you to select a file and directory.
\end{tabular} \\
\hline
\end{tabular}

Table 525: "Import/Export Type"
\begin{tabular}{|l|l|}
\hline "Import" & \begin{tabular}{l} 
Requirement: A file is selected in "Select file for compare or import". \\
The file can contain text list entries for both the global text list and text lists. \\
Global text list
\end{tabular} \\
& \begin{tabular}{l} 
- \begin{tabular}{l} 
CODESYS reads the file, compares the text list entries for the same source \\
text, and accepts differences in the translations. CODESYS overwrites any \\
translations in the project. \\
Text lists \\
- \\
CODESYS reads the file, compares the text list entries for the same ID, \\
and accepts differences in the source text and translations into the project. \\
CODESYS overwrites any text list entries in the project.
\end{tabular} \\
- If the file contains a new ID, then the text list entry is imported into the text \\
list of the project and the text list is added.
\end{tabular} \\
\hline "Import replacement file" & \begin{tabular}{l} 
Requirement: A replacement file is selected in "Select file for compare or \\
import". \\
The replacement file contains replacements for the global text list. \\
CODESYS processes the replacement file row by row and performs the speci- \\
fied replacements in the global text list.
\end{tabular} \\
\hline "Export" & \begin{tabular}{l} 
The structure of the replacement file is described in the section "Managing static \\
text in a global text list".
\end{tabular} \\
\hline "Export text differences only" & \begin{tabular}{l} 
Requirement: The file that CODESYS writes to is selected in "Select export file". \\
CODESYS exports all texts from all text lists of the current project. All lan- \\
guages available in the project are inserted as columns in the export file. The \\
file can be used for the external translation of the language-dependent texts.
\end{tabular} \\
\begin{tabular}{ll} 
Requirement: An import file is selected for the comparison in "Select file for \\
compare or export", and an export file that CODESYS writes to is selected in \\
"Select export file". \\
CODESYS reads the import file and then uses that information to compare the \\
rows of the active text list. CODESYS ignores the rows that match. If rows differ, \\
then CODESYS writes the row to the export file and, if necessary, copies trans- \\
lations from the text list. CODESYS accepts the translations from the import file \\
and overwrites them if necessary.
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\boldsymbol{y y}^{4}\) Chapter 6.4.1.9.10 "Managing text in text lists" on page 1891
- \(\xi^{*}\) "Updating the global text list with a replacement file" on page 1895

\section*{Command 'Remove Language'}

Symbol: 느
Function: Removes the selected language column from the text list.
Call: Main menu "Textlist", context menu
Requirement: A text list or a global text list is open and active. A field is selected in the column of the language that you wish to remove.
See also
- \({ }^{4}\) Chapter 6.4.1.9.10 "Managing text in text lists" on page 1891

\section*{Command 'Rename Language'}

Symbol: 는

Function: Opens a dialog for specifying a new name for a language that is displayed in the text list as a column heading.
Call: Menu bar: "Text List"; context menu.
Requirement: A text list or global text list is active. A field in the language column to be renamed is selected.

See also
- \(\quad\) Chapter 6.4.1.9.10 "Managing text in text lists" on page 1891

\section*{Command 'Remove Unused Text List Entries'}

Symbol:
Function: This command checks whether a text list entry in the project is used as static text. If not, CODESYS removes it from the text list.

Call: Main menu "Textlist", context menu
Requirement: The "GlobalTextList" is open and active. A field in the table is selected.
See also
- \({ }^{4}\) Chapter 6.4.1.9.10 "Managing text in text lists" on page 1891

\section*{Command 'Check Visualization Text IDs'}

Symbol:
Function: This command checks whether the ID of a text list entry in the project is correct and reports the result.

Call: Main menu "Textlist", context menu
Requirement: The "GlobalTextList" is open and active. A field in the table is selected.
If CODESYS finds during checking that the global text list and the static texts of the visualizations do not correspond, this could be because the global text list is or was write protected. The requirement for this is that you have set up a user management system in the project.

See also
- \({ }^{4}\) Chapter 6.4.1.9.10 "Managing text in text lists" on page 1891

\section*{Command 'Update Visualization Text IDs'}

Symbol
Function: This command updates all inconsistent IDs in a static text list.
Call: Main menu "Textlist \(\rightarrow\) Paste Text", context menu
Requirement: The "GlobalTextList" is open and active. A field in the table is selected. The object is write protected.
If CODESYS finds during checking that the global text list and the static texts of the visualizations do not correspond, this could be because the global text list is or was write protected. The requirement for this is that you have set up a user management system in the project.

See also
- \(\quad\) Chapter 6.4.1.9.10 "Managing text in text lists" on page 1891
- \(\Leftrightarrow\) Chapter 6.4.1.6.6 "Protecting Objects in the Project by Access Rights" on page 1826

\section*{Command 'Add Text List Support'}

\section*{Symbol:}

Function: This command adds text list support to the selected DUT object (type: enumeration).
CaII: Context menu of a standard DUT object (type: enumeration \(\stackrel{\rightharpoonup}{*}_{*}\) ).
Text list support allows the localization of the enumeration component identifier and the display of the symbolic component value in a text output of a visualization.

See also
- \({ }^{*} \boldsymbol{y}\) Chapter 6.4.1.21.2.9 "Object 'DUT"' on page 2461
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.21.13 "Command 'Remove Text List Support"' on page 2732

\section*{Command 'Remove Text List Support'}

Symbol:
Function: This command removes text list support from the selected enumeration object.
Call: Context menu of an object of an enumeration with text list support (
Text list support allows the localization of the enumeration component identifier and the display of the symbolic component value in a text output of a visualization.
47293
See also
- \({ }^{*} \boldsymbol{y}\) Chapter 6.4.1.21.2.9 "Object 'DUT'" on page 2461
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.21.12 "Command 'Add Text List Support"' on page 2732

\section*{Menu 'Trace'}
6.4.1.21.3.22.1 Command 'Add Variable'. ..... 2732
6.4.1.21.3.22.2 Command 'AutoFit'. ..... 2733
6.4.1.21.3.22.3 Command 'Compress'. ..... 2733
6.4.1.21.3.22.4 Command 'Configuration' ..... 2733
6.4.1.21.3.22.5 Command 'Cursor' ..... 2733
6.4.1.21.3.22.6 Command 'Download Trace'. ..... 2734
6.4.1.21.3.22.7 Command 'Export Symbolic Trace Config'. ..... 2735
6.4.1.21.3.22.8 Command 'Load Trace'. ..... 2737
6.4.1.21.3.22.9 Command 'Mouse Zooming'. ..... 2737
6.4.1.21.3.22.10 Command 'Convert to Multi-Channel'. ..... 2737
6.4.1.21.3.22.11 Command 'Convert to Single-Channel' ..... 2738
6.4.1.21.3.22.12 Command 'Online List' ..... 2739
6.4.1.21.3.22.13 Command 'Reset Trigger'. ..... 2740
6.4.1.21.3.22.14 Command 'Reset View'. ..... 2740
6.4.1.21.3.22.15 Command 'Save Trace'. ..... 2741
6.4.1.21.3.22.16 Command 'Start Trace'. ..... 2741
6.4.1.21.3.22.17 Command 'Stop Trace'. ..... 2741
6.4.1.21.3.22.18 Command 'Stretch'. ..... 2742
6.4.1.21.3.22.19 Command 'Upload Trace'. ..... 2742
6.4.1.21.3.22.20 Command 'Statistics'. ..... 2742

\section*{Command 'Add Variable'}

Function: This command adds a trace variable to the configuration.

\section*{Call: Main menu "Trace".}

A new variable is displayed in the tree view of the trace configuration. The associated variables configuration appears to the right in "Variable Settings".

See also
- \(\Rightarrow\) Chapter 6.4.1.21.4.16.2 "Dialog 'Trace Configuration'" on page 2804
- \(\Leftrightarrow\) Chapter 6.4.1.13.3.3 "Creating trace configuration" on page 2010

\section*{Command 'AutoFit'}

Symbol: Ir
Function: This command scales the y-axis of the trace diagram for optimum display of all graphs, making sure that the \(y\)-values fit in the visible region of the diagrams. The command works with both single-channel and multi-channel displays.

Call: Menu bar: "Trace"; context menu.

Trace in a single-channel display
Trace in a multi-
Trace in a multi-
channel display
When all trace variables are displayed in one diagram, the trace is in single-channel display.

When the trace variables are displayed in multiple diagrams, the trace is in multi-channel display.

See also
- \(\Rightarrow\) Chapter 6.4.1.21.4.16.2 "Dialog 'Trace Configuration'" on page 2804
- \({ }^{\text {• }}\) Chapter 6.4.1.13.3.6 "Navigating into trace data" on page 2015

\section*{Command 'Compress'}

Symbol: 花
Function: This command compresses the trace graph by zooming into the displayed time range by a fixed percentage.
Call: Main menu "Trace", or context menu.
See also
- \({ }^{4}\) Chapter 6.4.1.21.3.22.18 "Command 'Stretch'" on page 2742
- \& Chapter 6.4.1.13.3.6 "Navigating into trace data" on page 2015

\section*{Command 'Configuration'}

Function: This command opens the "Trace Configuration" dialog box for enabling the configuration of the data recording.
Call: Main menu "Trace", or context menu.
See also
- \({ }^{\mu}\) Chapter 6.4.1.21.4.16.2 "Dialog 'Trace Configuration"' on page 2804
- 步 "Subdialog 'Variable Settings'" on page 2807

\section*{Command 'Cursor'}

Symbol:

\section*{Function：This function}
－inserts a trace cursor into the trace diagram when no trace cursor is available
－inserts a second trace cursor into the trace diagram when 1 trace cursor is available
－removes the trace cursors when 2 trace cursors are available
Call：Menu bar：＂Trace＂；context menu．
A trace cursor is a small black triangle with a vertical black line running parallel to the \(y\)－axis．
\begin{tabular}{ll}
\begin{tabular}{l} 
Trace diagram \\
without trace
\end{tabular} & \begin{tabular}{l} 
In this mode，you can process the trace diagram with the mouse pointer．The \(x\)－value that \\
focuses on with the cursor is displayed in the status bar with normal style．Example：＂Time： \\
cursors
\end{tabular}\(\quad 1 \mathrm{~m} 23 \mathrm{~s} 456 \mathrm{~ms}\) ；Value： 1 ＂
\end{tabular}

Trace diagram In the status bar and \(y\)－value，CODESYS prints the time that was marked by the trace cursor． with one trace cursor

Example：＂Time： 1 m 23 s 456 ms ＂

Trace diagram In the status bar，CODESYS prints the two times and the time interval that are marked by the with two trace cursors
two trace cursors．Example：＂Time： 1 m 23 s 456 ms －Time： \(1 \mathrm{~m} 24 \mathrm{~s} 456 \mathrm{~ms}(\mathrm{\Delta} 1 \mathrm{~s})\)＂．

User input in the If one or two trace cursors are available，then you can move them along the \(x\)－axis． trace diagram
\begin{tabular}{|l|c|l|}
\hline Mouse Input & Symbol & Effect \\
\hline \begin{tabular}{l} 
Drag the triangle of a trace \\
cursor to another position．
\end{tabular} & \begin{tabular}{l} 
While the mouse button is pressed，the cursor can be moved without \\
restriction．The current \(y\)－value is always displayed in the status bar． \\
When the mouse button is released，the cursor jumps to the nearest \\
measuring point
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline Keyboard Shortcuts & Effect \\
\hline \begin{tabular}{l}
［Left arrow］ \\
［Right arrow］\(]\)
\end{tabular} & CODESYS moves the black trace cursor to the next measuring point． \\
\hline \begin{tabular}{l}
{\([\) Shift \(]+[\) Left Arrow］} \\
［Shift \(]+[\) Right Arrow \(]\)
\end{tabular} & CODESYS moves the gray trace cursor to the next measuring point． \\
\hline
\end{tabular}

See also
－\＃Chapter 6．4．1．13．3．6＂Navigating into trace data＂on page 2015

\section*{Command＇Download Trace＇}

\section*{Symbol：⿷匚⿱口⿰口口木}

Function：This command transfers the trace configuration on the controller to the associated application，and starts the data recording．The recorded data is transferred back to the develop－ ment system．The trace diagram shows the current samples and continues．
Call：Menu bar：＂Trace＂；context menu．
Requirement：The command is available when the assigned application is in online mode．
See also
－\＆Chapter 6．4．1．13．3．4＂Operating the data recording＂on page 2014

\section*{Command 'Export Symbolic Trace Config'}

Function: This command exports a trace configuration to a traceconfig file.
Call: Main menu "Trace", or context menu.
Requirement: The origin application includes a symbol configuration that defines the configured trace variables as symbols. Access to the IEC variables where data was recorded is therefore symbolic. Then you can use the trace configuration for various similar applications.

Using the configuration file

You can transfer this file to any runtime system. At runtime, its CmpTraceMgr runtime system component can access and perform data recording. The configuration file also includes information about the application context in addition to the configuration data.

The configuration file defines the following context:
- Application name
- Trace name
- Task name

The application that is executed at runtime must fulfill the following conditions:
- The application has the same name as the origin application.
- The trace that is configured in the application has the same as the trace that is configured in the origin application
- The task that is running in the data recording has the same name as the task that is configured in the origin application.

\section*{NOTICE!}

The configuration is not loaded automatically. You must execute the command explicitly.
You can proceed as follows:
- Access the trace manager programmatically via IEC code by using library interfaces.
- Register the configuration file with the trace manager. Then the trace manager loads the configuration file when the application is started.

For more information about the functionality of the trace manager, refer to "Trace Manager Runtime System Component Description".

\section*{Sample configu- \\ ration file}

\section*{Configuration \\ file \\ Trace_Trigge \\ r.traceconfi \\ g}
```

[key]; [value]
Version; 0x03050000
Name; Application.Trace_Trigger
ApplicationName; Applicātion
ApplicationDataGuid; 00000000-0000-0000-0000-0000000000000
IecTaskName; MainTask
Comment;
Trigger.Flags; 5
Trigger.Edge; 2
Trigger.Position; 0
Trigger.UpdatesAfterTrigger; 50
Trigger.Variable.Name; PLC_PRG.B.OUT
Trigger.Variable.AddrFlags; 0x00000101
Trigger.Variable.Class; 0
Trigger.Variable.Size; 1
Trigger.Level;
Condition.Name;
Condition.AddrFlags; 0x00000000
Condition.Class; 0
Condition.Size; 0
EveryNCycles; 1
BufferEntries; 100
Flags; 16
O.Variable; PLC_PRG.S5.OUT
0.Address.AddrFlags; 0x00000101
0.Class; 7
O.Size; 2
O.GraphColor; 4278190335
O.GraphType; 3
O.MinWarningColor; 4278190080
0.MaxWarningColor; 4294901760
0.CriticalLowerLimit; 0
O.CriticalUpperLimit; 0
0.ActivateMinWarning; 0
0.ActivateMaxWarning; 0
O.YAxis; 0
0.Data;
1.Variable; PLC PRG.B.OUT
1.Address.AddrFlags; 0x00000101
1.Class; 0
1.Size; 1
1.GraphColor; 4278222848
1.GraphType; 1
1.MinWarningColor; 4278190080
1.MaxWarningColor; 4294901760
1.CriticalLowerLimit; 0
1.CriticalUpperLimit; 0
1.ActivateMinWarning; 0
1.ActivateMaxWarning; 0
1.YAxis; 0
1.Data;

```

See also
- \(\xi^{*}\) Chapter 6.4.1.13.3.7 "Managing trace" on page 2016

\section*{Command 'Load Trace'}

Function: This command makes it possible to load a file, which contains the configuration and data, and was saved to the file system of the development system. The "Load Trace" dialog box opens.
Call: Main menu "Trace", or context menu.

Dialog box
"Load Trace"
\begin{tabular}{|l|l|}
\hline "File name" & Name of the file that is loaded \\
\hline "File type" & \begin{tabular}{l} 
File format \\
\\
\\
\\
\\
• *.trace: "Trace file" that includes the trace configuration \\
\\
\hline
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{4}\) Chapter 6.4.1.21.3.22.15 "Command 'Save Trace'" on page 2741
- \& Chapter 6.4.1.13.3.7 "Managing trace" on page 2016

\section*{Command 'Mouse Zooming'}

Symbol: \(-\frac{1}{\infty}\) (command disabled), 国 (command enabled)
Function: This command enables and disables mouse zooming in the trace diagram.
Call: Menu bar: "Trace"; context menu.

User input in the If the command is enabled, then you can stretch a box with the mouse. When you release the trace diagram mouse button, the display zooms in on the box and the data is enlarged.

See also
- ② Chapter 6.4.1.13.3.6 "Navigating into trace data" on page 2015

\section*{Command 'Convert to Multi-Channel'}

Function: This command switches the display in the trace editor from single-channel to multichannel.

Call: Menu bar: "Trace"; context menu.

Multi-channel Multi-channel display means that the trace variables are displayed in multiple diagrams. display


See also
- \(\boldsymbol{y}^{*}\) Chapter 6.4.1.21.4.16.2 "Dialog 'Trace Configuration'" on page 2804
- \(\Rightarrow\) Chapter 6.4.1.13.3.6 "Navigating into trace data" on page 2015

\section*{Command 'Convert to Single-Channel'}

Function: This command switches the display in the trace editor from multi-channel to singlechannel.

Call: Menu bar: "Trace"; context menu.

Single-channel display

If a trace is displayed as single-channel, then all trace variables are included in one diagram.


See also
- \(\quad\) Chapter 6.4.1.21.4.16.2 "Dialog 'Trace Configuration'" on page 2804
- \({ }^{\text {• }}\) Chapter 6.4.1.13.3.6 "Navigating into trace data" on page 2015

\section*{Command 'Online List'}

Function: This command opens the "Online List" dialog. If the trace editor of a DeviceTrace object is active, then all traces that are running on the controller are displayed in a tree view. If the trace editor of an application-specific trace object is active, then only this trace is displayed if it is running on the controller.
Call: Menu bar: "Trace"; context menu of the trace editor.
Requirement: The runtime system uses the CmpTraceMgr components. An application belonging to the device is in online mode.

\section*{NOTICE!}

\section*{Closing the DeviceTrace editor terminates the connection to the controller.}

Please note that the connections to the controller is also terminated when the last open "DeviceTrace" editor is closed. In order for device traces to be displayed again in the project, you have to reload them into the "DeviceTrace" objects.
At this time, closing the editor is also the recommended procedure for deliberately terminating the connection to the controller. Logging out is not enough for this.

\section*{Dialog 'Online}

List'
\begin{tabular}{|l|l|}
\hline "Delete from runtime" & Stops and removes the selected trace from the running application. \\
\hline "Upload" & \begin{tabular}{l} 
This command is visible when a Device Trace is loaded in the trace editor. A \\
DeviceTrace is a trace that runs on the controller: In the device tree, it can be \\
represented with a DeviceTrace object directly below a device. \\
When you execute this command, the trace that is selected in the tree view is \\
loaded from the runtime system into the trace editor. Any existing configuration in \\
the project is overwritten. For example, the device can provide traces for data of \\
the processor load (cpuload, plcload), which then you can track in the trace \\
editor in CODESYS. \\
An individual "DeviceTrace" object is necessary in the device tree for each trace \\
of the device that should be displayed in the project.
\end{tabular} \\
\hline
\end{tabular}

See also
- \# Chapter 6.4.1.13.3.5 "Accessing All Traces on the Controller" on page 2014
- \& Chapter 6.4.1.21.2.7 "Object 'DeviceTrace"' on page 2426
- \({ }^{*} \boldsymbol{*}\) Chapter 6.4.1.21.2.31 "Object 'Trace"' on page 2541
- \# Chapter 6.4.1.21.3.22.19 " Command 'Upload Trace"' on page 2742

\section*{Command 'Reset Trigger'}

Symbol: 热
Function: This command resets the trace configuration after a triggered data recording. Then the application can record new data and react to a trigger again.

Call: Main menu "Trace", or context menu.
Requirement: After triggering, the complete data is in the buffer of the development system.
See also
- \({ }^{4}\) Chapter 6.4.1.13.3.4 "Operating the data recording" on page 2014

\section*{Command 'Reset View'}

Symbol: \(\mathfrak{Q}^{2}\)
Function: This command resets the trace diagram to the default view.
Call: Main menu "Trace", or context menu.

Requirement: The display in the trace diagram has been changed by zooming, scrolling, or "AutoFit".

See also
- \(\sum^{*}\) Chapter 6.4.1.21.3.22.2 "Command 'AutoFit'" on page 2733
- \({ }^{4}\) Chapter 6.4.1.13.3.6 "Navigating into trace data" on page 2015

\section*{Command 'Save Trace'}

Function: This command saves the data to a file on the development system. Depending on the file format, the configuration may also be saved. The "Save Trace" dialog box opens.

Call: Main menu "Trace", or right-click.

Dialog Box
'Save Trace'
\begin{tabular}{|c|c|}
\hline "File name" & Name and location of the trace file \\
\hline "File type" & \begin{tabular}{l}
File format \\
- *.trace: \\
"Trace file" contains the data and configuration. \\
You can run the "Load Trace" command to load the file to the trace editor when offline. \\
- *.txt: \\
"Text file" contains the recorded data. You can load this file type and edit it with tools that support CSV format. \\
It cannot be loaded to the trace editor when offline because the trace editor cannot read this format. \\
- *.trace.csv \\
"Trace CSV file (data only)" contains the recorded data. Address information is provided for each trace variable. The created file can be read in the runtime system. The data is imported but the trace cannot be started because the variable addresses are not saved.
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{\star}>\) Chapter 6.4.1.21.3.22.8 "Command 'Load Trace"' on page 2737
- \(\xi^{\circ}\) Chapter 6.4.1.13.3.7 "Managing trace" on page 2016

\section*{Command 'Start Trace'}

Symbol:
Function: This command starts the data recording on the controller when it is stopped.
Call: Main menu "Trace", or context menu.
Requirement: The assigned application on the runtime system is running and a trace configuration is loaded.

See also
- \({ }^{\leftrightarrows}\) Chapter 6.4.1.13.3.4 "Operating the data recording" on page 2014

\section*{Command 'Stop Trace'}

Symbol:

Function: This command stops the data recording of a trace.
Call: Main menu "Trace", or context menu.
Requirement: The assigned application on the runtime system is running and executes a trace.
See also
- \(\Leftrightarrow\) Chapter 6.4.1.13.3.4 "Operating the data recording" on page 2014

\section*{Command 'Stretch'}

Symbol: \(\overleftrightarrow{m}\)
Function: This command stretches the trace graph by zooming out of the displayed time range by a fixed percentage.
Call: Main menu "Trace", or context menu.
See also
- \({ }^{\circ}\) Chapter 6.4.1.21.3.22.3 "Command 'Compress'" on page 2733
- \({ }^{4}\), Chapter 6.4.1.13.3.6 "Navigating into trace data" on page 2015

\section*{Command 'Upload Trace'}

Function: This command establishes the connection to the PLC device, if not already connected. Then it opens the "Online List" dialog listing the traces running on the controller. Then the selected trace is loaded to the trace editor by means of the "Upload" command in the dialog.
Call: Menu bar: "Trace"; context menu of the trace editor.
Requirement: The editor of a "DeviceTrace" object is open. The runtime system uses the CmpTraceMgr components (trace manager). At least one application in the runtime system in running. The communication settings for the PLC are configured correctly in the CODESYS project.

\section*{NOTICE!}

\section*{Closing the DeviceTrace editor terminates the connection to the controller.}

Please note that the connections to the controller is also terminated when the last open "DeviceTrace" editor is closed. In order for device traces to be displayed again in the project, you have to reload them into the "DeviceTrace" objects.
At this time, closing the editor is also the recommended procedure for deliberately terminating the connection to the controller. Logging out is not enough for this.

\section*{See also}
- *"Dialog 'Online List"' on page 2740
- \(y^{4}\) Chapter 6.4.1.21.2.7 "Object 'DeviceTrace"' on page 2426
- \({ }^{4}\) Chapter 6.4.1.13.3.5 "Accessing All Traces on the Controller" on page 2014

\section*{Command 'Statistics'}

Function: This command opens the "Trace Statistics" dialog box, which shows statistics about each trace variable.
Call: Main menu "Trace", or right-click.
Requirement: The trace editor contains samples.

Dialog Box The analyzed time range and duration are shown in the caption.
'Trace Statistics'
The table contains one line per signal.
\begin{tabular}{|l|l|}
\hline "Signal" & Name pf the trace variable (for example, PLC_PRG. S1) \\
\hline "Index" & 0-based index of the signal order \\
\hline "N" & Number of measurements for the calculations \\
\hline "Min" & Smallest value \\
\hline "Avg" & Average \\
\hline "Median" & Middle value when the values are ordered by size \\
\hline "RMS" & Root mean square \\
\hline "StdDev" & Standard deviation \\
\hline "Max" & Largest value \\
\hline "Integral" & Integral \\
\hline "Min \(\Delta t[s] "\) & Smallest change of time intervals for successive values \\
\hline "Avg \(\Delta t[s] "\) & Average change of time intervals for successive values \\
\hline "Median \(\Delta t[s] "\) & Median change of time intervals for successive values \\
\hline "StdDev \(\Delta t[s] "\) & Standard deviation of change of time intervals for successive values \\
\hline "Max \(\Delta t[s] "\) & Largest change of time intervals for successive values \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
Click a column head in the \\
table.
\end{tabular} & \begin{tabular}{l} 
CODESYS sorts the table by that column, changing the order from ascending to \\
descending and back. \\
Default: The table is sorted ascending by the "Index" column. The signals are \\
then sorted in the same order as in the signal tree.
\end{tabular} \\
\hline Click in the line. & \begin{tabular}{l} 
The line is selected. You can select or clear other lines by pressing [Shift]+ \\
[arrow] up or down.
\end{tabular} \\
\hline\([C t r l]+[C]\) & \begin{tabular}{l} 
CODESYS copies the selected lines as text to the clipboard. The values of the \\
individual columns are tab-separated, and the lines are delimited with the control \\
character [CR] or [LF]. \\
Requirement: At least one line is selected.
\end{tabular} \\
\hline
\end{tabular}

\section*{See also}
- Hy Chapter 6.4.1.13.3 "Data Recording with Trace" on page 2007

\section*{Other}
6.4.1.21.3.23.1 Command 'Add Watch'......................................................... 2743
6.4.1.21.3.23.2 Command 'Implement Interfaces'......................................... 2744
6.4.1.21.3.23.3 Command 'Limit Results to Current Declaration'................... 2744

\section*{Command 'Add Watch'}

\section*{Symbol: 6d}

Function: This command adds the variable of the current location of the cursor to a watchlist for the purpose of online monitoring.

Call: Right-click a variable in an editor when the application is in online mode.
This command adds the variable to the currently opened watchlist. If a watchlist is not open, then the variable is added to the "Watch 1 " list and that view opens.

See also
- \(\boldsymbol{y y}^{*}\) Chapter 6.4.1.13.2.3 "Using watch lists" on page 2002
- \(\Leftrightarrow\) Chapter 6.4.1.13.2 "Monitoring of Values" on page 1995

\section*{Command 'Implement Interfaces'}

Function: This command updates the implemented interfaces for a function block.
Call: Context menu of the selected function block (FB) in the device tree.
Requirement: The function block implements an interface that you have modified. For example, an additional method was added to the interface.

> In object-oriented programming, if you derive a function block (FB) from a base function block, which implements one or more interfaces, for the purpose of inheritance, then the following applies:

> When you execute the "Implement Interfaces" command for the derived FB, all interface methods and interface attributes of the base FB are accepted into the derived FB in the form of stubs (without implementation). Then you are responsible for making sure that an "empty" method/attribute in the derived FB does not conflict with an implemented one in the base FB. The following actions are taken to support you in this case: If there es a base implementation for a method/attribute, then CODESYS adds a pragma attribute \{error..\} in the first line of the affected derived interface method or interface attribute that will generate the error message. If there is no base implementation for the method/ attribute, then there is a pragma attribute entry for a warning. After editing the block, you must remove the error pragma attribute entry explicitly.

\section*{See also}
- \({ }^{*} \boldsymbol{y}\) Chapter 6.4.1.21.2.21.5 "Object 'Interface"' on page 2484
- \({ }^{4}\) Chapter 6.4.1.9.22.3 "Implementing interfaces" on page 1931

\section*{Command 'Limit Results to Current Declaration'}

Function: When multiple declarations have been found, this command collapses the display in the cross-reference list. It shows only the results for the declaration that you selected explicitly in the list.
Call: Right-click.
Requirement: The cross-reference list is active. Multiple declarations for the searched symbol are listed as cross-references.

See also
- \({ }^{*}>\) Chapter 6.4.1.9.14.1 "Using the cross-reference list to find occurrences" on page 1906
6.4.1.21.4 Dialogs
6.4.1.21.4.1 General ..... 2745
6.4.1.21.4.2 Dialog 'Import Assistant' ..... 2745
6.4.1.21.4.3 Dialog 'Library Reference Conversion' ..... 2746
6.4.1.21.4.4 Dialog 'Select Function Block'. ..... 2746
6.4.1.21.4.5 Dialog 'Device Conversion' ..... 2747
6.4.1.21.4.6 Dialog 'Breakpoint Properties'. ..... 2747
6.4.1.21.4.7 Dialog 'Permissions' ..... 2748
6.4.1.21.4.8 Dialog Box 'Prepare Value'. ..... 2749
6.4.1.21.4.9 Dialog 'New Breakpoint'. ..... 2750
6.4.1.21.4.10 Dialog 'Monitoring Range' ..... 2752
6.4.1.21.4.11 Dialog 'Properties'. ..... 2753
6.4.1.21.4.12 Dialog 'Project Settings'. ..... 2766
6.4.1.21.4.13 Dialog 'Project Environment'. ..... 2778
6.4.1.21.4.14 Dialog 'Options'. ..... 2781
6.4.1.21.4.15 Dialog 'Customize' ..... 2800
6.4.1.21.4.16 Dialog 'Trace Configuration' ..... 2803
6.4.1.21.4.17 Dialog Box 'Trend storage'. ..... 2809
6.4.1.21.4.18 Dialog Box 'Advanced Trend Settings' ..... 2810
6.4.1.21.4.19 Dialog 'Certificate Selection'. ..... 2811

\section*{General}

The dialogs of the CODESYS user interface bascally are described on the help pages for the CODESYS menu commands or CODESYS objects. The help book "Dialogs" contains only descriptions of dialogs, which
- appear only after multi-step calls after a certain menu command call or within an object editor,
- or which are not placed on a help page for a command or for an object because of their complexity (multiple subdialogs).

\section*{Dialog 'Import Assistant'}

Function: The dialog allows for the transfer of CODESYS options and package installations from an older CODESYS installation that was found in the local computer.
Call: The dialog opens when a recently installed CODESYS version is started for the first time and an older version is installed on the computer.
\begin{tabular}{|l|l|}
\hline "Program settings" & \begin{tabular}{l} 
■: The user-specific CODESYS options are transferred from the older installa- \\
tion to the new installation.
\end{tabular} \\
\hline "Packages" & \begin{tabular}{l} 
目: The packages installed with the older CODESYS version are transferred to \\
the Package Manager of the new version. See the list of discovered package \\
installations with the "Name", "Version", and "Installation date".
\end{tabular} \\
\hline "Import" & \begin{tabular}{l} 
The program settings and/or options are transferred to the current CODESYS \\
version.
\end{tabular} \\
\hline "Skip" & \begin{tabular}{l} 
The program settings and/or options are not transferred to the current \\
CODESYS version.
\end{tabular} \\
\hline
\end{tabular}

See also
- M Chapter 6.4.1.2.1.1 "Setting CODESYS options" on page 1802

\section*{Dialog 'Library Reference Conversion'}

Function: The dialog defines how references to libraries that are no longer available are to be handled. Note: The undefined library references can be found in the Global Library Manager located in the "POUs" view.

Call: When opening a CoDeSys V2.3 project in V3, the dialog opens when the converter detects a library which cannot be used anymore in the current CODESYS version.

A CoDeSys V2.3 project can be converted into a CODESYS V3 project only if the CODESYS V2.3 Converter package is installed in CODESYS V3. The package is available in the CODESYS Store.

Table 526: "What do you want to do?"
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
"Convert and install the library \\
as well."
\end{tabular} & \begin{tabular}{l} 
The converter also converts the library file into the new format. It remains refer- \\
enced in the project. It is installed automatically in the library repository in the \\
"Other" category. If the library does not provide the necessary project information \\
for an installation, then the "Enter Project Information" dialog opens for the infor- \\
mation to be added.
\end{tabular} \\
\hline \begin{tabular}{l} 
"Use the following library that \\
has already been installed"
\end{tabular} & \begin{tabular}{l} 
The previously used library is replaced by another library. The "Browse" button \\
opens a dialog for selecting from the local library repository.
\end{tabular} \\
\hline \begin{tabular}{l} 
"Ignore the library. The \\
reference will not appear in the \\
converted project"
\end{tabular} & The library reference is removed from the project. \\
\hline
\end{tabular}
"Remember this mapping for all future occurrences of that library reference"

The settings made here in the dialog are also used for future project conversions.

See also
- \(\sum^{*}\) Chapter 6.4.1.3.3 "Opening a V2.3 project" on page 1809
- \(\sum^{*}\) Chapter 6.4.1.21.3.9.5 "Command 'Library Repository"' on page 2657

\section*{Dialog 'Select Function Block'}

Function: The dialog is used for selecting a function block for I/O mapping. The function block should be mapped to the I/O channel selected in the "<device name> I/O Mapping" tab or to the object selected in the "<device name> IEC Objects" tab.

\section*{Call:}
- Tab "<device name> I/O Mapping", command button ↔"Add FB for I/O channel"
- Tab "<device name> IEC Objects", command buttonґ "Add"

The dialog provides all function blocks from the active application and the libraries included in the project which fulfill the following:
- The function block has the \{attribute 'io_function_block'\} attribute.
- The function block contains input or output parameters that match the channel type (input, output, data type) and has the \{attribute 'io_function_block_mapping'\} attribute.

When a function block is selected that provides multiple matching parameters, only the first one is mapped automatically to the channel. The others can only be assigned manually in the "<device name> I/O Mapping" tab.

After the function block is assigned, the parameter of the function block instance is entered in the "Variable" column of the mapping table. Then the path is composed as follows:
<application name>.<device channel name>_<FB name>_<continuous FB
instance number>. <FB parameter name>
Example: App1.Out_4_Int_myScale_Output_Int_1.iOutput for the parameter iOutput of the first insēred \(\overline{\text { instance of }} \overline{\text { the }}\) function block myScale_Output.
\begin{tabular}{|l|l|}
\hline "Find" & Input field for searching for function block names \\
\hline "Type" & \begin{tabular}{l} 
Function blocks in the tree structure that match the channel type. Nodes: appli- \\
cation, library name(s)
\end{tabular} \\
\hline "Documentation" & \begin{tabular}{l} 
Shows the available documentation for the library selected in the tree or the \\
library block.
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{3}>\) Chapter 6.4.1.21.2.8.12 "Tab '<device name> I/O Mapping'" on page 2444
- \({ }^{\star}\) Chapter 6.4.1.20.6.3.23 "Attribute 'io_function_block', 'io_function_block_mapping'" on page 2293

\section*{Dialog 'Device Conversion'}

Function: The dialog defines how references to devices that are no longer available are to be handled.

Call: When opening a CoDeSys V2.3 project in V3, the dialog opens when the converter detects a device reference which cannot be used anymore.

A CoDeSys V2.3 project can be converted into a CODESYS V3 project only if the CODESYS V2.3 Converter package is installed in CODESYS V3. The package is available in the CODESYS Store.

Table 527: "What do you want to do?"
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
"Use the following device that \\
has already been installed"
\end{tabular} & \begin{tabular}{l} 
CODESYS replaces the previously used device in the device tree with another \\
device. The "Browse" button opens a dialog for selecting from the local device \\
repository.
\end{tabular} \\
\hline \begin{tabular}{l} 
"Ignore the device. All device \\
specific objects will not be \\
available in the new project"
\end{tabular} & \begin{tabular}{l} 
The device entry with all objects inserted below it is removed from the device \\
tree.
\end{tabular} \\
\hline
\end{tabular}
"Remember this mapping for all future occurrences of that device"

The settings made here in the dialog are saved in the CODESYS Options, in the "CODESYS V2.3 Converter" category. As a result, they are also valid for future project conversions.

See also
- \(\quad\) n Chapter 6.4.1.3.3 "Opening a V2.3 project" on page 1809
- \({ }^{4}\) ) Chapter 6.4.1.21.3.9.8 "Command 'Device Repository"' on page 2663

\section*{Dialog 'Breakpoint Properties'}

Function: The dialog is used to display or change the properties of the selected breakpoint in the "Breakpoints" view.

\section*{Call:}
- "Breakpoints" view, 匪"Properties" button- "Breakpoints" view, 汤"New" button, "New Breakpoint" command or "New Data Breakpoint" command
Requirement: An entry is selected in the list of breakpoints.
The dialog is identical to the "New Breakpoint" dialog which is opened in the "Debug" menu by means of the respective commands. Therefore, see the description in the help for the "New Breakpoint" dialog.

See also
- \({ }^{4}\) Chapter 6.4.1.21.4.9 "Dialog 'New Breakpoint"' on page 2750

\section*{Dialog 'Permissions'}

Function: The permissions of user groups are defined here with which they can execute specific actions on specific objects in the project.
Call: Menu bar: "Project \(\rightarrow\) User Management".
Every change made in the dialog is applied immediately.

Actions All possible actions on objects of the projects are listed in "Actions". The actions are divided into four categories and assignments to all current objects of the project are listed below each action. For each "action->object" assignment, you can define the permission for each existing user group.
Action categories:
\begin{tabular}{|l|l|}
\hline\(\square\) "Commands" & Actions regarding the execution of commands \\
\hline \begin{tabular}{l} 
"Users, groups and \\
permissions"
\end{tabular} & \begin{tabular}{l} 
Actions regarding the configuration of user accounts, user groups, and their \\
permissions
\end{tabular} \\
\hline\(\square\) "Object types" & Actions regarding the creation of object types \\
\hline\(\square\) "Project objects" & \begin{tabular}{l} 
Actions regarding the viewing, modification, removal, and child-object handling of \\
objects of the project
\end{tabular} \\
\hline
\end{tabular}

Actions in detail:
\begin{tabular}{|l|l|}
\hline F "Execute" & Execute a menu command \\
\hline 8 "Create" & Create a new object in the project \\
\hline 8 "Add or remove children" & Add or remove a child object below an existing object \\
\hline 8"Modify" & \begin{tabular}{l} 
Modify an object in the editor or modification of user, group, and permission \\
settings in the corresponding editor/dialog
\end{tabular} \\
\hline F "Remove" & Delete or remove an object \\
\hline F"View" & Open the view of an object in the editor \\
\hline
\end{tabular}
\begin{tabular}{|l|l}
\hline\(\Rightarrow\) & Possible target of an action. This can be specific objects of the project, or the
\end{tabular} user, group, and permission configuration.

Permissions All defined user groups (except the "Owner" group) are listed in "Permissions" with a toolbar for configuring the permissions of a group.
\begin{tabular}{|l|l|}
\hline F Granted & \begin{tabular}{l} 
The action, which is selected in the actions view, on the selected target(s) is granted for \\
the selected group.
\end{tabular} \\
\hline - Denied & \begin{tabular}{l} 
The action, which is selected in the "Actions" view, on the selected target(s) is denied for \\
the selected group.
\end{tabular} \\
\hline H & \begin{tabular}{l} 
The permission that executes the actions, which are selected in the "Actions" view, on \\
the selected targets has not been defined explicitly. However, the actions are granted by \\
default; for example, because the corresponding permissions have been granted to the \\
parent object. Example: The group has the permission for the object "myplc". As a result, \\
it also has the permission by default for the object "myplc.pb_-1".
\end{tabular} \\
\hline No symbol & \begin{tabular}{l} 
The action, which is expanded in the actions view, on the selected targets has not been \\
denied explicitly. However, it is denied by default; for example, because it has been \\
denied to the parent object.
\end{tabular} \\
\hline & \begin{tabular}{l} 
There are currently multiple actions selected in the Actions view for which the group \\
does not have the same permission.
\end{tabular} \\
\hline
\end{tabular}

Toolbar:
\begin{tabular}{|c|c|}
\hline ヶ"Grant" & The selected action on the selected target object is granted explicitly for the selected group. \\
\hline - "Deny" & The selected action on the selected target object is denied explicitly for the selected group. \\
\hline * "Clear" & The permission for the selected action on the selected target object is reset to the default value for the selected group. \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Export/Import" & \begin{tabular}{l}
Opens a menu with the commands \\
- "Export all permissions" \\
- "Export selected permissions" \\
- "Import permissions"
\end{tabular} \\
\hline "Export all permissions" & \begin{tabular}{l}
Exports all actions and their configured access permissions of the current project to a user-specific file of data type *. perms. \\
To do this, the "Export Permissions" dialog opens for you to specify a file name and to select a location in the file directory. The default file type is Permissions (*.perms).
\end{tabular} \\
\hline "Export selected permissions" & \begin{tabular}{l}
Exports all selected actions and their configured access permissions of the current project to a user-specific file of data type *. perms. \\
To do this, the "Export Permissions" dialog opens for you to specify a file name and to select a location in the file directory. The default file type is Permissions (*.perms).
\end{tabular} \\
\hline "Import permissions" & \begin{tabular}{l}
The contents of \(a^{*}\). perms file is merged with the actions and permissions of the current project. Groups that are part of the file but not part of the project are ignored. The actions and permissions are aligned by name. \\
To do this, the "Import Permissions" opens for you to select the *. perms file from the file system.
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{\Perp}\) Chapter 6.4.1.6.6 "Protecting Objects in the Project by Access Rights" on page 1826

\section*{Dialog Box 'Prepare Value'}

Function: This dialog box is used for preparing a value for a forced variable. CODESYS executes the prepared action with the next forcing.

CODESYS opens the dialog box in the following situations:
- When clicking in the field "Prepared value" of a forced variable in the declaration section
- When clicking in the inline monitoring field of a forced variable
- When clicking in the field "Prepared value" of a forced variable in the monitoring window
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
"Prepare a new value for the \\
next write or force operation"
\end{tabular} & Value that CODESYS writes to the variable with the next force operation \\
\hline \begin{tabular}{l} 
"Remove a preparation with a \\
value"
\end{tabular} & CODESYS deletes the prepared value. \\
\hline \begin{tabular}{l} 
"Release the force, without \\
modifying the value"
\end{tabular} & \begin{tabular}{l} 
CODESYS retains the forced value and ends forcing. CODESYS marks the \\
variable <Unforce>.
\end{tabular} \\
\hline \begin{tabular}{l} 
"Release the force and restore \\
the variable to the value it had \\
before forcing it"
\end{tabular} & \begin{tabular}{l} 
CODESYS resets the forced value and ends forcing. The variable is marked with \\
<Unforce and restore>.
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.8.16 "Command 'Force Values'" on page 2649

\section*{Dialog 'New Breakpoint'}

Function: In the dialog, you define the settings for a new breakpoint or data breakpoint. It is identical to the "Breakpoint Properties" dialog which is used in the "Breakpoints" view.
Call:
- "Debug \(\rightarrow\) New Breakpoint"
- "Debug \(\rightarrow\) New Data Breakpoint"

Requirement: The application is in online mode.

Tab 'Condition' The dialog defines the requirements for which program processing should halt at a breakpoint.

\section*{NOTICE!}

Using conditional breakpoints slows down code execution, even when the condition does not yield TRUE.

Conditional breakpoints required a CODESYS runtime >= V3.5.4.0.

Table 528: "Tasks"
> "Break only when the breakpoint is hit in one of the following tasks"
\(\boxed{\square}\) : CODESYS evaluates the breakpoint only if it is reached by specific tasks. The required tasks must be activated.

For example, you can define a single debug task and also prevent other tasks that use the same block from being affected when debugging.

Table 529: "Hit Count"
\begin{tabular}{|l|l|}
\hline "Hit Count" & "Break always": The program always halts at this breakpoint. \\
& \begin{tabular}{l} 
Alternative: The program halts at the breakpoint when the breakpoint has been \\
hit as often as defined in the following (type in the required hit count or select it \\
from the number list):
\end{tabular} \\
& \begin{tabular}{l} 
- "Break when the hit count is equal to" \\
- "Break when the hit count is a multiple of" \\
- "Break when the hit count is greater than or equal to"
\end{tabular} \\
\hline
\end{tabular}

Table 530: "Condition"
\begin{tabular}{|l|l}
\hline "Break, when true" & \begin{tabular}{l}
\(\boxed{V}\) \\
breakpoint only when the result yields TRUE. You can define a condition as a \\
valid Boolean expression. Examples: \(x>100, x[y]=z, ~ a ~ A N D ~ b, ~ b o o l V a r . ~\)
\end{tabular} \\
\hline
\end{tabular}

Tab 'Data' Requirement: This is used for the properties of a data breakpoint.

The function of data breakpoints depends on the target system. Currently, data breakpoints are possible only with the CODESYS Control Win V3.
\begin{tabular}{|l|l|}
\hline \multicolumn{2}{|l|}{ On the tab, the variable or memory address is specified for which the data breakpoint is set or will be set. } \\
\hline \begin{tabular}{l} 
"Break execution when the \\
value of the variable or \\
address changes"
\end{tabular} & \begin{tabular}{l} 
• Input of a qualified variable name \\
\(\bullet \quad . .\). \\
Variables" category \\
Examples: variable: PLC_PRG. fb_DoSth. dwVariable; address: 16\#12A,
\end{tabular} \\
\hline "Size" & \begin{tabular}{l} 
Number of bytes of the specified variable or memory address above which \\
should be monitored for changes. When a new variable or memory address is \\
specified, a value that matches the data type or memory is set automatically at \\
first. \\
Note: The "Size" and quantity depend on the target system. For the CODESYS \\
Control Win V3, a maximum of four data breakpoints with a maximum size of 8 \\
bytes can be defined. \\
Example: 4 for data type DWORD \\
Example: 2 for data type DWORD: Only the two first bytes of the variable are \\
monitored.
\end{tabular} \\
\hline
\end{tabular}

Tab 'Execution
Here. an existing breakpoint or data breakpoint can be converted into an execution point. Point Settings'
\begin{tabular}{|l|l|}
\hline "Execution point (execution \\
does not stop at breakpoint)" & \begin{tabular}{l} 
l. The breakpoint becomes an execution point. Processing does not halt at this \\
point and the given code is executed. \\
- Execution point of a breakpoint: activated: \\
- Execution point of a data breakpoint: activated: \(;\); deactivated:
\end{tabular} \\
\hline "Execute the following code" & \begin{tabular}{l} 
Code that is executed when the execution point is reached. \\
Looping structures (For, While) and IF or CASE expressions are not possible.
\end{tabular} \\
\hline \begin{tabular}{l} 
"Print a message in the device \\
log"
\end{tabular} & \begin{tabular}{l} 
This option is available only when you select the "Enable logging in breakpoints" \\
option in "Project Settings \(\rightarrow\) Compile Options". \\
CODESYS can print variables with the placeholder \{variable name\} in the mes- \\
sage text.
\end{tabular} \\
\hline
\end{tabular}

Tab 'Location' Requirement: The command "New breakpoint" was selected.
\begin{tabular}{|l|l|}
\hline "POU" & POU of the active application where the breakpoint is placed. \\
\hline "Position" & \begin{tabular}{l} 
Position of the breakpoint in the POU. Entry as row and column numbers (text \\
editor) or as network or element numbers.
\end{tabular} \\
\hline "Instances" & \begin{tabular}{l} 
In the case of function blocks, you have to define whether the breakpoint should \\
be set in the implementation or in an instance.
\end{tabular} \\
& \begin{tabular}{l} 
"Instance Path". \\
\\
\(\square\) CODESYS sets the breakpoint in the implementation.
\end{tabular} \\
\hline
\end{tabular}
"Enable breakpoint immediately"
The breakpoint is activated.
\(\square\) : The breakpoint is not activated. To activate later, click the 西 button in the "Breakpoints" view.

See also
- « \({ }^{*}\) Chapter 6.4.1.21.4.6 "Dialog 'Breakpoint Properties"' on page 2747
- \& Chapter 6.4.1.12.3 "Using Breakpoints" on page 1981

\section*{Dialog 'Monitoring Range'}

Function: This dialog restricts the range of array elements whose values are displayed during monitoring.

Call: Click in the column field "Data Type" that belongs to the array variable.
Requirement: A POU is in online mode and is being monitored. In addition, a variable of the POU has the data type "ARRAY".
\begin{tabular}{|l|l|}
\hline "Valid range" & \begin{tabular}{l} 
The validity range of the array elements that are monitored. \\
Example of a three-dimensional array: \([1 \ldots 10][-3 \ldots 3][-10 \ldots 10]\)
\end{tabular} \\
\hline \begin{tabular}{l} 
"Maximum number of array \\
elements"
\end{tabular} & \begin{tabular}{l} 
Number of elements of the array variables \\
Example: 1470
\end{tabular} \\
\begin{tabular}{l} 
When you edit one of the settings "Start", "End", or "Scroll range of 1000 elements", both of the other settings are \\
adapted automatically.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Start" & Index of the first array element whose value is displayed. \\
\hline "End" & Index of the last array element whose value is displayed. \\
\hline \begin{tabular}{l} 
"Scroll range of 1000 \\
elements"
\end{tabular} & Scrollbar for selecting a range from the set of array elements. \\
\hline
\end{tabular}

See also
- © Chapter 6.4.1.13.2.2 "Calling of monitoring in programming objects " on page 1996

\section*{Dialog 'Properties'}
6.4.1.21.4.11.1 General ..... 2753
6.4.1.21.4.11.2 Dialog Box 'Properties' - 'Common' ..... 2753
6.4.1.21.4.11.3 Dialog 'Properties' - 'Boot Application' ..... 2754
6.4.1.21.4.11.4 Dialog: Properties: Security. ..... 2754
6.4.1.21.4.11.5 Dialog 'Properties' - 'Build' ..... 2755
6.4.1.21.4.11.6 Dialog 'Properties' - 'Build' (C-integration). ..... 2756
6.4.1.21.4.11.7 Dialog 'Properties' - 'Access Control' ..... 2757
6.4.1.21.4.11.8 Dialog 'Properties' - 'External file' ..... 2757
6.4.1.21.4.11.9 Dialog Box 'Properties' - 'Bitmap'. ..... 2758
6.4.1.21.4.11.10 Dialog 'Properties - Application Build Options'. ..... 2759
6.4.1.21.4.11.11 Dialog 'Properties' - 'Target memory settings'. ..... 2759
6.4.1.21.4.11.12 Dialog 'Properties' - 'Network Variables' ..... 2760
6.4.1.21.4.11.13 Dialog 'Properties' - 'Network Settings'. ..... 2761
6.4.1.21.4.11.14 Dialog 'Properties' - 'CFC Execution Order' ..... 2761
6.4.1.21.4.11.15 Dialog 'Properties' - 'SFC Settings'. ..... 2762
6.4.1.21.4.11.16 Dialog 'Properties' - 'Link to File'. ..... 2763
6.4.1.21.4.11.17 Dialog 'Properties' - 'Cam' ..... 2763
6.4.1.21.4.11.18 Dialog 'Properties' - 'Image Pool' ..... 2764
6.4.1.21.4.11.19 Dialog 'Properties' - 'TextList' ..... 2765
6.4.1.21.4.11.20 Dialog 'Properties' - 'Options' ..... 2765
6.4.1.21.4.11.21 Dialog 'Properties' - 'Monitoring'. ..... 2766

\section*{General}

This dialog box is for the configuration of the properties of an object in CODESYS. In addition, depending on the object, it contains different tabs that each handle a category of properties.
Call: Menu "View", context menu of the object in the "Devices", "POUs" or "Modules" view.

\section*{Dialog Box 'Properties' - 'Common'}

Function: This dialog box shows common information about the selected object.
Call: Main menu "View \(\boldsymbol{\rightarrow}\) Properties", or context menu of the object ("Common").
Requirement: An object is selected in the device tree or POUs view.

Table 531
\begin{tabular}{|l|l|}
\hline "Name" & Object name as shown in the device tree or POUs view \\
\hline "Object type" " & Type of object (for example, POU, application, or interface) \\
\hline "Open with" & Type of editor to display or edit the object \\
\hline
\end{tabular}

\section*{Dialog 'Properties' - 'Boot Application'}

Function: The settings on this tab define when and how a boot application is created from the application.
Requirement: The device supports the settings.
Call: Select the application object; context menu: "Properties"; menu bar: "View \(\rightarrow\) Properties", "Boot Application" category
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
"Create implicit boot \\
application on download"
\end{tabular} & A boot application is created automatically when downloading the application. \\
\hline \begin{tabular}{l} 
"Create implicit boot \\
application on Online Change"
\end{tabular} & A boot application is created automatically when for an online change. \\
\hline \begin{tabular}{l} 
"Remind boot application on \\
project close"
\end{tabular} & Before closing the project, CODESYS prompts to create the boot application. \\
\hline \begin{tabular}{l} 
"Verify boot application after \\
creation"
\end{tabular} & \begin{tabular}{l} 
After the boot application is created, an independent service checks whether or \\
not the boot application has been created correctly.
\end{tabular} \\
\hline
\end{tabular}

Regardless of the presets defined here, you are always able to create a boot application explicitly when you login.

See also
- \(\boldsymbol{y y}^{4}\) Chapter 6.4.1.11.7 "Generating boot applications" on page 1978
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.7.6 "Command 'Online Change'" on page 2629

\section*{Dialog: Properties: Security}

Function: The dialog contains the properties of the application for encryption. If the CODESYS Security Agent is installed, then you can start a wizard for the encryption of downloads, online changes, and boot applications.
Call:
- Menu bar: "View \(\rightarrow\) Properties"
- Context menu of an application object

Table 532: "Encryption Technology"
If the "Enforce encryption of downloads, online changes and boot applications" option is selected in the "Security Screen" view in the "Security level" group, then the encryption technology is set to "Encryption with certificates" and cannot be changed in this dialog.
"No Encryption"
\begin{tabular}{|l|l|}
\hline "Simple Encryption" & \begin{tabular}{l} 
You can download the boot application to the controller only when the defined \\
dongle (license key) is connected to the computer. \\
The dongle is provided by ABB AG or by the respective hardware manufacturer. \\
The firmcode is displayed. Type in the delivered product code.
\end{tabular} \\
\hline \begin{tabular}{l} 
"Encryption with license \\
management"
\end{tabular} & \begin{tabular}{l} 
You can download the boot application to the controller only after you have \\
specified the product code and firmcode, and the respective dongle is connected \\
to both the development computer and the controller. You receive the codes from \\
the vendor that manages the licenses.
\end{tabular} \\
\hline "Encryption with certificates" & \begin{tabular}{l} 
You can download the boot application to the controller only when a valid certifi- \\
cate exists for it. The "Certificates" group is enabled. See the description below. \\
The option is already selected if the "Enforce encryption of downloads, online \\
changes and boot applications" option is selected on the "User" tab of the \\
"Security Screen" view. \\
You can also select the "Digitally sign application code" option.
\end{tabular} \\
\hline
\end{tabular}

Table 533: "Certificates"
Note: If the "Enforce encryption of downloads, online changes and boot applications" option is selected in the "Security Screen" view in the "Security Level" group, then the encryption technology is set to "Encryption with certificates" and cannot be changed in the "Properties" dialog.
\begin{tabular}{|c|c|}
\hline 7 & The "Certificate Selection" dialog opens. Here you can select previously installed certificates of devices for which the encryption of download, online change, and boot application is enabled. The list can contain several entries if several devices are authorized to run this application. \\
\hline "Digitally sign application code" & The application is signed with a digital signature. The certificate for the digital signature is specified in the "Security Screen" view on the "User" tab. \\
\hline Area for the display of the selected certificates with corresponding information & \begin{tabular}{l}
Information per certificate: \\
- "Issued for" \\
- "Issued by" \\
- "Valid from" \\
- "Valid until" \\
- "Thumbprint"
\end{tabular} \\
\hline "Encryption Wizard" & This button is available only if the CODESYS Security Agent is installed. It starts the wizard with the same name. See the help for CODESYS Security Agent in this case. \\
\hline
\end{tabular}

See also
- \({ }^{4}\) Chapter 6.4.1.9.18 "Protecting an application" on page 1915
- « Chapter 6.4.1.21.3.4.18 "Command 'Security Screen"" on page 2592
- Help about CODESYS Security Agent

\section*{Dialog 'Properties' - 'Build'}

Symbol: 组
Function: The dialog contains options for building (build operation) the object.
Call: Menu bar: "View \(\rightarrow\) Properties"; context menu of the object in the device tree
\begin{tabular}{|c|c|}
\hline Name & Description \\
\hline "Exclude from build" & \begin{tabular}{l}
This object and recursively its child objects are not considered for the next compile process. \\
The object entry is displayed in green fonts in the "Devices" or "POUs" view.
\end{tabular} \\
\hline \begin{tabular}{l}
"External implementation" \\
"(Late link in the runtime system)"
\end{tabular} & \begin{tabular}{l}
CODESYS does not generate any code for this object when compiling the project. The object is linked as soon as the project is running on the target system, provided it is available there (for example, in a library). \\
The object name is postfixed with (EXT) in "Devices" or "POUs" view.
\end{tabular} \\
\hline "Enable system call" & \begin{tabular}{l}
A system call (runtime system) for functions is possible. \\
Background: As opposed to CoDeSys V2.3, the ADR operator in V3 can be used with function names, program names, function block names, and method names. It replaces the INSTANCE_OF operator. \\
BUT: It is not possible to call function pointers from within CODESYS.
\end{tabular} \\
\hline "Link always" & \begin{tabular}{l}
The object is marked by the compiler and therefore always included in the compile information. This means that it is always compiled and downloaded to the PLC. \\
Note: The pragma \{attribute 'linkalways'\} can also be used to instruct the compiler to always include an object.
\end{tabular} \\
\hline "Compiler defines" & \begin{tabular}{l}
Here you can specify defines or conditions for compiling the object (conditional compile). You can also type in the expression expr, which is used in these kinds of pragmas. Multiple entries are possible as a comma-separated list (see \{define \} statements). \\
Example: hello, test:='1'
\end{tabular} \\
\hline \multicolumn{2}{|l|}{"Additional compiler definitions from the device description"} \\
\hline "Defined in device" & List of compiler definitions that originate from the device description. These compiler definitions are used in the build if they are not listed in the "Ignored definitions" field. \\
\hline "Ignored definitions" & List of compiler definitions from the device description that are not used in the build. \\
\hline \(\square\) & Copies the selected compiler definition from the "Defined in device" field to the "Ignored definitions" field. \\
\hline < & Moves the selected compiler definition from the "Ignored definitions" field to the "Defined in device" field. The compiler definition is used in the build. \\
\hline
\end{tabular}

See also
- \& Chapter 6.4.1.20.6.4 "Conditional Pragmas" on page 2318
- \(\boldsymbol{y y}^{4}\) Chapter 6.4.1.20.6.3.25 "Attribute 'linkalways"' on page 2294

\section*{Dialog 'Properties' - 'Build' (C-integration)}

Function: In this dialog, you configure the build environment and the necessary data for the integration of the \(C\) development environment.

Call: Main menu "View", context menu of the object "C Code Module"
Requirement: The object "C Code Module" is selected in the device tree.

\section*{NOTICE!}

The dialog in this form is valid only for CODESYS Control Win V3 and Visual Studio. For other environments, the dialog can look different or may not even be available at all.
\begin{tabular}{|l|l|}
\hline "Visual Studio location" & \begin{tabular}{l} 
Installation path of Visual Studio on the hard disk \\
You can also select the path with the input assistant or search for it with the \\
magnifying glass.
\end{tabular} \\
\hline "Windows SDK location" & \begin{tabular}{l} 
Installation path of Windows SDK on the hard disk \\
You can also select the path with the input assistant or search for it with the \\
magnifying glass.
\end{tabular} \\
\hline \begin{tabular}{l} 
"Temporary build folder \\
Location"
\end{tabular} & Path on the hard disk for the temporary build files \\
\hline
\end{tabular}

See also

\section*{Dialog 'Properties' - 'Access Control'}

Function: The dialog defines the access rights of user groups for objects.
Call: Main menu "View \(\rightarrow\) Properties", context menu of an object in the view "Device" or "POUs".
Requirement: An object is selected in the view "Device" or in the view "POUs".
\begin{tabular}{|c|c|}
\hline "Groups, actions and permissions" & \begin{tabular}{l}
A table which displays the following user groups access rights on objects: \\
- "View" \\
- "Modify" \\
- "Remove" \\
- "Add/remove children" \\
Perform a double click on the access right symbol to open the drop down list with the available rights.
\end{tabular} \\
\hline
\end{tabular}

See also
- \& Chapter 6.4.1.6.6 "Protecting Objects in the Project by Access Rights" on page 1826
- \& Chapter 6.4.1.21.4.7 "Dialog 'Permissions'" on page 2748

\section*{Dialog 'Properties' - 'External file'}

Function: The dialog is used to view and edit the properties of the external file. The properties were defined when the object was created. Changed properties are saved by pressing the "OK" button.

Call: Menu bar: "View \(\rightarrow\) Properties"; context menu of the object
Requirement: The object of the external file is selected in the "Devices" view or the "POUs" view.

Table 534
\begin{tabular}{|l|l|}
\hline "What do you want to do with the external file?" \\
\hline "Remember the link" & The file is available in the project only as long as it exists in the defined file path. \\
\hline \begin{tabular}{l} 
"Remember the link and \\
embed into project"
\end{tabular} & \begin{tabular}{l} 
CODESYS saves an internal copy of the file in the project, as well as the link to \\
the defined file path. The update option selected below applies as long as the \\
external file exists there. Otherwise CODESYS uses the version saved in the \\
project.
\end{tabular} \\
\hline "Embed into project" & \begin{tabular}{l} 
CODESYS saves only one copy of the file in the project. There is no longer a link \\
to the external file.
\end{tabular} \\
\hline
\end{tabular}

Table 535: "When the external file changes, then"
\begin{tabular}{|l|l|}
\hline "Reload the file automatically" & If the external file changes, then CODESYS updates the file in the project. \\
\hline \begin{tabular}{l} 
"Prompt whether to reload the \\
file"
\end{tabular} & \begin{tabular}{l} 
If the external file changes, then a dialog prompt opens whether CODESYS \\
should also update the file in the project.
\end{tabular} \\
\hline "Do nothing" & The file remains unchanged in the project, even if the external file changes. \\
\hline
\end{tabular}

Table 536: "Linked File"
Requirement: Either the "Remember the link" option or the "Remember the link and embed into project" option is selected.
The following information about the linked file is displayed: "Name", "Location", "Size", "Changed".
"Display File Properties" \(\quad\) Clicking this button opens the default "Properties of <file name>" dialog, which you can also open in the Windows file system by right-clicking the file.

Table 537: "Embedded file"
Requirement: Either the "Remember the link and embed into project" option or the "Embed into project" option is selected.
Display of the following information about the embedded file: "Size" and "Changed"
"Update the embedded file" \(\boxed{\square}\) : If the external file that was added to the project is changed in the specified file path, then CODESYS updates the embedded file in the project.

See also
- \({ }^{\star}\) Chapter 6.4.1.21.2.10 "Object 'External File'" on page 2464

\section*{Dialog Box 'Properties' - 'Bitmap'}

Function: The dialog is for assigning a bitmap file to the object. The image will be used in the graphic view of the Library Manager and in the Toolbox view of the FBD/LD/IL editor.
Call: Main menu "View \(\rightarrow\) Properties", context menu of the object
Requirement: The object is selected in the view "Devices" or in the view "POUs"
\begin{tabular}{|l|l|}
\hline "Render pixels of this color & The selected color will be displayed transparently. \\
translparently: "
\end{tabular}

Render pixels of this color
The selected color will be displayed transparently.

\section*{Dialog 'Properties - Application Build Options'}

Function: The dialog includes settings that CODESYS uses for creating a boot application for the controller.

Call: Menu bar: "View \(\rightarrow\) Properties"; context menu of an application object
\begin{tabular}{|l|l|}
\hline "Download application info" & \begin{tabular}{l} 
This feature requires compiler version \(>=3.5 .0 .0\), runtime version >= \\
3.5.0.0. \\
The information about the application contents is also downloaded to \\
the PLC. We recommend that you keep this option enabled because \\
it allows for a difference check between the current application and \\
the application on the PLC. This compares the number of blocks, \\
data, and memory locations. \\
To get the information about the differences, click "Details" in the \\
"Applications" tab of the device editor. This is also in the message \\
view that opens when you are downloading an application to the \\
PLC when it is different from the one already on the PLC.
\end{tabular} \\
\hline "Stop parent application in case of & \begin{tabular}{l} 
Available for applications with a parent application.
\end{tabular} \\
\hline "Dynamic memory settings" & \begin{tabular}{l} 
Memory is allocated dynamically for the application, for example \\
when using the operator __NEW. In this case, define the "Maximum \\
size of memory (bytes)". \\
Caution: The entire memory is not available for creating objects \\
dynamically. Instead, the system always uses part of it for manage- \\
ment information.
\end{tabular} \\
\hline
\end{tabular}

See also
- \(y^{*}\) Chapter 6.4.1.11.5 "Generating Application Code" on page 1976
- \({ }^{4}\) Chapter 6.4.1.11.7 "Generating boot applications" on page 1978
- をy Chapter 6.4.1.21.2.2 "Object 'Application'" on page 2410

\section*{Dialog 'Properties' - 'Target memory settings'}

Function: The dialog allows for changing the memory settings of the target device.
Call: Menu bar: "View \(\rightarrow\) Properties"; context menu of the application
Requirement: The application is selected in the "Devices" view.
\begin{tabular}{|l|l|}
\hline "Override target memory \\
settings" & \begin{tabular}{l}
\(\checkmark:\) The memory settings stored in the device description are overridden by the \\
values specified in "Input size", "Output size", and "Memory size". \\
Note: If the memory settings of the target device are changed, then it is no \\
longer possible to log in to an existing application on the target device, nor is it \\
possible to perform an online change.
\end{tabular} \\
\hline "Input size" & \begin{tabular}{l} 
Input fields for the memory sizes used to override the values \\
"memory-layout \(\backslash\) \input-size", "memory-layout \(\backslash\) output-size", and \\
"Output size" \\
"memory-layout \\
memory-size" stored in the device description.
\end{tabular} \\
Requirement: The "Override target memory settings" option is selected.
\end{tabular}

See also
- \({ }^{\circ}\) Chapter 6.4.1.21.2.2 "Object 'Application'" on page 2410

\section*{Dialog 'Properties' - 'Network Variables'}

Symbol:
Function: In this dialog, you define network properties for the variable list that is selected in the device tree. Furthermore, any variables in it that are declared as network variables are also available.
Call: "Context menu of variable list in device tree \(\rightarrow\) Properties", "Network Variables" tab
\begin{tabular}{|l|l|}
\hline "Network type" & UDP \\
\hline "Task" & \begin{tabular}{l} 
Task of the current application that controls the variables to be sent. \\
CODESYS always sends the variables at the end of a task cycle.
\end{tabular} \\
\hline "List identifier" & Used to identify the network variable list. Must be unique \\
\hline "Pack variables" & \begin{tabular}{l} 
The size of the packages (telegrams) that are transmitted depends \\
on the network type. In the case of "UDP", a package is 256 bytes. \\
\(\boxed{V}\)
\end{tabular} \\
& \begin{tabular}{l} 
CODESYS bundles the variables for sending in packages in \\
order to reduce as much as possible the number of packages to \\
send. In the case of variables of type array or structured data types, \\
this can lead to the splitting of the variables into multiple telegrams. \\
As a result, data inconsistencies are possible within these variables, \\
even if the variable size is smaller than the package size. \\
\(\square: ~ C O D E S Y S ~ g e n e r a t e s ~ o n e ~ p a c k a g e ~ p e r ~ v a r i a b l e . ~\)
\end{tabular} \\
\hline "Transmit checksum" & \begin{tabular}{l} 
Che A checksum is provided for each variable package. The receiver \\
checks the checksum to make sure that the variable definitions \\
match from the sender and receiver. A package with non-matching \\
checksums is not accepted.
\end{tabular} \\
\hline "Acknowledgement" & \begin{tabular}{l} 
V: CODESYS sends an acknowledgement message for each \\
received data package. If the sender does not receive a confirmation \\
before it sends again, then an error is written to the diagnostic struc- \\
ture. \\
Note: For the NetVarUdp library version 3.5.7.0 and later, a receiver
\end{tabular} \\
channel is no longer assigned when confirmed transfer is not \\
selected. In this way, network variable exchange is also possible \\
between two controllers on one hardware device .
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Transmit on event", "Variable" & \begin{tabular}{l}
\(V:\) CODESYS sends the variables as soon as the defined variable \\
yields TRUE.
\end{tabular} \\
\hline "Settings" & \begin{tabular}{l} 
Protocol-specific settings; possible entries depend on the network \\
library. \\
"Port": Number of the port that CODESYS uses for data exchange \\
with other network units. The "Default value" is "1202". You can \\
change the current value in the "Value" field at any time. Select the \\
field, press the [Space Bar], and type the value. \\
Caution: The other nodes in the network must define the same port. \\
If more than one UDP connection is defined in the project, then the \\
port numbers in all configurations are adapted to this value. \\
"Broadcast Adr.": The "Default value" is 255.255 .255 .255, which \\
means that data exchange will take place with all network units. \\
You can change the current value in the "Value": select the field, \\
press the [space bar], and type the address or address range of \\
a subnetwork (for example, 197. 200.100.255 when communica- \\
tion should be with all nodes that have an IP address in the range \\
\(197.200 .100 . x\).
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\Rightarrow\) Chapter 6.4.1.10.5 "Network Variables" on page 1946
- \(\quad\) を Chapter 6.4.1.21.2.19 "Object 'Network Variable List (Receiver)"' on page 2475
- \({ }^{\circ}\) Chapter 6.4.1.21.2.18 "Object 'Network Variable List (Sender)'" on page 2475

\section*{Dialog 'Properties' - 'Network Settings'}

If the device supports the network functionality, then the current network settings for a GNVL (global network variable list) can be displayed and changed in the "Properties" dialog of the object. These are the settings that were used when adding the GNVL in the "Add Network Variable List (Receiver)" dialog.

See also
- « "Dialog Box 'Add Network Variable List (Receiver)'" on page 2475
-

\section*{Dialog 'Properties' - 'CFC Execution Order'}

Function: The tab switches the mode of the execution order for CFC objects.
Call: Context menu: "Properties" of a CFC object in the "Devices" view or "POUs" view

Tab 'CFC Execu-
tion Order'
\begin{tabular}{|c|c|}
\hline "Execution order" & In the CFC editor, you position the elements and therefore also the networks freely. Two modes are available to prevent the execution order in the CFC POU from being undefined. \\
\hline "Auto Data Flow Mode" & \begin{tabular}{l}
In this mode, the execution order is determined automatically by data flow, or in case of ambiguity, by network topology. The POUs and the outputs are numbered internally. The networks are executed from top to bottom and left to right. \\
Advantage: The automatically defined execution order is optimized by time and by cycle. You do not need any information about the internally managed execution order during the development process. \\
The following commands are provided afterwards in the "CFC \\
\(\rightarrow\) Execution Order" menu: \\
- "Display Execution Order" \\
- "Set Start of Feedback" \\
The elements in the CFC editor are displayed without markers and without numbering. It is not possible to change the execution order manually. For networks with feedback, you can also set a starting point.
\end{tabular} \\
\hline "Explicit Execution Order Mode" & \begin{tabular}{l}
In this mode, you can define the execution order explicitly. To do this, the elements are displayed in the CFC editor with markers and numbering, and menu commands are provided for defining the order. \\
The following commands are provided in the "CFC \(\rightarrow\) Execution Order" menu: \\
- "Send to Front" \\
- "Send to Back" \\
- "Move Up" \\
- "Move Down" \\
- "Set Execution Order" \\
- "Order by Data Flow" \\
- "Order by Topology" \\
Note: Up to CODESYS V3.5 SP1, this was the usual behavior of CFC POUs. Pay attention that it is your responsibility to adapt the execution order and assess the consequences and impacts. This is another reason why the execution order is always displayed.
\end{tabular} \\
\hline "Apply to All CFCs" & Changes the mode for all other CFC objects in the project to the mode selected in the list \\
\hline
\end{tabular}

See also
- \(\xi^{2}\) Chapter 6.4.1.9.5.3.2 "Automatic Execution Order by Data Flow" on page 1867
- \(\quad\) Chapter 6.4.1.21.3.13.10 "Command 'Set Start of Feedback"' on page 2688
- \(\sum_{\text {• }}\) Chapter 6.4.1.21.3.13.11 "Command 'Send to Front"" on page 2688
- \({ }^{*}>\) Chapter 6.4.1.21.3.13.12 "Command 'Send to Back"" on page 2689
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.13.13 "Command 'Move Up"' on page 2689
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.13.14 "Command 'Move Down"" on page 2690
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.13.16 "Command 'Order by Data Flow'" on page 2691
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.13.17 "Command 'Order by Topology'" on page 2691

\section*{Dialog 'Properties' - 'SFC Settings'}

Function: The dialog defines the default settings for all POUs used in the project, which are programmed in SFC.
Call: Main menu "View \(\rightarrow\) Properties", context menu of a SFC POU in the view "Device" or "POUs".

Flag
\begin{tabular}{|l|l|}
\hline List of all possible SFC flags & \begin{tabular}{l} 
"Use" \\
execution. The SFC flag is activated and will be considered in the program \\
"Declare" \\
\\
(If: The SFC flag is declared automatically. \\
If "Declare" is activated, but "Use" is not activated, the variable will be declared \\
but the flag has no effect in the program execution. \\
Hint: If you have manually declared a SFC variable you have to disable the dec- \\
laration of this flag in the "SFC Settings". Otherwise the automatically generated \\
flag will overwrite the manually declared flag. \\
Hint: A automatically declared flag variable is only visible in the online mode in \\
the declaration part of the SFC editor.
\end{tabular} \\
\hline "Use defaults" & \begin{tabular}{l} 
: The settings of this dialog overwrites the "SFC settings" of the single POUs. \\
\hline
\end{tabular} \\
\hline See also
\end{tabular}
- \# Chapter 6.4.1.20.1.5.6 "SFC Flags" on page 2067

\section*{Build}

Table 538: "Code generation"
"Calculate active transitions
CODESYS generates code for the currently active transition only. only"

\section*{Dialog 'Properties' - 'Link to File'}

Function: The dialog defines the link of an external file with the contents of the global variable list (GVL). You can either export the GVL to an external file or import it from an external file.

Call: Menu bar: "View \(\rightarrow\) Properties"; context menu of an object of type "Global Variable List"
\begin{tabular}{|l|l|}
\hline "File name" & Input field of the file path \\
\hline "Export before compile " & \begin{tabular}{l} 
冋: Before each compile of the project (for example with [F11]), CODESYS saves \\
a file with the extension gvl in the path, which is specified in the "File name" \\
field.
\end{tabular} \\
\hline "Import before compile" & \begin{tabular}{l}
\(\square:\) The export file which is specified in the "File name" field is read automati- \\
cally before each project compile. Therefore you can import a GVL which was \\
exported from another project, for example to set up a communication by means \\
of network variables.
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\Rightarrow\) Chapter 6.4.1.21.2.12 "Object 'GVL' - Global Variable List" on page 2465
- linktarget [cds_configuring_network_variables_exchange] doesn't exist but @y.link.required='true'

\section*{Dialog 'Properties' - 'Cam'}

Function: Use this dialog to define the global variables of the cam.

Table 539: "Dimensions"
\begin{tabular}{|l|l|}
\hline "Master start/end position" & \begin{tabular}{l} 
The start and end positions of the master define the range of the master values \\
and therefore the scale of the horizontal axis of the cam. The default settings are \\
given in angular degrees with 0 and 360 as limiting values.
\end{tabular} \\
\hline "Slave start/end position" & \begin{tabular}{l} 
The associated slave positions are determined by the graph type that is defined \\
for the cam. However, the segment depicted by the curves (this is also the scale \\
of the vertical axis) can be defined by the slave start and end positions given \\
here.
\end{tabular} \\
\hline
\end{tabular}

Table 540: "Period"
These settings affect the work in the cam editor and cam table. Depending on these parameters, the slave start point is adjusted automatically when the end point is changed, as well as the other way around. This adjustment optimizes the period transition to be as smooth and jerk-free as possible.
\begin{tabular}{|l|l|}
\hline "Smooth transition" & \(\boxed{\text { : The values for position, velocity, and acceleration are adjusted automatically. }}\) \\
\hline "Slave period" & \begin{tabular}{l} 
Indicates when the slave period is repeated mechanically. The slave position at \\
the start and end of the master period may then be in an interval of a whole \\
number multiple of this value. \\
This value is effective only if the "Smooth transition" check box is selected.
\end{tabular} \\
\hline
\end{tabular}

Table 541: "Continuity Requirements"
Activation of these options for the continuity of the curve does not have any effect when editing the cam. It does, however, prompt a continuity check, which reports any violations to the message view (CAM). It is not possible to edit jumps in the position curve. The default setting also requires the continuity of velocity and acceleration. You can clear these options, for example in the special case of a curve that consists of only linear segments. However, this will lead to kinks in the position curve. By default, the jerk (3rd derivative) is not tested for jumps.
\begin{tabular}{|l|l|}
\hline "Position" & \\
\hline "Velocity" & \multicolumn{1}{l}{ 目: The entire curve is tested for jumps. } \\
\hline "Acceleration" & \\
\hline "Jerk" & \\
\hline
\end{tabular}

Table 542: "Compile Format"
When compiling, MC_CAM_REF structure variables are generated. A cam is described according to the following options:
"Polynomial (XYVA)" \(\quad\)\begin{tabular}{l} 
Polynomial description of the individual points consisting of the master position, \\
\hline
\end{tabular} slave position, slave velocity, and slave acceleration.

\section*{Dialog 'Properties' - 'Image Pool'}

Function: The dialog allows for setting the basic properties of the selected image pool.
Call: "View \(\rightarrow\) Properties" of an "Image Pool" object type; context menu of an "Image Pool" object type.
\begin{tabular}{|l|l|}
\hline "Download only used images" & \multicolumn{1}{l}{ : Instead of loading all images from the image pool, CODESYS loads only the } \\
images that are actually used in the application on the PLC.
\end{tabular}

Table 543: "Symbol library settings"
\begin{tabular}{|l|l|}
\hline "Mark library as symbol library" & \begin{tabular}{l} 
Marks the image pool as a symbol library for use in a visualization. The symbol \\
library receives the key VisuSymbolLibrary = TRUE as file property in the \\
project information. The VisuElements library is inserted automatically as a \\
placeholder library in the "POUs" pool of the Library Manager. \\
Requirement: A library project is open. \\
CODESYS displays symbol libraries that are installed in the repository in the
\end{tabular} \\
"Project Settings" "Visualization" category, "Symbol Libraries" tab).
\end{tabular}

See also
- \({ }^{\text {² }}\) Chapter 6.4.1.21.2.15 "Object 'Image Pool'" on page 2468
- \(\Leftrightarrow\) Chapter 6.4.1.9.11 "Using image pools" on page 1899
- \(y^{y}\) Chapter 6.4.1.21.4.12.10 "Dialog 'Project Settings' - 'Visualization'" on page 2776

Dialog 'Properties' - 'TextList'
Function: The dialog allows for setting the basic properties of the selected text list.
Call: "View \(\rightarrow\) Properties" of an "Text List" object type; context menu of an "Image List" object type.
\begin{tabular}{|l|l|}
\hline "Download by visualization" & \(\checkmark\) : The text list is downloaded with the visualization to the controller. \\
\hline "Internal" & \(\checkmark:\) : The text list can be used only in a library. It is not available in an ordinary \\
& CODESYS project. \\
\hline
\end{tabular}

See also
- \(\boldsymbol{y}^{2}\) Chapter 6.4.1.21.2.28 "Object 'Text List"' on page 2532

\section*{Dialog 'Properties' - 'Options'}

Options (Con- Function: This dialog provides the settings for monitoring an login for objects of type device. troller) The availability of the options depends on the device description.
Call: Context menu of the device, or main menu "View \(\rightarrow\) Properties", if the device is selected.
"Monitoring interval (ms)" \(\quad\) Interval of the monitoring (10 ms - 1000 ms )

Table 544: "Interactive Login Mode"
\begin{tabular}{|l|l|}
\hline This mode is used to prevent an accidentally login to a different controller. \\
\hline "None" & \begin{tabular}{l} 
No interaction with the user during login. Corresponds to the behavior of pre- \\
vious versions.
\end{tabular} \\
\hline "Enter ID" & \begin{tabular}{l} 
During login CODESYS asks to enter an ID. The ID is stored in the controller. \\
Without a valid ID no login is possible. \\
When login a second time, CODESYS does not ask again for the ID if the \\
computer name, the user name, the device name and the device address have \\
not changed. The information is saved in the project options.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Press key" & \begin{tabular}{l} 
During login a dialog prompts and requests the user to press a key on the \\
controller. The timeout for this action is saved in the device description.
\end{tabular} \\
\hline "Wink (= blink an LED)" & During login a led blinks on the connected controller. \\
\hline
\end{tabular}

Table 545: "Symbol Configuration"
"Access variables in sync with IEC tasks"Default setting, consistent access is not permitted
\(\checkmark\) : Consistent access is permitted
The setting only will take effect when all applications and boot applications are re-downloaded to the controller.

Note: If the option is activated, then the jitter for all IEC applications may increase on this device! The consistent access can disturb the real-time capability.

Siehe also
- \(\quad \triangleq\) Chapter 6.4.1.21.2.8 "Object 'Device' and Generic Device Editor" on page 2427
- \(乡\) Chapter 6.4.1.21.2.27 "Object 'Symbol Configuration"' on page 2523

\section*{Dialog 'Properties' - 'Monitoring'}

Function: The tab contains options for the monitoring of transitions in SFC.
Call: Select transition object, click "Properties"; menu bar: "View \(\rightarrow\) Properties".
\begin{tabular}{|l|l|}
\hline "Enable monitoring" & \begin{tabular}{l}
\(\checkmark:\) An implicit variable is created for the transition, which is then always given \\
the current property value when the application calls the Transition method. The \\
value stored last in this variable is displayed in the monitoring.
\end{tabular} \\
\hline "Monitoring using call" & \begin{tabular}{l}
\(\checkmark:\) The transition to be monitored is read by directly calling the transition. \\
Note: When you activate this option, you have to consider possible side effects. \\
These kinds of side effects can occur if additional operations are implemented in \\
the transition.
\end{tabular} \\
\hline
\end{tabular}

See also
- \(y^{*}\) Chapter 6.4.1.21.2.21.11 "Object 'Transition'" on page 2499
- \({ }^{4}\) Chapter 6.4.1.20.6.3.26 "Attribute 'monitoring"' on page 2295

\section*{Dialog 'Project Settings'}
6.4.1.21.4.12.1 General. ..... 2767
6.4.1.21.4.12.2 Dialog 'Project Settings' - 'SFC' ..... 2767
6.4.1.21.4.12.3 Dialog 'Project Settings' - 'Users and Groups' ..... 2768
6.4.1.21.4.12.4 Dialog Box 'Project Settings' - 'Compileoptions' ..... 2769
6.4.1.21.4.12.5 Dialog Box 'Project Settings' - 'Compiler Warnings'. ..... 2770
6.4.1.21.4.12.6 Dialog 'Project Settings' - 'Source Download'. ..... 2770
6.4.1.21.4.12.7 Dialog 'Project Settings' - 'Page Setup'. ..... 2772
6.4.1.21.4.12.8 Dialog 'Project Settings' - 'Security'. ..... 2772
6.4.1.21.4.12.9 Dialog 'Project Settings' - 'Static Analysis Light'. ..... 2773
6.4.1.21.4.12.10 Dialog 'Project Settings' - 'Visualization' ..... 2776
6.4.1.21.4.12.11 Dialog 'Project Settings' - 'Visualization Profile'. ..... 2777

\section*{General}

Symbol:
Function: The object contains the basic configuration of the project. In the "Project Settings" dialogs the configuration can be adjusted.

Call: Double click on the "Project Settings" object in the device tree, or main menu "Project \(\rightarrow\) Project Settings".
CODESYS saves the project settings directly in the project. If a project is transferred to another system for example the "Project Settings" object is transferred as well without the need of a project archive.

The project settings are valid project wide. Dependent on the installed packages the dialogs provide settings for several categories, as for example "SFC" or "User and Groups".

See also
- \(\quad\) y Chapter 6.4.1.21.3.9.4 "Command 'Package Manager'" on page 2655

\section*{Dialog 'Project Settings' - 'SFC'}

Symbol: [
Function: This dialog is used for configuring the settings of SFC objects. The properties of each new SFC object automatically have the configured settings.
Call: Menu bar: "Project \(\rightarrow\) Project Settings" ("SFC").
Requirement: A project is open.

Tab 'Flags'
\begin{tabular}{|l|l|}
\hline \multicolumn{2}{|c|}{ Implicitly generated variables for checking and monitoring the processing in an SFC diagram } \\
\hline "Active" & \(\boxed{\square}\) : The corresponding variable is used. \\
\hline "Declare" & \begin{tabular}{l}
\(\square:\) The corresponding variable is created automatically. Otherwise, you have to \\
declare the variable explicitly if you intend to use it ("Use" is selected).
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l}
\hline "Apply to all" & \begin{tabular}{l} 
In this dialog, CODESYS applies changes to existing SFC objects. CODESYS \\
selects the "Use defaults" check box in the properties of the SFC POUs.
\end{tabular}
\end{tabular}

\section*{NOTICE!}

Automatically declared variables are visible in the declaration part of the SFC editor only in online mode.

\section*{Tab 'Build'}

Table 546: "Code Generation"
"Calculate active transitions only"

Table 547: "SFC Library"
This part of the dialog is available only for compiler versions < 3.4.1.0.
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
"Company" \\
"Title" \\
"Version"
\end{tabular} & Defines the SFC library that CODESYS uses by default. \\
\hline "Namespace" & \begin{tabular}{l} 
Enables unique references to libraries. Required when various versions of the \\
library are available on the system. Please make sure that there are no discrep- \\
ancies between the namespace defined in the library manager and the name- \\
space defined for the individual object. The SfcIec.library data is used for \\
the default settings that CODESYS provides with the default profile.
\end{tabular} \\
\hline
\end{tabular}


Each SFC block stores the information via the library version that applied when you added the block. This can cause you to use multiple library versions within the same project. In order to prevent this, you are prevented from defining specific versions of IecSfc. library (as of compiler version 3.4.1.0). The library version, which you use for all SFC blocks in the project, is defined with a placeholder. CODESYS resolves the placeholder depending on the compiler version in use. The allocation of the library version to the compiler version is defined in the library profile.

\section*{See also}
- \(\xi^{*}\) Chapter 6.4.1.20.1.5.6 "SFC Flags" on page 2067
- \& Chapter 6.4.1.17.3 "Information for Library Developers" on page 2035

\section*{Dialog 'Project Settings' - 'Users and Groups'}

Symbol: 88
Function: This dialog is for the configuration of the user management for the current project.
Call: Menu "Project \(\rightarrow\) Project Settings", category "Users and Groups"

Tab 'User'
\begin{tabular}{|l|l|}
\hline \multicolumn{2}{|l|}{ Displays the users and their memberships in groups } \\
\hline "Add" & Opens the "Add User" dialog. \\
\hline "Edit" & Opens the "Edit User" dialog. \\
\hline "Delete" & \begin{tabular}{l} 
An error message appears if you attempt to delete the last user of a group, since \\
a group must have at least one member.
\end{tabular} \\
\hline
\end{tabular}

Table 548: "Add User / Edit User"
\begin{tabular}{|l|l|}
\hline \multicolumn{2}{|l|}{ Input fields for setting up a new user account or changing an existing one } \\
\hline "Active" & \begin{tabular}{l} 
: You may use the user account, default \\
\(\square:\) The user cannot log in. If he attempts to login again with incorrect login data, \\
this can result in automatic deactivation of the account; see below: Settings.
\end{tabular} \\
\hline "Memberships" & \begin{tabular}{l} 
List of all user groups that you have defined in addition to the group "Everyone" \\
(to which each new user automatically belongs). \\
保
\end{tabular} \\
\hline
\end{tabular}

Table 549: "Export/Import"
\begin{tabular}{|l|l|}
\hline "Export Users and Groups" & \begin{tabular}{l} 
The command opens the standard dialog for saving a file in the local file system. \\
You can store the users and groups definitions of the project in a file *. users in \\
xml format.
\end{tabular} \\
\hline "Import Users and Groups" & \begin{tabular}{l} 
Export users and groups opens the standard dialog for browsing the local file \\
system for a file. Search for a file with extension *. users in order to import the \\
users and groups definitions, stored in this file, into the project.
\end{tabular} \\
\hline
\end{tabular}

\section*{Tab 'Groups'}
\begin{tabular}{|l|l|}
\hline \multicolumn{2}{|l|}{ Display of the groups and their members. A group can also be a member of a group. } \\
\hline "Add" & Opens the "Add Group" dialog. \\
\hline "Edit" & Opens the "Edit Group" dialog. \\
\hline "Delete" & \begin{tabular}{l} 
If you delete a group, the user accounts of the members remain unchanged. \\
You cannot delete the groups "Everyone" and "Owner".
\end{tabular} \\
\hline
\end{tabular}

On button "Export/Import" please see above the "User" paragraph.

\section*{Tab 'Settings'}

Display of the groups and their members in a tree structure. A group can also be a member of a group.
"Maximum number of authentication trials"
(standard) : If the user has attempted to login with an incorrect password the number of times specified here, the user account is deactivated. \(\square\) : The number of the unsuccessful attempts is unlimited
"Automatic logoff after time of inactivity"
\(\checkmark\) : The user is automatically logged out if CODESYS does not register any user actions by mouse or keyboard during the time period (minutes) specified here.

See also
- \({ }^{4}\) Chapter 6.4.1.6 "Protecting and Saving Projects" on page 1819


\section*{Dialog Box 'Project Settings' - 'Compileoptions'}

Symbol: \({ }^{\#}\)
Function: This dialog box is for configuring the compiler options.
Call: Main menu "Project \(\rightarrow\) Project Settings" ("Compileoptions" category).
Requirement: A project is open.

Table 550: "Compilerversion"
\begin{tabular}{|l|l|}
\hline "Fix Version" & \begin{tabular}{l} 
Defines the compiler version that CODESYS uses when compiling and down- \\
loading for compile (for example, "3.5.6.0" for version 3.5 SP6).
\end{tabular} \\
\hline
\end{tabular}

Table 551: "Settings"
\(\left.\begin{array}{|l|l|}\hline \begin{array}{l}\text { "Allow unicode characters for } \\ \text { identifiers" }\end{array} & \begin{array}{l}\text { Cleared by default because using Unicode characters in identifier names is not } \\ \text { permitted in the IEC standard. May be required for some foreign languages (for } \\ \text { example, Asian languages). }\end{array} \\ \hline \text { "Replace constants" } & \begin{array}{l}\text { (default): CODESYS loads the value directly for every scalar constant (not for } \\ \text { strings, arrays, and structures). In online mode, CODESYS marks the constants } \\ \text { with a symbol that is prepended to the value in the declaration editor or moni- } \\ \text { toring view. In this case, access is not possible, for example by means of an ADR } \\ \text { operator, forcing, and writing. } \\ \square\end{array} \\ \hline \text { "Access to constants is possible, but it prolongs the computation time. }\end{array}\right\}\)

Table 552: "Compiler Warnings"
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
"Maximum number of \\
warnings"
\end{tabular} & \begin{tabular}{l} 
Refers to the warnings that CODESYS prints to the messages view. \\
You define the selection of displayed compiler warnings in the "Project Settings" \\
dialog box in the "Compiler Warnings" category.
\end{tabular} \\
\hline
\end{tabular}

See also
- 学 Chapter 6.4.1.21.4.12.5 "Dialog Box 'Project Settings' - 'Compiler Warnings'" on page 2770
- \({ }^{\text {® }}\) Chapter 6.4.1.21.2.8.9 "Tab 'Log'" on page 2437

\section*{Dialog Box 'Project Settings' - 'Compiler Warnings'}

Symbol:
Function: This dialog box is used for selecting the compiler warnings that CODESYS displays in the messages view during a compile process.
Call: Call: Main menu "Project \(\rightarrow\) Project Settings" ("Compiler Warnings" category).
Requirement: A project is open.


See also
- \(\boldsymbol{y y}^{\text {• }}\) Chapter 6.4.1.21.4.12.4 "Dialog Box 'Project Settings' - 'Compileoptions"' on page 2769
- \(\quad \Leftrightarrow\) Chapter 6.4.1.21.3.6.4 "Command 'Build'" on page 2619

\section*{Dialog 'Project Settings' - 'Source Download'}

Symbol:
Function: This dialog defines the compilation and the storage of the source code as a sourcecode download archive on one or more controllers.
Call: "Project \(\rightarrow\) Project settings" menu, "Download source code" category

A source-code download archive is a project archive with the name Archiv.prj.

The settings affect the command "Online
\(\rightarrow\) Load source code to connected controller". These settings do not affect the command "File \(\rightarrow\) Load source code to controller".

Table 553: "Destination device"
Defines the location of the project archive.
\begin{tabular}{|l|l|}
\hline "<name of controller>" & \begin{tabular}{l} 
Selected controller. CODESYS loads the project archive to this controller. \\
Requirement: the project contains several controllers.
\end{tabular} \\
\hline "<all devices in project>" & CODESYS loads the project archive to all controllers in the project. \\
\hline
\end{tabular}

Table 554: "Content"
\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|l|}{Defines the contents of the project archive.} \\
\hline "Use compact download" & The project archive contains only that device in the project that contains the active application.
: The project archive contains all the devices in the project \\
\hline \multirow[t]{2}{*}{"Additional Files"} & Opens the "Additional files" dialog where you can select additional files for downloading. \\
\hline & \begin{tabular}{l}
Not all types of additional files are available for each project. \\
"Download information files" - Project information files \\
"Library profile" - Includes the applied profile \\
"Project information" - Includes the project information \\
"Referenced devices" - Includes all device descriptions of third party devices into the archive \\
"Referenced libraries" - Includes all referenced libraries into the archive \\
"Referenced visualisation styles" - Includes the used styles \\
"Visualisation profile" - Includes the used profile \\
The most important types "Referenced devices" and "Referenced libraries" should always be included, if the archive shall be usable by Automation Builder installations without availability of the required devices or libraries.
\end{tabular} \\
\hline
\end{tabular}

Table 555: "Timing"
\begin{tabular}{|l|l|}
\hline \multicolumn{1}{|l|}{ Defines the time at which CODESYS creates a project archive. } \\
\hline \begin{tabular}{l} 
"Implicitly at program download \\
and online change"
\end{tabular} & \begin{tabular}{l} 
Each time an application is loaded or an online change is made, CODESYS \\
additionally loads the project archive to the target device(s) with no further \\
prompt.
\end{tabular} \\
\hline "Implicitly at creating boot \\
project" & \begin{tabular}{l} 
Each time a boot application is created, CODESYS additionally loads the project \\
archive to the target device(s) with no further prompt.
\end{tabular} \\
\hline \begin{tabular}{l} 
"Implicitly at creating boot \\
project, download and online \\
change"
\end{tabular} & \begin{tabular}{l} 
Each time a boot application is created, an application is loaded or an online \\
change is made, CODESYS additionally loads the project archive to the target \\
device(s) with no further prompt.
\end{tabular} \\
\hline \begin{tabular}{l} 
"Prompt at program download \\
and online change"
\end{tabular} & \begin{tabular}{l} 
Each time an application is loaded or an online change is made, CODESYS \\
opens a prompt, where you can select whether CODESYS should load the \\
project archive to the controller.
\end{tabular} \\
\hline "Only on demand" & \begin{tabular}{l} 
A prompt opens only if the command "Online \\
(
\end{tabular} \\
whether CODESYS shource code to connected controller" is called. There you can select
\end{tabular}

See also
- \(\Rightarrow\) Chapter 6.4.1.21.3.7.7 "Command 'Source Download to Connected Device"" on page 2631
- \({ }^{*}\) Chapter 6.4.1.21.3.2.11 "Command 'Source Download"" on page 2560

\section*{Dialog 'Project Settings' - 'Page Setup'}

Symbol:
Function: This dialog defines the layout for the print version of the project contents. This layout is used for the printout of the project information by clicking "File \(\rightarrow\) Print" and the printout of the project documentation by clicking "Project \(\rightarrow\) Document".

Call: Main menu "Project \(\rightarrow\) Project Settings" ("Page Setup")
You can change settings the following:
- "Paper"
- "Margins"
- "Header and Footer"
- "Document"
- "Title Page"

Table 556: "Edit Header, Edit Footer"
The headers and footers are structured in table style. You can configure rows and columns, and add text and images to the resulting cells.
\begin{tabular}{|l|l|}
\hline "Row spanning" & Number of rows that CODESYS should merge into a single column. \\
\hline "Column spanning" & Number of columns that CODESYS should merge into a single row. \\
\hline\(>\) & \begin{tabular}{l} 
Opens the list of available placeholders for the "Text" field. When printing the \\
page, CODESYS provides the placeholders with the current values.
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\xi^{2}\) Chapter 6.4.1.21.3.2.14 "Command 'Page Setup'" on page 2560
- « Chapter 6.4.1.21.3.5.19 "Command 'Document' " on page 2606
- «s Chapter 6.4.1.21.3.2.12 "Command 'Print'" on page 2560

\section*{Dialog 'Project Settings' - 'Security'}

Symbol: 8
Function: this dialog is for the configuration of the project protection by a password, a dongle, or a certificate.
Call: Menu bar: "Project \(\rightarrow\) Project Settings" (category "Security").

\section*{NOTICE!}

If the encryption password is lost you can no longer open the project. You can also no longer restore it.
\begin{tabular}{|l|l|}
\hline "No protection" & \begin{tabular}{l} 
O: The project file is not protected from unauthorized access and data manipula- \\
tion. \\
Note: We strongly recommend that you use security functionality. \\
O: The "Password", "Dongle", and "Certificates" options cannot be selected.
\end{tabular} \\
\hline "Integrity check" & \begin{tabular}{l} 
When you create a new project, this option is enabled by default. \\
O: The project file is stored in a proprietary format and its integrity is checked \\
each time the project is loaded. The file may be incompatible with older versions \\
of the development system. \\
Please note that the project file is not encrypted. To better protect your data, \\
activate one of the encryption functions.
\end{tabular} \\
\hline "Encryption" & \begin{tabular}{l} 
O: The "Password", "Dongle", and "Certificates" encryption functions can be \\
selected.
\end{tabular} \\
\hline "Password" & \begin{tabular}{l} 
Entering, changing and confirming the encryption password. \\
If you save the project with these settings you must enter the password later in \\
order to open the project again, even if it is to be loaded as a library reference.
\end{tabular} \\
\hline "Dongle" & \begin{tabular}{l} 
Requirement: you have connected the CODESYS security key (dongle) to the \\
computer. \\
"Add": The dialog "Add Registered Dongle" opens.
\end{tabular} \\
\hline "Registered dongles" & Drop-down list of the registered dongles. \\
\hline "Certificates" & \begin{tabular}{l} 
Certificates are used for the encryption of contents of the open project file. \\
Requirement: The certificates for all users who share the project must be \\
installed in the local memory. \\
Ral : The "Certificate selection" dialog opens.
\end{tabular} \\
\hline
\end{tabular}

Table 557: Adding a registered dongle
\begin{tabular}{|l|l|}
\hline "Dongle" & Drop-down list of all connected dongles. \\
\hline "Update" & CODESYS refreshes the drop-down list. \\
\hline "Flash" & \begin{tabular}{l} 
The LEDs of the currently selected dongle flash for two seconds (if it supports \\
this function).
\end{tabular} \\
\hline
\end{tabular}

The dongle must be connected to the computer when CODESYS loads the project, even if it is loaded as a library reference.

\section*{Dialog 'Project Settings' - 'Static Analysis Light'}

Symbol:
Function: This dialog activates the tests that the light version of CODESYS Static Analysis performs each time code is generated.
Call: Menu bar: "Project \(\rightarrow\) Project Settings" ("Static Analysis Light" category).

You can exclude lines of code from the static code analysis by marking the code with the pragma \{analysis . . .\} or the pragma \{attribute 'analysis' := '...'\}.

Additional compile tests
\begin{tabular}{|c|c|}
\hline "SA0033: Unused variables" & \begin{tabular}{l}
Finds variables that are declared, but not used within the compiled program code. \\
For GVL variables: If there are multiple applications in one project, then only the objects under the currently active application are affected. If there is only one application, then the objects in the POUs view are also affected.
\end{tabular} \\
\hline "SA0028: Overlapping memory areas" & Detects the locations where two or more variables reserve the same storage space. For example, this occurs for the following declarations: var1 AT \%QB21: INT and var2 AT \%QD5: DWORD. In this case, both variables use byte 21 , which means that the memory range of the variables overlap. \\
\hline "SA0006: Write access from multiple tasks" & Detects variables that are written by more than one task. \\
\hline SA0004 "Multiple write access on output" & \begin{tabular}{l}
Detects outputs that are written to more than one location. \\
Note: No error is reported when an output variable (VAR_IN_OUT) is written in different branches of IF and CASE statements. \\
Note: A pragma cannot deactivate this rule.
\end{tabular} \\
\hline "SA0027: Multiple use of identifiers" & \begin{tabular}{l}
Detects multiple uses of a name/identifier for a variable or an object (POU) within the scope of a project. \\
The following cases are detected: \\
- The name of an enumeration constant is the same as in another enumeration in the application or used in an included library. \\
- The name of a variable is the same as an object in the application or an included library. \\
- The name of a variable is the same as for an enumeration constant in and enumeration in the application or an included library. \\
- The name of an object is the same as another object in the application. \\
- The name of a variable is the same as the name of a method. \\
- The name of an object is the same as the name of a superordinate object ("parent object").
\end{tabular} \\
\hline "SA0167: Temporary function block instances" & The test detects function block instances that are declared as temporary variables. This concerns instances that are declared in a method or in a function or as VAR TEMP, and therefore are reinitialized in each processing cycle and for each POŪ call. \\
\hline
\end{tabular}
```

Examples SA0003: Empty statements
;
(* Comment *);
iVar;
SA0006: Concurrent access

```
```

FUNCTION_BLOCK ADD_FB

```
FUNCTION_BLOCK ADD_FB
g_iTemp1 := g_iTemp1 + INT#1;
g_iTemp1 := g_iTemp1 + INT#1;
PROGRAM PLC_PRG //controlled by MainTask
PROGRAM PLC_PRG //controlled by MainTask
g_iTemp1 := g_iTemp1 + INT#2;
g_iTemp1 := g_iTemp1 + INT#2;
g_xTemp2 := g_iTemp1 > INT#10;
g_xTemp2 := g_iTemp1 > INT#10;
PROGRAM PLC_PRG_1 //controlled by SubTask
PROGRAM PLC_PRG_1 //controlled by SubTask
g_iTemp1 := g_iTemp1 - INT#3;
g_iTemp1 := g_iTemp1 - INT#3;
g_xTemp2 := g_iTemp1 < INT#-10;
g_xTemp2 := g_iTemp1 < INT#-10;
SA0004 Multiple write access on output
VAR_GLOBAL
    g_xVar AT %QXO.0 : BOOL ;
    g_iTest AT %QWO : INT ;
END_VAR
PROGRAM PLC_PRG
IF iCondition < INT#O THEN
    g_xVar := TRUE;
    g_iTest := INT#12;
END_IF
CASE iCondition OF
    INT#1:
        g_xVar := FALSE;
    INT#2:
        g_iTest := INT#11;
    ELSE
        g xVar := TRUE;
        g-iTest := INT#9;
END_CASE
```


## SA0006: Write access from multiple tasks

```
FUNCTION BLOCK ADD FB
g_iTemp1 := g_iTemp1 + INT#1;
PROGRAM PLC_PRG // Controlled by MainTask
g_iTemp1 := g_iTemp1 + INT#2;
g_xTemp2 := g_iTemp1 > INT#10;
PROGRAM PLC_PRG_1 //Controlled by Su.bTask
g_iTemp1 := g_iTemp1 - INT#3;
g_xTemp2 := g_iTemp1 < INT#-10;
```


## SA0027: Multiple use of name

```
PROGRAM PLC_PRG
VAR
ton : INT; // error SAO027
END_VAR
```

SA0029: Different notation in implementation and declaration
The PLC_PRG POU and a fnc function POU are in the device tree.

```
PROGRAM PLC_PRG
VAR
    iVar:INT;
        123test_var_: INT;
END_V̄AR
    ivar := iVar + 1; // notation different to that in the
declaration part -> SA0029
            123TEST_var_ := _123test_var_INT; // notation different to
that in the decla
    Fnc(); // notation different to that in the devices tree ->
SA0029
END VAR
```


## SA0167: Temporary function block instances

```
PROGRAM PLC_PRG
VAR
END_VAR
VAR_TEMP
    yafb: AFB;
END_VAR
FUNCTION Fun : INT
VAR_INPUT
END VAR
VAR
    funafb: AFB;
END_VAR
METHOD METH: INT
VAR INPUT
END VAR
VAR
    methafb: AFB;
END_VAR
```

See also

- ② Chapter 6.4.1.9.13.3 "Analyzing code statically" on page 1904


## Dialog 'Project Settings' - 'Visualization'

Symbol: ${ }^{(1)}$
Function: The dialog is used to configure the project-wide settings for objects of type "Visualization".

Call: Menu bar: "Project $\rightarrow$ Project Settings", "Visualization" category
Requirement: A project is open.

## Tab 'General'

Table 558: "Visualization Directories"

| "Text list files" | Directory which contains text lists that are available in the project to configure <br> texts for different languages. CODESYS uses the directory, for example to <br> import or export text lists. |
| :--- | :--- |
| "Image files" | After clicking $\ldots .$. , the "Select Directory" dialog opens which allows for the selec- <br> tion of a directory in the file system. |
|  | Directory which contains image files that are available in the project. Multiple <br> folders are separated with a semicolon. CODESYS uses the directory, for <br> example to import or export image files. <br> After clicking $\ldots$, the "Select Directory" dialog opens which allows for the selec- <br> tion of a directory in the file system. |

Table 559: "Advanced"

| "Activate property handling in all element properties" | You can also configure a visualization element with a property 䛼 in those of its properties in which you select an IEC variable. Then CODESYS creates additional code for the property handling when a visualization is compiled. <br> Requirement: Its IEC code contains at least an object of type "Interface property" (a property 野). <br> PLC_PRG (PRG) <br> - $\mathrm{I}_{\mathrm{C}}^{2}$ Property_A <br> Get <br> Set <br> Requirement: "Visible" is selected. |
| :---: | :---: |
| "Enable implicit checks for visualization POUs" | The implicit check is also performed for visualization POUs. As a result, additional code is generated, which increases memory usage. When memory is limited, this option should be disabled. |

See also

- Object 'Property'

Tab 'Symbol
Libraries'
Table 560: "Visualization Symbol Libraries"

| "Symbol libraries" | List of all installed symbol libraries (example: VisuSymbols) |
| :--- | :--- |
| "Assigned" | : Symbol library is selected in the project and CODESYS makes it available in <br> the "Visualization ToolBox" view of a visualization. |
|  | $\square:$ Symbol library is installed in the library repository, but CODESYS does not <br> make it available in the "Visualization ToolBox" view of a visualization. |

See also

- CODESYS Visualization
- Dialog 'Add Visualization'


## Dialog 'Project Settings' - 'Visualization Profile'

Symbol: ${ }^{\text {( }}$
Function: The dialog enables the setting of the visualization profile.

Requirement: A project is open.

Table 561: "Visualization Profile"

| "Certain profile" | Profile that CODESYS uses in the project and that determines the visualization <br> elements that are available in the project. <br> The selection list contains all the profiles installed so far. |
| :--- | :--- |

## Dialog 'Project Environment'

6.4.1.21.4.13.1 General. ..... 2778
6.4.1.21.4.13.2 Dialog 'Project Environment' - 'Library Versions'. ..... 2778
6.4.1.21.4.13.3 Dialog 'Project Environment' - 'Compiler Version'. ..... 2778
6.4.1.21.4.13.4 Dialog 'Project Environment' - 'Device Versions'. ..... 2779
6.4.1.21.4.13.5 Dialog 'Project Environment' - 'Visualization Profile' ..... 2779
6.4.1.21.4.13.6 Dialog 'Project Environment' - 'Visualization Styles' ..... 2780
6.4.1.21.4.13.7 Dialog 'Project Environment' - 'Visualization Symbols'. ..... 2780

## General

Function: You use this dialog for checking the actuality of the software and of the files, which are included in the project. CODESYS checks for example the selected compiler and finds out if there is a newer version. In such a case you can update the affected components.
Call: Main menu"Project"

## Dialog 'Project Environment' - 'Library Versions'

Function: This dialog displays the libraries of the opened project for which newer versions are available.
Call: Main menu "Project $\rightarrow$ Project Environment", tab "Library Version"
This dialog opens automatically when you open a project containing outdated libraries.

Table 562

| The list shows the name of the outdated library with version, the currently available version and the planned action. |  |
| :---: | :---: |
| "Action" | Double-click inside the field to select the desired actions. |
| "Check for updates when loading this project" | Checking takes place each time the project is opened. : Checking takes place once only. |
| "Set all to newest" | CODESYS uses the newest available version of the library. |
| "OK" | CODESYS performs the selected action(s). |

## Dialog 'Project Environment' - 'Compiler Version'

Function: This dialog shows the current compiler version of the project and provides the capability of updating.
Call: Main menu "Project $\boldsymbol{\rightarrow}$ Project Environment" ("Compiler Version" tab).

Table 563

| "Current compiler version in <br> project" | Shows the set compiler version for the open project. |
| :--- | :--- |
| "Recommended, newest <br> version" | Shows the latest version. |
| "Action" | - "Do not update": The compiler version of the project remains the same. <br> - "Update to x.x.x.x": The selected compiler version is set for the project. |
| "Check for updates when <br> loading this project" | V: CODESYS checks for new versions each time the project is opened. If there <br> is a new version, then the respective update dialog opens automatically. <br> : The compiler version is not checked. The update dialogs do not open auto- <br> matically. |
| "Set all to newest" | The compiler version is set to the latest version. |

## Dialog 'Project Environment' - 'Device Versions'

Function: This dialog shows the devices of the open project in which there are new versions available.

Call: Main menu "Project $\rightarrow$ Project Environment" ("Device Versions" tab)
This dialog opens automatically when you open a project that contains an outdated device.

Table 564

| Names of the outdated devices and their versions, as well as the current version and the planned action. |  |
| :--- | :--- |
| "Action" | Double-click in the field to select the required actions. |
| "Check for updates when | l: The check is performed when the project is opened. <br> loading this project" |
| "Set all to newest" | The check is performed one time only. |

## Dialog 'Project Environment' - 'Visualization Profile'

Function: This dialog shows the current visualization profile of the project. The profile can be updated here.

Call: Menu bar: "Project $\rightarrow$ Project Environment" ("Visualization Profile" tab).

| "Current visualization profile in <br> the project" | The set visualization profile of the open project. |
| :--- | :--- |
| "Recommended, newest <br> profile" | The newest version. |
| "Action" | $\bullet$ "Do not update": The visualization profile of the project remains unchanged. <br> - "Update to x.x.x.x": CODESYS updates the project to the selected visualiza- <br> tion profile. |
| "Check for updates when |  |
| loading this project" | Q: CODESYS checks for new profiles each time the project is opened. If there is <br> a new version, then the respective update dialog opens automatically. <br> $\square: ~ N o t ~ t e s t ~ o f ~ t h e ~ p r o f i l e ~ w h e n ~ o p e n i n g ~ t h e ~ p r o j e c t . ~ T h e ~ u p d a t e ~ d i a l o g s ~ d o ~ n o t ~$ |
| open automatically. |  |

See also

- Help about visualization


## Dialog 'Project Environment' - 'Visualization Styles'

Function: This dialog shows the current visualization style of the project and provides the capability of updating it.

Call: Menu bar: "Project $\rightarrow$ Project Environment" ("Visualization Styles" tab).

Table 565: "Newer versions are available for the following visualization styles currently in use"

| "Visualization style" | Version of the set visualization style in the open project |
| :--- | :--- |
| "Current" | Current version of the visualization style (example: 3.5.6.0) |
| "Recommended" | Recommended version of the visualization style (example: 3.5.7.0) |
| "Action" | - "Do not update": The visualization style of the project remains unchanged. <br> - "Update to x.x.x.x": CODESYS updates the project to the version of the <br> selected visualization style. |
| "Check for updates when <br> loading this project" | V: CODESYS checks for new versions each time the project is opened. If there <br> is a new version, then the respective update dialog opens automatically. <br> $\square: ~ T h e ~ v e r s i o n ~ i s ~ n o t ~ c h e c k e d . ~ T h e ~ u p d a t e ~ d i a l o g s ~ d o ~ n o t ~ o p e n ~ a u t o m a t i c a l l y . ~$ |
| "Set all to newest" | CODESYS updates the version. |

See also

- Help for visualization, section "Visualization style"


## Dialog 'Project Environment' - 'Visualization Symbols'

Function: The dialog lists installed symbol libraries and allows for you to assign symbol libraries to a project.
Call: Menu bar: "Project $\rightarrow$ Project Environment", "Visualization Symbols" tab
Requirement: The open project contains a visualization and has been saved with a compiler version < 3.5.7.0. CODESYS recognizes symbol libraries in compiler version 3.5.7.0 and higher.

| "Symbol library" | List of all installed symbol libraries |
| :--- | :--- |
| "Active" | : Symbol library is selected for the project. CODESYS provides its symbols in <br> the "Visualization Toolbox" view. <br> $\square$ |
|  | : Symbol library has been previously installed only in the library repository. |

See also

- Help for visualization, "Using the symbol library in the visualization" chapter


## Dialog 'Options'

6.4.1.21.4.14.1 General. ..... 2781
6.4.1.21.4.14.2 Dialog 'Options' - 'Automation Builder'. ..... 2781
6.4.1.21.4.14.3 Dialog 'Options' - 'C Compiler'. ..... 2782
6.4.1.21.4.14.4 Dialog 'Options' - 'CFC Editor'. ..... 2784
6.4.1.21.4.14.5 Dialog 'Options' - 'Declaration Editor'. ..... 2785
6.4.1.21.4.14.6 Dialog 'Options' - 'Device Description Download'. ..... 2785
6.4.1.21.4.14.7 Dialog 'Options' - 'Device Editor'. ..... 2786
6.4.1.21.4.14.8 Dialog 'Options' - 'Diagnosis'. ..... 2786
6.4.1.21.4.14.9 Dialog 'Options' - 'External tools'. ..... 2786
6.4.1.21.4.14.10 Dialog 'Options' - 'FBD, LD, and IL'. ..... 2787
6.4.1.21.4.14.11 Dialog 'Options' - 'Help'. ..... 2789
6.4.1.21.4.14.12 Dialog 'Options' - 'Help'. ..... 2789
6.4.1.21.4.14.13 Dialog 'Options' - 'IEC 60870-5-104'. ..... 2789
6.4.1.21.4.14.14 Dialog 'Options' - 'International Settings'........ ..... 2790
6.4.1.21.4.14.15 Dialog 'Options' - 'Libraries'. ..... 2790
6.4.1.21.4.14.16 Dialog 'Options' - 'Library Download'. ..... 2790
6.4.1.21.4.14.17 Dialog 'Options' - 'Load and Save'. ..... 2791
6.4.1.21.4.14.18 Dialog 'Options' - 'Message View'. ..... 2792
6.4.1.21.4.14.19 Dialog 'Options' - 'Monitoring'. ..... 2792
6.4.1.21.4.14.20 Dialog 'Options' - 'PLCopenXML'. ..... 2793
6.4.1.21.4.14.21 Dialog 'Options' - 'Proxy Settings' ..... 2793
6.4.1.21.4.14.22 Dialog 'Options' - 'Refactoring' ..... 2794
6.4.1.21.4.14.23 Dialog 'Options' - 'SFC Editor'. ..... 2795
6.4.1.21.4.14.24 Dialog 'Options' - 'SmartCoding'. ..... 2796
6.4.1.21.4.14.25 Dialog 'Options' - 'Startup settings'. ..... 2797
6.4.1.21.4.14.26 Dialog 'Options' - 'Text Editor'. ..... 2798

## General

Function: You use the dialog box for selecting the CODESYS options. With these options you configure the appearance and the behavior of the user interface. CODESYS saves the current configuration as standard settings in the local system.

Call: Main menu "Tools $\rightarrow$ Options"

## Dialog 'Options' - 'Automation Builder'

Symbol: $3^{*}$
Function: This dialog is for the configuration of the settings for the Automation Builder.
Call: menu "Tools $\rightarrow$ Options", category "Automation Builder"

## Tab 'DeviceTree'

| "Show warning message on delete objects" | A warning appears whether the selected object should really be deleted from the project. |
| :---: | :---: |
| "Show device type" | $\checkmark$ : In the project tree the device type is displayed in brackets |
| "Show device tag" | $\checkmark$ : Show device tag |


| "Close Add / Update object |  |
| :--- | :--- |
| dialog after single transaction" | 目: The "Add/Update Object" dialog is closed after a single transaction. |
| "Display all versions" | all : Some devices in all versions are displayed. If the checkbox is not set (default), then <br> only the latest version is displayed. |

## Tab 'Project'

| "Check integrity on open project" | The integrity of an open project is automatically checked in the background. « Chapter 6.4.1.21.3.5.10 "Command 'Check integrity"' on page 2603 |
| :---: | :---: |
| "Check configuration on the fly directly on modify" | $\square$ : The configuration can be checked directly when changing. |
| "Incremental update of configuration data" | $\checkmark$ : Performs an incremental update of configuration data. |
| "Activate legacy version of CSV signal export / import" | If this checkbox is set, the old version of the CSV signal export/import is activated. |

Tab 'Editors' Max parallel opened editors allowed 25 (max. 99).

| "//O-mapping" |  |
| :--- | :--- |
| "Use tree based I/O mapping <br> dialog" | $\square:$ The view of the I/O mapping dialog is defined, here use tree based I/O <br> mapping dialog. |
| "Use list based I/O mapping <br> dialog" | $\square:$ The view of the I/O mapping dialog is defined, here use list based I/O <br> mapping dialog. |
| "Use both I/O mapping dialog" | 凹: The view of the I/O mapping dialog is defined, here use both based I/O <br> mapping dialog. |

Tab 'General' $\square$ : "Participate in ABB usability improvement program" (Function not yet active.)

## Dialog 'Options' - 'C Compiler'

Symbol:
Function: This dialog is for the configuration of the settings for the "C Compiler".
Call: menu "Tools $\boldsymbol{\rightarrow}$ Options", category "C Compiler"

## Tab 'GCC 4.7.3'

| "Path to Compiler executable" | Path to the file location. |  |
| :---: | :---: | :---: |
|  | "...") Opens the file manager to search for the file location. |  |
|  | "Reset": Resets the input. |  |
| "Environment Variables" | "New..": A new input window opens. | "Variable name:" Enter new variables. |
|  |  | "Variable value:" Enter new variables. |
|  | "Edit...": A new input window opens. | "Variable name:" Edit new variables. |
|  |  | "Variable value:" Edit new variables. |
|  | "Delete": Deletes the entries. |  |
|  | "Reset": Deletes the entries. |  |


| "Include path" | "New..": A new input <br> window opens. | "Path:" Enter a new path. |
| :--- | :--- | :--- |
|  | "Edit...": A new input <br> window opens. | "Path:" Edit include path. |
|  | C:IProgram Files (x86)\ABB\AutomationBuilder\CCodeToolchain\FWAPI\2.11 |  |
|  | "Delete": Deletes the entries. |  |
|  | "Reset": Deletes the entries. |  |

Tab 'GCC 4.7.3
PM595-4ETH'

| "Path to Compiler executable" | Path to the file location. |  |
| :---: | :---: | :---: |
|  | "...": Opens the file manager to search for the file location. |  |
|  | "Reset": Resets the input. |  |
| "Environment Variables" | "New..": A new input window opens. | "Variable name:" Enter new variables. |
|  |  | "Variable value:" Enter new variables. |
|  | "Edit...": A new input window opens. | "Variable name:" Edit new variables. |
|  |  | "Variable value:" Edit new variables. |
|  | "Delete": Deletes the entries. |  |
|  | "Reset": Deletes the entries. |  |
| "Include path" | "New..": A new input window opens. | "Path:" Enter a new path. |
|  | "Edit...": A new input window opens. | "Path:" Edit include path. |
|  | C:\Program Files (x86)\ABB\AutomationBuilder\CCodeToolchain\FWAPI\2.11 |  |
|  | "Delete": Deletes the entries. |  |
|  | "Reset": Deletes the entries. |  |

Tab 'GCC ++
4.7.3

PM595-4ETH'

| "Path to Compiler executable" | Path to the file location. |  |
| :---: | :---: | :---: |
|  | "...": Opens the file manager to search for the file location. |  |
|  | "Reset": Resets the input. |  |
| "Environment Variables" | "New..": A new input window opens. | "Variable name:" Enter new |
|  |  | "Variable value:" Enter new |
|  | "Edit...": A new input window opens. | "Variable name:" Edit new |
|  |  | "Variable value:" Edit new |
|  | "Delete": Deletes the entries. |  |
|  | "Reset": Deletes the entries. |  |
| "Include path" | "New..": A new input window opens. | "Path:" Enter a new path. |
|  | "Edit...": A new input window opens. | "Path:" Edit include path. |
|  | C:\Program Files (x86)\ABB\AutomationBuilder\CCodeToolchain\FWAPI\2.11 |  |


|  | "Delete": Deletes the entries. |
| :--- | :--- |
|  | "Reset": Deletes the entries. |

## Tab 'External diff tool'

| "Path to external <br> diff tool" | "..": Opens the file manager to search for the file location. |
| :--- | :--- |

## Dialog 'Options' - 'CFC Editor'

Symbol: 으
Function: This dialog is for the configuration of the settings for editing and printing in the CFC editor.

Call: menu "Tools $\rightarrow$ Options", category "CFC Editor"

## Tab 'General'

| "Enable AutoConnect" | V: If you drag a CFC element onto the work area of the editor and insert it, <br> CODESYS automatically connects together unconnected pins that 'touch' one <br> another. Make sure that you do not create unwanted connections when shifting <br> elements! |
| :--- | :--- |
| "Prepare values in <br> implementation part" | in online mode you can also prepare variable values for writing and forcing <br> in the implementation part of the CFC module. In addition, CODESYS displays <br> the values you have just prepared in the inline monitoring box of the variable in <br> angle brackets. |

## Tab 'View'

| "Display grid points" | $\checkmark$ : Grid points at which you can position the elements are visible in the editor. |
| :---: | :---: |
| "Show box icon" | Existing function blocks that are linked with a bitmap are displayed by CODESYS in the CFC editor as symbols. <br> Requirement: You have either created the link for a function block or a function in the object properties or loaded it via a library. |
| "Edit Line Colors" | Opens the "Edit Line Colors" dialog for the definition of the colors of the connecting lines, depending on the data type applied. The lines appear in offline and online mode in these colors, unless CODESYS paints over them with the thick black and blue lines used to display the boolean data flow. <br> - "Add Type:" Adds a data type to the list. <br> "Delete Type" |
| "Font" | Display of the font and button for changing the font. |

- ${ }^{4}$ Chapter 6.4.1.21.4.11.9 "Dialog Box 'Properties' - 'Bitmap"" on page 2758

Tab 'Print'

| Setting the "Layout Options" |  |
| :--- | :--- |
| "Fit method" | "Page" or "Poster" |
| "Scale" | Possible values: $20 \%-200 \%$ |

See also

- ${ }^{*}$ Chapter 6.4.1.9.5.3.3 " Programming in the CFC editor" on page 1871
- $\Rightarrow$ Chapter 6.4.1.20.1 "Programming Languages and Editors" on page 2047


## Dialog 'Options' - 'Declaration Editor'

Symbol:
Function: This dialog is for the configuration of the display settings for the declaration editor.
Call: Main menu "Tools $\rightarrow$ Options", category "Declaration Editor"

| "Textual only" | Textual view of the declaration editor |
| :---: | :---: |
| "Tabular only" | Tabular view of the declaration editor |
| "Switchable between textual and tabular" | The declaration editor offers two buttons for switching between the textual and tabular views: <br> 䝉: textual view $\square$ tabular view <br> The following option defines the view that appears by default when opening a programming object: <br> - "Always textual" <br> - "Always tabular" <br> - "Remember recent setting (per object)" <br> - "'Remember recent setting (global)" |

See also

- $y^{*}$ Chapter 6.4.1.9.4.2 "Using the declaration editor" on page 1851


## Dialog 'Options' - 'Device Description Download'

Symbol:
Function: This dialog is for the configuration of addresses of download servers for device descriptions.

Call: Menu "Tools $\rightarrow$ Options", category "Download the Device Descriptions".
See also

- $\quad$ \& Chapter 6.4.1.18 "Managing devices" on page 2038

Table 566: "Download server"
List of download servers containing device descriptions. By default 'https://store.codesys.com/ CODESYSDevs' is entered as the download server.

If you select the button "Download Missing Device Descriptions" in the "Device Repository" dialog, CODESYS uses the servers entered here and uses the set login data for the proxy server.

| Double-click on "(Enter new <br> download server here...)" | An input field opens in which you can enter the URL address of a server. |
| :--- | :--- |
| $[D e l]$ | Deletes the selected download server. |

See also

- ${ }^{4} \gg$ Chapter 6.4.1.21.3.9.8 "Command 'Device Repository'" on page 2663
- ${ }_{y}{ }_{y}$ Chapter 6.4.1.21.4.14.21 "Dialog 'Options' - 'Proxy Settings"" on page 2793


## Dialog 'Options' - 'Device Editor'

Symbol:
Function: This dialog includes settings for displaying the device editor.
Call: Menu bar: "Tools $\rightarrow$ Options"; category: "Device Editor".

## Tab 'View'

| "Show generic device configuration views" | This tab with the list of device parameters is available in the device editors of parameterizable devices. |
| :---: | :---: |
| "Create cross references for IEC addresses (clean necessary)" | $\checkmark$ : CODESYS creates the cross-references for unmapped I/Os. |
| "Communication page" | - "Classic mode": The "Communication" tab of the device editors appears as a split window with the left side showing the current configured gateway channels in a tree structure and the right side showing the associated data and information. <br> - "Simple mode": The "Communication" tab appears as described in the corresponding section in the help. <br> Additional modes may also be available from customer-specific extensions. |
| "Show implicit files for application download on the editor of a PLC" | The tab for synchronized files is available in the device editors. Synchronized files are downloaded to the PLC at the time of application download. These can be external files that were added to the application, or implicit files such as a source code archive. |
| "Show access rights page" | The "Access Rights" tab is available in the device editors. <br> Note: Depending on the device, the device description may overwrite this setting. |

See also

- $\quad \triangleq$ Chapter 6.4.1.21.2.8.3 "Tab 'Communication Settings'" on page 2427
- ${ }^{*}$ Chapter 6.4.1.21.2.8.7 "Tab 'Synchronized Files'" on page 2436


## Dialog 'Options' - 'Diagnosis'

Symbol:
Function: This dialog is for the "Diagnosis" setting and views.
Call: menu "Tools $\rightarrow$ Options", category "Diagnosis"

Table 567: 'Diagnosis view'

| "Enable subtree diagnosis" | $\boxed{\square}$ : The subtree diagnosis is switched on. |
| :--- | :--- |
| "Enable debug columns" | $\square$ : Debug columns are enabled. |

## Dialog 'Options' - 'External tools'

Symbol: 礝
Function: This dialog is for setting of "External tools".
Call: menu "Tools $\rightarrow$ Options", category "External tools"

| Tool | Version |
| :---: | :---: |
| "Panel builder" | "Default" |
|  | "Custom": Opens the file manager to search for the file location. Sometimes it is required to used dedicated versions of these tools (qualified versions, versions supporting more legacy types, ...) |
| "Drive composer pro" | "Default" |
|  | "Custom": Opens the file manager to search for the file location. Sometimes it is required to used dedicated versions of these tools (qualified versions, versions supporting more legacy types, ...) |


| "Restore defaults" | Resets the custom settings to default. |
| :--- | :--- |

The modified settings will be valid after restart of Automation Builder.

## Dialog 'Options' - 'FBD, LD, and IL'

Symbol: ©
Function: This dialog is used for configuring the display options for the FBD/LD/IL editor.
Call: "Tools $\rightarrow$ Options" (category "FBD, LD, and IL".

Tab 'General'
Table 568: "View"

| "Show network title" | The network title is displayed in the upper left corner of the network. |
| :--- | :--- |
| "Show network comment" | The network comment is displayed in the upper left corner of the network. When <br> the network title is also shown in CODESYS, the comment is shown in the line <br> below. |
| "Show box icon" | The block symbol is displayed in the block element in the FBD and LD editor. <br> The standard operators also have symbols. |
| "Show operand comment" | CODESYS shows the comment that you indicated for a variable in the imple- <br> mentation part. The operand comment refers to the local occurrence of the <br> variable only, as opposed to the symbol comment. <br> This comment is truncated automatically depending on available space. <br> You can limit the comment to a defined width by activating the option "Fixed size <br> for operand fields". |
| "Show symbol comment" | The comment that you indicated for a variable or symbol in the declaration is <br> displayed in CODESYS above the variable name. You can also assign a local <br> operand comment in addition to or instead of the symbol comment. |
| "Show symbol address" | If an address is assigned to a symbol (variable), then this address is displayed <br> above the variable name. |
| "Show network separators" | A separator is displayed between the individual networks. |

Table 569: "Behavior"

| "Placeholder for new <br> operands" | The operand field of pins for the new function block is left blank (instead of <br> "???"). |
| :--- | :--- |
| "Empty operands for function <br> block pins" | Adds blank operands instead of ???. |

Table 570: "Font"
Click the input field to open the "Font" dialog.

| "Fixed size for operand fields" | : "Edit operand sizes" can be enabled. |
| :--- | :--- |
| "Edit operand sizes" | The "Operand Sizes" dialog opens for setting the number of characters and <br> lines. |

## Tab 'FBD'

Table 571: "View"

| "Networks with line breaks" | $\square$ <br> as : Display of the network with line breaks so that so that CODESYS can show <br> as possible in the current width of the window. |
| :--- | :--- |
| "Connect boxes with straight | $\boxed{\square}$ : The length of the lines between the elements are fixed and short. |
| line" |  |

Table 572: "Behavior"

| "Default network content" | Drop-down list: Contents of a new network |
| :--- | :--- |
| "After insertion select" | Drop-down list: Element that CODESYS selects after inserting a new network |

## Tab 'LD'

Table 573: "View"

| "Networks with line breaks" | Display of the network with line breaks so that so that CODESYS can show <br> as many blocks as possible in the current width of the window. |
| :--- | :--- |

Table 574: "Behavior"

| "Default network content" | Drop-down list: Contents of a new network |
| :--- | :--- |
| "After insertion select" | Drop-down list: Element that CODESYS selects after inserting a new network |

## Tab 'IL'

Table 575: "View"

| "Enable IL" | The IL implementation language is available in the development system. |
| :--- | :--- |

Table 576: "Behavior"

| "Default network content" | Drop-down list: Contents of a new network |
| :--- | :--- |
| "After insertion select" | Drop-down list: Element that CODESYS selects after inserting a new network |

## Tab 'Print'

Table 577: "Layout Options"

| "Fit method" | Drop-down list for resizing. |
| :--- | :--- |
| "Avoid cutting of elements" | Elements that do not fit on the page are printed on the next page. |
| "Mark connections on adjacent <br> pages" | Enabled for selection when "Avoid cutting of elements" is selected. |

See also

- $\Leftrightarrow$ Chapter 6.4.1.20.1 "Programming Languages and Editors" on page 2047


## Dialog 'Options' - 'Help'

Symbol: (10
Function: This dialog defines whether CODESYS Online Help or CODESYS Offline Help opens when help is called.
Call: Menu bar: "Tools $\rightarrow$ Options"; category: "Help".
"Use CODESYS Online Help, if
available" $\quad \begin{aligned} & \text { default setting }\end{aligned}$ default setting.

- $\quad \square$ CODESYS Offline Help opens when CODESYS Help is called.

See also
-

## Dialog 'Options' - 'Help'

Symbol: (1)
Function: This dialog activates the online help if available.
Call: menu "Tools $\rightarrow$ Options", category "Help"

## "Use Online Help if available".

## Dialog 'Options' - 'IEC 60870-5-104'

Symbol: \#
Function: In this dialog you can set this notation of the "Address format".
Call: menu "Tools $\rightarrow$ Options", category "IEC 60870-5-104"

## Tab 'Adress for- <br> mat'

$\checkmark 1.2 .3$ (separated by dots)
$\square$ 1-2-3 (separated by hyphens)
66051 (decimal number, big endian --> 0x10203)
197121 (decimal number, little endian --> 0x30201)

## Dialog 'Options' - 'International Settings'

## Symbol:

Function: This dialog is for the setting of the language in the user interface and in the help.
Call: Menu bar: "Tools $\boldsymbol{\rightarrow}$ Options", category "International Settings".

## Dialog 'Options' - 'Libraries'

Symbol:
Function: This dialog helps you to manage the mappings of library references that CODESYS uses during the conversion of an old project. If you have not yet stored any mapping for a certain library, you must redefine the mapping each time when opening an old project in which this library is integrated.
Call: Menu bar: "Tools $\boldsymbol{\rightarrow}$ Options"; category: "Libraries".
A mapping defines what a library reference looks like following the conversion of the project to the current format. There are three possibilities:

- You retain the reference. This means that CODESYS similarly converts the library into the current format (*.library) and installs it in the local library repository.
- You replace a reference with another reference. This means that one of the installed libraries replaces the library that was integrated until now.
- You delete the reference. This means that the converted project no longer integrates the library.

CODESYS applies all the listed mappings to the library references of an old project the next time it is converted. Hence, you must repeat the mapping definition if the same library is integrated again in a project that is to be converted. You can enter a new mapping in the last line.

| "Source Library" | Path of the library that is integrated in the project before the conversion. <br> A double-click an entry makes the field editable and the button for the input <br> assistance appears. |
| :--- | :--- |
| "Target Library" | Name and location of the library that is to be integrated in the project after the <br> conversion. <br> A double-click an entry opens the dialog "Set target system library". |

Table 578: "Set target system library"

| "Scan" | The "Select Library" dialog opens. You can select a library from the library repo- <br> sitory here. The dialog corresponds to the dialog in the library repository. |
| :--- | :--- |
| "Ignore" | When CODESYS converts the project, CODESYS always removes the existing <br> source library from the project. |

## Dialog 'Options' - 'Library Download'

Symbol 值
Function: This dialog is for the setting of download servers.
Call: menu "Tools $\boldsymbol{\rightarrow}$ Options", "Library Download" category

If you click on the button "Download Missing Libraries" in the library manager, CODESYS browses these download servers for libraries marked as missing in the library manager and uses the set login details for the proxy server.

| "Download servers" | URL of a server containing library files |
| :--- | :--- |
| Double-click on "(Enter new <br> download server here)" | An input field opens in which you can enter the URL address of a server. |

See also

- ${ }^{*}$ Chapter 6.4.1.21.2.16 "Object 'Library Manager'" on page 2469
- $\Leftrightarrow$ Chapter 6.4.1.21.4.14.21 "Dialog 'Options' - 'Proxy Settings'" on page 2793


## Dialog 'Options' - 'Load and Save'

Symbol:
Function: The dialog contains settings for the behavior of CODESYS when loading and saving a project.
Call: Menu bar: "Tools $\rightarrow$ Options", "Load and Save" category

| "Create backup files" | Each time the project is saved, CODESYS also saves the project as the file <project name>.project in addition to the file <project name>. backup. You can rename the backup file and open it in the programming system. |
| :---: | :---: |
| "Automatically save every ... minutes" | CODESYS automatically saves the project at the specified time intervals in a file <project name>. autosave, which you can reload following non-regular closing of the programming system. <br> CODESYS deletes the . autosave file whenever the project is closed or saved regularly. CODESYS retains the . autosave file in the case of an irregular termination. When you open a project for which there is an associated autosave file, the "Auto Save Backup" dialog opens. In this dialog you select whether the . autosave file or the version of the project last saved by the user should be opened. |
| "Save before build" | CODESYS saves the project automatically before each build operation. |
| "Create project recovery information" | Requirement: The "No protection" option is selected in the project settings in the "Security" category. This means that the project is not protected against unauthorized access and data manipulation, and there is no integrity check when the project is loaded. $\square$ If a project crashes during editing, then the next time the project is opened, a prompt is displayed asking whether or not you want to restore the unsaved data and create a new project file. If you click "Yes", then another dialog opens. In this dialog, you can select whether you want to open the restored project or open the project comparison. This project comparison displays the differences between the last saved project and the restored project. <br> Note: The project restore records every change on the hard disk when the change is made. If a power failure or hard disk error occurs on the hard disk during this operation, then the last change may be lost. |
| "Advanced Settings" | The "Advanced Settings" dialog opens. |


| "At startup" | List box for the startup screen of CODESYS: <br> - "Show start page": The start page of CODESYS is shown. <br> - "Load last loaded project" <br> - "Show "Open Project" dialog" <br> - "Show "New Project" dialog" <br> - "Show empty environment" |
| :---: | :---: |
| "News page" | URL that is opened by means of the command "Help $\rightarrow$ CODESYS CODESYS Homepage". <br> By default, this page is http://www.codesys.com/startpage. |

Table 579: Dialog "Advanced Settings"

| "Project compression" | Requirement: The "No protection" option is selected in the project settings in <br> the "Security" category. This means that the project is not protected against <br> unauthorized access and data manipulation, and there is no integrity check when <br> the project is loaded. <br> List box for the compression level that is used when saving the project. <br> - "Least compression - best speed (recommended)" |
| :--- | :--- |
| "Load Behavior" "Medium compression - medium speed" |  |
| - "Most compression - worst speed" |  |

## See also

-     * Chapter 6.4.1.6 "Protecting and Saving Projects" on page 1819


## Dialog 'Options' - 'Message View'

Symbol: 目
Function: In this dialog the number of messages can be determined.
Call: menu "Tools $\boldsymbol{\rightarrow}$ Options", category "Message View"
"Maximum numer of messages" xxx

Default, 500, max. 9999, min. 20

## Dialog 'Options' - 'Monitoring'

Symbol: 風
Function: This dialog includes settings for displaying the variable values in monitoring.
Call: Menu bar: "Tools $\boldsymbol{\rightarrow}$ Options"; category: "Monitoring".

Table 580: "Display Mode for Integer Variables"

| "Binary" | The value of the variable is displayed in the corresponding format in online |
| :--- | :--- |
| mode. |  |$\quad$| "Decimal" | This option corresponds to the setting of the command "Debug |
| :--- | :--- |
|  | $\rightarrow$ Display Mode". |

Table 581: "Floating Point Variables"

| "Number of displayed digits" | Decimal places that are represented in online mode when REAL values are <br> displayed. <br> Note: The settings apply to the watch list, the monitoring of the declaration editor <br> and the trace editor. The configuration for inline monitoring of the editor is set in <br> the text editor options. |
| :--- | :--- |

See also

- \& "Tab 'Monitoring'" on page 2800


## Dialog 'Options' - 'PLCopenXML'

Symbol:
Function:This dialog contains settings for the behavior of CODESYS when exporting or importing PLCopenXM.
Call: Main menu "Tools $\rightarrow$ Options", category "PLCopenXML"

Table 582: "PLCopenXML Export Settings"

| "Additionally export <br> declarations as plain text" | By default, CODESYS splits the declaration parts in accordance with the PLCo- <br> penXML scheme into individual variables and thus loses the formatting and <br> some comment information. <br> 目: Formatting and comments are retained. CODESYS additionally writes the |
| :--- | :--- |
| plain text of the exported declaration part into the PLCopenXML file and thus |  |
| extends the PLCopenXML scheme. |  |

Table 583: "PLCopenXML Import Settings"

| "Import folder structure" | $\checkmark$ : If the import file contains information about the folder structure of the objects, <br> CODESYS also imports this structure. <br> $\square$ <br> $\square$ |
| :--- | :--- |
| SODESYS imports objects without structure. |  |

## Dialog 'Options' - 'Proxy Settings'

Symbol:
Function: You use this dialog for storing the authentication data for the proxy server which is currently used for accessing the internet from CODESYS.
Call: Main menu "Tools $\rightarrow$ Options", category "Proxy Settings"

Requirement: Internet access of the network via proxy server

| "Enter proxy credentials" | A double click opens the input request for the user name and the password for <br> the proxy server. |
| :--- | :--- |
| CODESYS uses the access data for the establishment of the connection to |  |
| the download server for libraries and the device description, for the establish- |  |
| ment of the connection to the CODESYS Store and for the command "View |  |
| $\rightarrow$ Start Page". |  |
| Requirement: If the internet access of your computer or of the network takes |  |
| place via a proxy server, then the button is available. |  |

- \# Chapter 6.4.1.21.2.16 "Object 'Library Manager'" on page 2469
- \& Chapter 6.4.1.21.3.4.20 "Command 'Start Page"' on page 2596


## Dialog 'Options' - 'Refactoring'

Symbol: ที
Function: The dialog is used for defining the operations in the project for which the automatic refactoring is suggested. The refactoring functionality helps you in your improvement endeavors.

Call: Menu bar: "Tools $\boldsymbol{\rightarrow}$ Options", "Refactoring" category

## 'Suggest Refactoring for the Following Operations'

| "Auto-declare" | When you change the name of a variable in a declaration by calling AutoDeclare ([Shift]+[F2]), the activated option "Apply changes by means of refactoring" appears. Then the "Refactoring" dialog opens and you can change the variable throughout the project. <br> - "On adding or removing variables, or on changing the scope" <br> You delete the names in the "Declare Variable" dialog and click "OK" to close the dialog. Then the "Refactoring" dialog opens for removing the variable throughout the project. <br> - "On renaming variables" <br> You specify the names in the "Declare Variable" dialog and click "OK" to close the dialog. Then the "Refactoring" dialog opens for renaming the variable throughout the project. <br> See the chapter: "Refactoring", "Changing a variable declaration and applying refactoring automatically". |
| :---: | :---: |
| "Unit conversion editor" | " On renaming of unit conversions" <br> - $\square$ : When you change the name of a conversion in the unit conversion editor, you are prompted whether CODESYS should perform "Automatic Refactoring" when renaming. |
| "Mapping editor" | "On renaming variables" <br> - $\square$ : When you change a variable name in the device editor ("//O Mapping" tab), you are prompted whether CODESYS should perform "Automatic Refactoring" when renaming. |


| "Navigator" | "On renaming objects" <br> - <br> 目: When you change the name of an object in the device tree or in <br> the POUs view, you are prompted whether CODESYS should perform <br> "Automatic Refactoring" when renaming. |
| :--- | :--- |
| "Tabular declaration editor" | "On renaming variables" <br> -V: When you change the name of a variable in the tabular declaration <br> editor, you are prompted whether CODESYS should perform "Automatic <br> Refactoring" when renaming. |

See also

- $\forall$ Chapter 6.4.1.9.16 "Refactoring" on page 1910
- $\Rightarrow$ Chapter 6.4.1.9 "Programming of Applications" on page 1844
- $\boldsymbol{y}^{\boldsymbol{y}}$ Chapter 6.4.1.21.3.3.40 "Command 'Refactoring' - 'Rename <...>"' on page 2577
- $\Leftrightarrow$ Chapter 6.4.1.21.3.3.32 "Command 'Auto Declare'" on page 2572
- $\xi^{*}$ Chapter 6.4.1.21.2.35 "Object 'Unit Conversion"' on page 2549


## Dialog 'Options' - 'SFC Editor'

Symbol: 医
Function: This dialog is used for configuring the settings for the SFC editor.
Call: Menu bar: "Tools $\rightarrow$ Options" ("SFC Editor" category).
See also

- ${ }^{*}$, Chapter 6.4.1.9.5.5.2 "Programming in SFC" on page 1880
- ๕ Chapter 6.4.1.21.3.12 "Menu 'SFC"" on page 2675
- ※ Chapter 6.4.1.20.1.5.1 "SFC editor" on page 2062


## Tab 'Layout'

Table 584: "Elements"
This defines the dimensions of the SFC elements: step, action, qualifier, property. The values are given in matrix units, where one matrix unit equals the font size that you set in the text editor options (text area / font). The settings are always active immediately in all open SFC editor views.

| "Step height" | Possible values: 1-100 |
| :--- | :--- |
| "Step width" | Possible values: 2-100 |
| "Action width" | Possible values: 2-100 |
| "Qualifier width" | Possible values: 2-100 |
| "Property width" | Possible values: 2-100 |

Table 585: "Font"
The example text shows the current font. Click it to change the font.

Table 586: "Step Actions"

| "Default insertion method" | - "Copy reference": The reference to the action objects that call the step are <br> also copied when the step is copied. The copied step and new step call the <br> same action. |
| :--- | :--- |
|  | "Duplicate implementation": The reference to the action objects that call the <br> step are linked to this step. When copying the step element, new action <br> objects are created for the new step, and the implementation is duplicated. |
|  | "Always ask": When inserting a step action, you are always prompted <br> whether the actions of a step element should be duplicated when it is copied, <br> or whether the reference to the existing action should be applied. <br> Note: If a step already contains an embedded action, then new inserted <br> actions of this step are also embedded. Likewise, new inserted actions are <br> not embedded when the step already contains a non-embedded action. In <br> these cases, you are no longer prompted for a duplication mode. |

Table 587: "Embedded Objects"
"Show action and transition $\quad \square$ : Action and transition objects that are embedded in the SFC box by a step are objects in the navigator" displayed in the "Devices" or "POUs" tree view.

Tab 'View'
Table 588: "Property Visibility"

| List of element properties for the categories "Common" and "Specific" with definitions of the display options. |  |
| :--- | :--- |
| "Property" | Defines the element properties displayed next to the element in the SFC dia- <br> gram. |
| "Value" | $\boxed{\square}$ : Display of the property value. |
| "With Name" | $\boxed{\square}:$ Display of the property value including name. |

Table 589: "Online"

| "Show step time" | $\boxed{\checkmark}$ : In online mode, CODESYS displays the step time to the right of the steps. |
| :--- | :--- |

## Dialog 'Options' - 'SmartCoding'

Symbol: 橉
Function: This dialog is for configuring the settings for easier coding.
Call: Menu bar: "Tools $\rightarrow$ Options", "SmartCoding" category

| "Declare unknown variables <br> automatically (AutoDeclare)" | $\boxed{\natural}$ : The "Declare Variable" dialog opens when you type an undeclared identifier <br> into an implementation language editor and then click away from the input line. <br> In order for the AutoDeclare function to be available in the ST editor as well, the <br> "Enable for ST editor" option also has to be selected. |
| :--- | :--- |
| "Enable for ST editor" | Requirement: The "Declare unknown variables automatically (AutoDeclare)" <br> option is selected. <br> $\square:$ The AutoDeclare function is also available in the ST editor. |
|  | $\square:$ The AutoDeclare function is not available in the ST editor. |


| "Show all instance variables in <br> Input Assistant" | $\boxed{\checkmark}$ : The "List components" function also lets you select the local variables of a <br> function block instance. <br> $\square$ |
| :--- | :--- |
| variables of a function block instance. |  |

See also

- ${ }^{\star}$ Chapter 6.4.1.20.1 "Programming Languages and Editors" on page 2047
- " "Smart tag functions" on page 1888
- ${ }^{*} \gg$ Chapter 6.4.1.9.14.1 "Using the cross-reference list to find occurrences" on page 1906
- $y^{4}$ Chapter 6.4.1.20.1.4.1 "ST Editor" on page 2049


## Dialog 'Options' - 'Startup settings'

Symbol: +
Function: In this dialog the "Version profile" and the "License" are set.
Call: menu "Tools $\rightarrow$ Options", category "Startup settings"

| "Version profil:" | Automation Builder 2.5 |
| :--- | :--- |
|  | $\square$ "Display selected dialog at each start": Refers to the version <br> of the AB to be displayed |


| "License:" | $\boxed{\square}$ "Default: take any available license" |
| :--- | :--- |
|  | $\square$ "Use only local license" |
|  | $\square$ "Display license selection dialog if shared licenses are |
|  | available" |

The modified settings will be valid after restart of the Automation Builder.

## Dialog 'Options' - 'Text Editor'

Symbol: []
Function: The dialog contains settings for displaying and working in a text editor.
Call: Menu bar: "Tools $\rightarrow$ Options", "Text Editor" category

Tab 'Theme' On this tab, you set the desired theme in the interface design of the ST editor.
"Theme"
Color theme for the text editor. The selected theme is shown in the "Preview" window. The available color schemes are stored in the installation directory in the Themes folder.

Tab 'Editing'

| "Number of undos" | Maximum number of editing steps that you can apply the "Edit $\rightarrow$ Undo" command to. |
| :---: | :---: |
| "Folding" | Defines the structuring of the code by indentation. <br> When you select an indentation, you can expand or collapse the indentation section by means of a plus and minus sign in front of the first line of each section. <br> - "Indent": CODESYS combines all lines that are indented in relation to the preceding line into one indentation unit. <br> - "Explicit": You mark the code segment explicitly with comments that should be combined in one indentation unit: a comment with three opening braces " $\{\{\{$ " has to be before the segment, and a comment with three closing braces "\}\}\}" has to be after the segment. The comments can contain additional text. Example: <br> IF iVar1=1 |
| "Word wrap" | - "Soft": The line break occurs at the edge of the editor window when 0 is specified for "Wrap margin". <br> - "Hard": The line break occurs after the number of characters specified for "Wrap margin". |
| " Tab width" | Number of characters |
| "Keep tabs" | CODESYS does not break up the space you have inserted with the [Tab] key into individual spaces afterwards. |
| "Indent width" | If you have selected "Smart" or "Smart with code completion" for the "AutoIndent" option, then CODESYS inserts the number of spaces at the beginning of the line. |
| "Autolndent" | - "None" <br> - "Block": A new line automatically applies the indentation of the previous line. <br> - "Smart": Lines that follow a line which contains a keyword (for example, VAR) indent automatically by the specified Indent width. <br> - "Smart with code completion": Indentation as in the case of the "Intelligent" option, but CODESYS also inserts the closing keyword (for example, END_VAR). |

## Tab 'Text Area'

| "Highlight current line" | $\boxed{\square}:$ The line where the cursor is located is highlighted. |
| :--- | :--- |
| "Matching brackets" | When the cursor is positioned before or after a bracket within a line of code, <br> the corresponding closing or opening bracket is marked by a frame. |


| "End of line markers" | ワ: The end of each editor line is marked by a small dash after the last character <br> (including spaces) of the line. |
| :--- | :--- |
| "Wrap guide" | W: When a soft or hard line break is activated, the defined line break position is <br> displayed with a vertical line. |
| "Font" | Clicking the field opens the default dialog for configuring the font. |

## Tab 'Margin'

| Settings for the left margin of the text editor window, which is separated from the input area by a vertical line: |  |
| :---: | :---: |
| "Line numbering" | The declaration and implementation parts of the editor are numbered on the left, each beginning with 1 . |
| "Highlight current line" | $\checkmark$ : The line number of the line where the cursor is located is highlighted. |
| "Show bracket scope" | $\square$ : Brackets include the lines between the keywords that open and close a construct (for example, IF and END_IF). When the option is enabled and the cursor is positioned before, after, or in one of the keywords of a construct, the bracket area is displayed with a square bracket in the margin. |
| "Mouse Actions" | You can assign one of the following actions to each of the specified mouse actions or mouse-keyboard combinations. CODESYS performs the selected action when you move the mouse to the plus or minus sign in front of the header of a bracketed area: <br> - "None": The mouse action does not trigger an action. <br> - "Select fold": CODESYS selects all lines of the bracketed area. <br> - "Toggle fold": CODESYS opens or closes the bracketed area, or if there are nested brackets, the first level of the bracketed area. <br> - "Toggle fold fully": CODESYS opens or closes all levels of a nested bracketed area. |

## Tab 'Monitoring'

| Settings for displaying the monitoring fields |  |
| :--- | :--- |
| "Enable inline monitoring" | $\square:$ Display of the monitoring fields behind the variables in online mode |
| "Number of displayed digits" | Number of comma places in the monitoring field |
| "String length" | Maximum length of string variable values in the monitoring field |

See also

- ${ }^{*}$ Chapter 6.4.1.9.5.4.2 "Programming structured text (ST)" on page 1879


## Dialog 'Customize'

6.4.1.21.4.15.1 General ..... 2800
6.4.1.21.4.15.2 Dialog 'Customize' - 'Menu'. ..... 2801
6.4.1.21.4.15.3 Dialog 'Customize' - 'Command Icons' ..... 2801
6.4.1.21.4.15.4 Dialog 'Customize' - 'Toolbars'. ..... 2802
6.4.1.21.4.15.5 Dialog Box 'Customize' - 'Keyboard' ..... 2802

## General

The dialog contains the tabs to configure the user interface.
You can reset the CODESYS settings to default by use of the "Reset" button.

## Dialog 'Customize' - 'Menu'

Function: With this dialog, you define the structure and contents of the user interface.
Call: Main menu "Tools $\rightarrow$ Customize" ("Menu").
When you click "OK" to close the dialog, the changes are visible in the menu bar of the CODESYS user interface.

Table 590: "Menu"
Display of currently defined menus, submenus, and included commands. In CODESYS, a menu or submenu caption is identified by the caption symbol (圆). The layout from top to bottom corresponds to the layout displayed later in the CODESYS menu.

| "Add Command" | Enabled when a command is selected. <br> Adds a command above the selected command. Opens the "Add Command" <br> dialog. <br> Use the "Add Command" dialog for selecting one or more commands. Left part: <br> List of categories. Right part: List of commands in the selected category. |
| :--- | :--- |
| "Add Separator" | Adds a separator above the selected command. |
| "Add Popup Menu" | Adds a popup menu above the selected menu, submenu, or command. Opens <br> the "Add Popup Menu" dialog. |
| "Edit Popup Menu" | Opens the "Edit Popup Menu" dialog. |
| "Reset" | Resets the default settings of the entire menu. |
| "Load" | Loads the settings from a stored file (<file name>.opt.menu). |

Table 591: "Add Popup Menu"

| In CODESYS, a new menu is shown in the menu bar only when the menu contains at least one command. |  |
| :--- | :--- |
| "Default text" | Select this check box when localization is available. |
| "Localized Texts" | List: Languages and localized texts. |
| "Add Language" | Opens a drop-down list of available languages. <br> In CODESYS, the selected language is displayed in the area "Localized Texts". <br> Use the "Text" column for typing the localized texts. |

See also

- $y^{4}$ Chapter 6.4.1.2.2.2 "Customizing menus" on page 1802
- ⓨ Chapter 6.4.1.21.4.15.4 "Dialog 'Customize' - 'Toolbars'" on page 2802


## Dialog 'Customize' - 'Command Icons'

Function: This dialog defines the icons of the menu commands.
Call: Menu bar: "Tools $\rightarrow$ Customize" ("Command Icons").

Table 592: "Command icon"

| "Assign" | Opens a dialog for selecting the new icon (*.ico). |
| :--- | :--- |
| "Remove" | Removes the user-defined icon. The default icon is active again. |
| "Reset" | Resets all default settings of the command icons. |
| "Load" | Loads the settings from a stored file (<file name>.opt.keyb). |
| "Save" | Saves the current settings to a file (<file name>.opt.keyb). |

See also

- \& Chapter 6.4.1.2.2.4 "Customize command icon" on page 1805


## Dialog 'Customize' - 'Toolbars'

Function: Use this dialog for generating new toolbars or customizing existing toolbars.
Call: Main menu "Tools $\boldsymbol{\rightarrow}$ Customize" ("Toolbars").
When you click " OK " to close the dialog, the changes are visible in the menu bar of the CODESYS user interface.

Table 593: "Toolbars"
Display of currently defined toolbars. In CODESYS, the associated commands are listed below each toolbar in the order they will appear in the toolbar.
Double-clicking a toolbar in the list switches to editing mode.

| "Add Toolbar" | Enabled when a toolbar is selected. <br> In CODESYS, this adds a toolbar above the selected toolbar and places the <br> cursor in the name field of the new toolbar. |
| :--- | :--- |
| "Add Command" | Enabled when you select a command or blank command entry below a toolbar. <br> Adds a command above the selected command. Opens the "Add Command" <br> dialog. <br> Use the "Add Command" dialog to select one or more commands. Left part: List <br> of categories. Right part: List of commands in the selected category. |
| "Add Separator" | Adds a separator above the selected command. |
| "Hide" | Hide the selected toolbar from the user interface. |
| "Show" | Shows the selected hidden toolbar in the CODESYS user interface. |
| "Reset" | Resets the default settings of the toolbars. |
| "Load" | Loads the settings from a stored file (<file name>.opt.tbar). |

See also

- Chapter 6.4.1.2.2.3 "Customizing toolbars" on page 1804
- « Chapter 6.4.1.21.4.15.2 "Dialog 'Customize' - 'Menu"' on page 2801


## Dialog Box 'Customize' - 'Keyboard'

Function: This dialog box is used for defining keyboard shortcuts (quick access keys or keyboard combinations) for commands.
Call: Main menu "Tools $\boldsymbol{\rightarrow}$ Customize" ("Keyboard").

Table 594: "Keyboard"

| "Shortcuts for selected <br> command" | Keyboard shortcuts for the selected command The drop-down list can include <br> more than one keyboard shortcut for the command. |
| :--- | :--- |
| "Press shortcut keys" | Input field for the keyboard shortcut of the selected field. Permitted combinations <br> include [Ctrl], [Alt], [Shift], and other keys. You clicking "Assign" to assign a <br> recorded keyboard shortcut to a selected command. |
| "Shortcut keys currently used <br> by" | Command assigned to the currently defined keyboard shortcut |


| "Reset" | Resets the default settings of the keyboard shortcuts. |
| :--- | :--- |
| "Load" | Loads the settings from a stored file (<file name>.opt.keyb). |

## Dialog 'Trace Configuration'

6.4.1.21.4.16.1 Dialog 'Advanced Trace Settings'.......................................... 2803
6.4.1.21.4.16.2 Dialog 'Trace Configuration'................................................... 2804

## Dialog 'Advanced Trace Settings'

Function: This dialog provides extended settings for recording data.
Call: "Advanced" button in "Trace Configuration" dialog, "Record Settings" subdialog
Requirement: The trace editor is open and active. The dialog "Trace configuration" is open and the top node of the trace record tree is selected so that the subdialog "Record settings" is available.

For the calculation of the values, you have to select a task in the "Trace Configuration" dialog.

The buffer size is defined as "number of samples". CODESYS calculates the time intervals that corresponds to this number and displays the result in normal fonts on the right outside the table (for example, "1h1m1s1ms"). The calculation is possible only with the help of the task configuration settings and when the task cycle time is known.

| "Measurement in every n- <br> th cycle" | Data sampling in every n task cycle <br> Default: 1; then the application performs the data sam- <br> pling in each task cycle. | Sampling interval which <br> is automatically calculated <br> from the value specified in <br> the Measurement in every <br> n -th cycle option <br> Example: 100 ms |
| :--- | :--- | :--- |
| "Recommended runtime |  |  |
| buffer size (samples)" |  |  |$\quad$| Calculated automatically and therefore recommended |
| :--- |
| number of samples which the application can store per |
| trace variable at runtime |
| This value determines the maximum ring buffer size in |
| the runtime system. This buffer size is automatically cal- |
| culated from the task cycle time and the value in the |
| "Measure in every n-th cycle" option, and the displayed. |
| This buffer size is used when the "Override runtime |
| buffer size" option is disabled. | | Maximum length of the <br> time interval during which <br> the application samples <br> data on the runtime system <br> Example: 2s |
| :--- |
| Note: The calculation is <br> possible only when the set- <br> tings of the trace configu- <br> ration are set in the "Trace <br> Configuration "dialog and <br> therefore the task cycle <br> time is known. |

$\left.\begin{array}{|l|l|l|}\hline \begin{array}{l}\text { "Override runtime buffer } \\ \text { size" }\end{array} & \begin{array}{l}\text { Enables the input field for you to specify your own } \\ \text { defined buffer size At runtime, the value which you have } \\ \text { specified here is used, not the value displayed in the } \\ \text { Recommended runtime buffer size (samples) option. } \\ \text { Example: } 100 \\ \text { At least } 10 \\ \text { V The application uses this value, not the value calcu- } \\ \text { lated by CODESYS from the Recommended runtime } \\ \text { buffer size (samples) option. }\end{array} & \begin{array}{l}\text { Maximum length of the } \\ \text { time interval during which } \\ \text { the application samples } \\ \text { data on the runtime system } \\ \text { without overwriting it. }\end{array} \\ \text { Example: } 6 \mathrm{~s}\end{array}\right\}$

Example | Column (1) notifies you about the number of samples. |
| :--- |
| Column (2) next to it displays the calculated time intervals which result. |
| _cds_img_trace_example_calculation.png |
| The information displayed in the example means that one sample is taken per task cycle in the |
| runtime system and continuously cached in a ring buffer, which can hold a maximum of 201 |
| samples. The enabled "Override runtime buffer size" option has the effect that, due to the ring |
| buffer, the first cached values are first overwritten after 100 samples. During this time, 10000 |
| samples are continuously displayed in the trace editor so that a value curve can be displayed |
| over an interval of 1 minute and almost 40 seconds. | l$l$

See also

- $\xi^{4}$ Chapter 6.4.1.21.2.31 "Object 'Trace'" on page 2541
- \# Chapter 6.4.1.13.3.3 "Creating trace configuration" on page 2010


## Dialog 'Trace Configuration'

## Symbol: ${ }^{\$ 1}$

Function: The dialog includes the trace configuration for the data recording.
Call

- "Trace $\rightarrow$ Configuration"; context menu
- Link "Configuration" in the trace editor
- Link "Add Variable" in the trace editor

Requirement: The editor of a trace object is open and active.
See also

- 乡y Chapter 6.4.1.13.3.3 "Creating trace configuration" on page 2010
- ${ }^{\circ}$ Chapter 6.4.1.13.3 "Data Recording with Trace" on page 2007
- ${ }^{*}$ Chapter 6.4.1.20.6.3.26 "Attribute 'monitoring'" on page 2295

Tree view 'Trace The tree view lists the variables that are traced and allows for access to the variable settings. Record'

| Selected trace name | The "Record Settings" subdialog is displayed on the right. |
| :--- | :--- |
| Selected trace variable | The "Variable Settings" subdialog is displayed on the right. |

Table 595: Context menu commands

| + "Add Variable" | Adds a new trace variable. The "Variable Settings" subdialog opens on the right <br> and it is partially configured. Select a variable in the input field of the "Variable" <br> setting to trace its value curve. |
| :--- | :--- |
| "Assign to Diagram" | Lists the diagrams (in the submenu on the right) where the selected variable is <br> not currently displayed. Select a diagram to display the variable there. <br> The command is available when a variable is selected in the tree view. <br> Hint: When the command is deactivated, the variable is already displayed in all <br> diagrams. |
| "Enabled" | Selected by default <br> Disabled variables are displayed as disabled. They are neither displayed nor <br> recorded. |

Tree view 'Presentation (Diagrams)'

The tree view lists the diagrams that are displayed in the trace editor and allows for access to their display mode.

| Selected node "Time axis" | The "Display Mode" subdialog for the time axis is displayed on the right. You can <br> specify the time axis display. See below. |
| :--- | :--- |
| Selected diagram name | The settings for the coordinate system of the diagram and a preview are dis- <br> played on the right. See below. |
| Selected node "Y-axis" | The "Display Mode" subdialog is displayed on the right. You can specify the axis <br> display. See below. |
| "Show variables" | The "Variable Settings" subdialog is displayed on the right. You can configure the <br> trace variable. See below. <br> Note: These are the same settings that can be accessed in the "Trace Record" <br> tree view. |
| Selected trace variable |  |

Table 596: Context menu commands

| + "Add Diagram" | Adds a new diagram below and displays it in the tree view "Presentation <br> (Diagrams)". |
| :--- | :--- |
| + "Add New Variable" | Adds a new trace variable. The "Variable Settings" subdialog opens on the right <br> and it is partially configured. Select a variable in the input field of the "Variable" <br> setting to trace its value curve. Specify its display. In addition, the variable is <br> assigned to the selected diagram. |
| "Add Existing Variable" | Lists all trace variables (in the submenu on the right) where the selected diagram <br> is not currently displayed. Select a variable in order to display it in the selected <br> diagram. <br> Hint: When the command is deactivated, all trace variables are already displayed <br> in the selected diagram. |

Subdialog 'Re- Requirement: The top node is selected in the "Trace Record" tree view. cord Settings'
$\left.\begin{array}{|l|l|}\hline \text { "Enable trigger" } & \begin{array}{l}\text { V: Triggering is enabled. The trace data is buffered at runtime only when a } \\ \text { trigger signal has been sent. You determine how the trigger signal is sent in the } \\ \text { "Trigger variable", "Trigger parameter", "Trigger edge", "Post-trigger (Samples)", } \\ \text { and "Trigger level" settings. } \\ \square: \text { Continuous display of current records }\end{array} \\ \hline \text { "Trigger variable" } & \begin{array}{l}\text { Signal that is used as a trigger. A complete instance path is required. } \\ \text { A valid trigger signal is an IEC variable, a property, a reference, a pointer, an } \\ \text { array element of the application, or an expression. Allowed types are all IEC- } \\ \text { based types except STRING, WSTRING, and ARRAY. Enumerations are allowed } \\ \text { when the base type is not STRING, WSTRING, or ARRAY The contents of a } \\ \text { pointer are not a valid signal. } \\ \text { When the runtime system uses the CmpTraceMgr component, a property that is }\end{array} \\ \hline \text { linked to the 'monitoring ' attribute can then be recorded as a variable. }\end{array}\right\}$

| "Resolution" | Unit of measure for the time stamp that is recorded per data set <br> - "ms": Time stamp (in milliseconds). <br> - " $\mu s ":$ Time stamp (in microseconds) for a task cycle time of 1 ms or less |
| :--- | :--- |
| "Automatic restart" | $\boxed{\checkmark}:$ Persistently saves the trace configuration and the last contents of the RTS <br> buffer to the target device. After the device has been restarted, the trace is <br> started automatically if the trigger has not occurred yet. |
| "Advanced" | Opens the "Advanced Trace Settings" dialog. |

See also

- $\Rightarrow$ Chapter 6.4.1.21.4.16.1 "Dialog 'Advanced Trace Settings'" on page 2803

Subdialog 'Vari- Requirement: A trace variable is selected in the "Trace Record" or "Display (Diagrams)" tree able Settings' view.

| "Variable" | Valid variable Variable; value recorded with full instance path. <br> Valid: <br> - IEC variable <br> - Property <br> - Reference <br> - Contents of the pointer <br> - Array element <br> Allowed data type <br> - IEC-based type except STRING, WSTRING, or ARRAY <br> - Enumeration when the base type is not STRING, WSTRING, or ARRAY <br> When the runtime system uses the CmpTraceMgr component, a property that is linked to the 'monitoring' attribute can then be recorded as a variable. |
| :---: | :---: |
| "Parameter" | Parameter whose data is recorded. <br> Requirement: Runtime system with CmpTraceMgr component <br> The "Input Assistant" dialog lists all valid system parameters in the "Parameters" category of the "Categories" tab. |
| $\square$ | Allows toggling between "Variable" and "Parameter" |
| "Color" | Color of the variable in the trace diagram |
| "Line type" | Display as line chart <br> - / "Line": Values are linked to form a line. <br> - ـ "Step": Values are linked in the form of steps <br> - "None": Values are not linked |
| "Point type" | Display as scatter chart <br> - - "Dot": Value is displayed as a dot <br> - $\mathbf{x}$ "Cross": Value is displayed as a cross. <br> - "None": value is not displayed |
| "Activate minimum warning" | V: Warning when less than the lower limit |
| "Critical lower limit" | If the value of the trace variable falls below the limit, the variable is displayed in the warning color. |
| "Color" | Warning color on falling below the limit |
| "Activate maximum warning" | $\checkmark$ : Warning when exceeding the upper limit |


| "Critical upper limit" | If the value of the trace variable exceeds the upper limit, the variable is displayed <br> in the warning color. |
| :--- | :--- |
| "Color" | Warning color on exceeding the limit |

Subdialog 'Dis- Requirement: An axis is selected in the tree view "Presentation (Diagrams)" play Mode'

| "Display Mode" | Scaling <br> - "Auto": <br> Automatically scaled time axis <br> - "Fixed length": <br> Time axis segment with a constant "Length" <br> - "Fixed" <br> Time axis segment from "Minimum" to "Maximum" |
| :---: | :---: |
| "Minimum" | Literal, variable (integer data type), or constant variable (integer data type). It contains the initial value of the segment. Requirement: The "Display Mode" is "Fixed". <br> Examples: 20,PLC_PRG.iLimit_Min, GVL.c_iLimit_Min <br> Note: The variable has to have an initial value. This is important for the offline display and the scaling subdivision. Example: iLimit_Min : INT := 20 |
| "Maximum" | Literal, variable (integer data type), or constant variable (integer data type). It contains the end value of the segment. Requirement: The "Display Mode" is "Fixed". <br> Examples: 80,PLC_PRG.iLimit_Max, GVL.c_iLimit_Max <br> Note: The variable has to have an initial value. This is important for the offline display and the scaling subdivision. Example: iLimit_Max : INT := 80 |
| "Length" | Constant segment length; the initial value is adapted automatically. |
| "Grid" | Diagram with grid line in the X -direction. Select the grid line color from the list box of colors. |

Table 597: "Tick marks"

| "Fixed spacing" | : Display of tick marks with "Distance" and "Subdivisions". |
| :--- | :--- |
| "Distance" | Distance between tick marks |
| "Subdivisions" | Number of subdivisions between two tick marks |


| "Font" | Font for the time axis. |
| :--- | :--- |


| Link "Preview" | Displays the preview of the diagram. |
| :--- | :--- |

[^41]

| "Background color" | Background color of the diagram. $\square$ opens the list box of colors. |
| :--- | :--- |
| "Background color on <br> selection" | Background color of the selected diagram. |


| Link "Add Variable" | Adds a new trace variable (in the "Trace Record" tree view). |
| :--- | :--- |
| Link "Delete Variable" | Deletes the selected trace variable (in the "Trace Record" tree view). |
| Link "Add Diagram" | Adds a new diagram (in the "Display" tree view). |
| Link "Delete Diagram" | Deletes the selected diagram (in the "Display" tree view). |
| Link "Reset Display Settings" | Resets the display settings of either the selected diagram or Y-axis to the default <br> values. |

"OK" $\quad$ Accepts the configuration changes and saves the trace configuration.

## Dialog Box 'Trend storage'

Function: This dialog box includes the configuration for buffering the trend data of a trend recording.

Call: "Trend Storage" button in the editor of a trend recording.

| "Maximum number of variables" | Maximum number of trend variables that can be managed in the database. If you increase this value afterwards, then will CODESYS perform a download and reconfigure the database. |
| :---: | :---: |
| "Store every N milliseconds" | Time interval (in ms) when the CmpTraceMgr runtime system component buffers the recorded data before storing it persistently in the database. The application calculates internally the number of task cycles from the time interval. The duration of a task cycle is defined in the task configuration. <br> A high value results in better runtime performance. The disadvantage is the increased risk of losing data if the controller crashes or shuts down. A low value reduces this risk. The disadvantage is the slower control over a trend visualization with large amounts of data. |
| "Limit" | Limit the recording <br> - "No Limit": Unlimited number of data records (not recommended) <br> - "Maximum number of records": Maximum number of data records that are stored in the database. A data record consists of time stamp and the values of the trend variables at this time. <br> - "Maximum storage size": Maximum size of the trend storage. The application calculates internally the number of data records. <br> Clicking the "down" symbol ( $\boldsymbol{*}$ ) of the drop-down list will set the units to kilobytes (KB), megabytes (MB), or gigabytes (GB). |

See also

- ${ }^{\wedge}$ Chapter 6.4.1.21.2.33 "Object 'Trend Recording"' on page 2545
- $\&$ Chapter 6.4.1.13.4.2 "Getting started with trend recording" on page 2017


## Dialog Box 'Advanced Trend Settings'

Function: This dialog box provides more settings for configuring trend recording.
Call: Click "Advanced" in the editor of a "TrendRecording" object.

| "Measure in every n-th cycle" | Frequency that the runtime system records data, depending on the number of <br> processed task cycles. Select a value from the drop-down list or type a value into <br> the input field. <br> Using the settings from the task configuration, CODESYS calculates the time <br> interval according to the frequency. Therefore, the calculation is possible only if <br> at least the task cycle time is set. The result is shown on the right of the input <br> field in normal syntax (for example, "1h1m1s1ms"). <br> Default: 1 means that data is recorded in each task cycle. |
| :--- | :--- |
| "Additional Runtime Buffer for" | Length of the time interval when the runtime system can record more data (for <br> example, 1000 ms ). <br> If a delay occurs when writing data in the runtime system component, then there <br> is a risk of data loss due to overwriting. In this case, the runtime system uses the <br> addition buffer. |

## See also

- ${ }^{4}$, Chapter 6.4.1.21.2.33 "Object 'Trend Recording'" on page 2545
- ${ }^{\Perp}$ Chapter 6.4.1.13.4 "Data Recording with Trend" on page 2016
- ${ }^{*}>$ Chapter 6.4.1.13.4.3 "Configuring trend recording" on page 2018


## Dialog＇Certificate Selection＇

Symbol：
Function：This dialog is used for selecting the certificates for encryption，decryption，and digital signatures．

Call：
－＂Security Screen＂view，＂User＂tab
－Main menu：＂View $\rightarrow$ Properties＂，＂Security＂tab when the＂Application＂is selected in the device tree．
－Main menu：＂Project $\rightarrow$ Project Settings＂，category＂Security＂

## Dialog＇Certifi－ cate selection＇

The purpose of the certificate that is selected in the dialog depends on the call location：
－Call location：＂Security screen＂view，＂User＂tab
－Certificate for digital signatures
－Certificate for the decryption of project files
－Certificate for encrypted communication
－Call：＂View $\rightarrow$ Properties＂of the application
－＂Certificates of devices that share the encrypted download and the boot application＂
－Call location：＂Project $\rightarrow$ Project settings＂，category＂Security＂
－＂Certificate for project encryption＂

| Listing of the selected certifi－ cates in a table | The following properties are displayed for each selected X． 509 certificate： <br> －＂Issued for＂ <br> －＂Issued by＂ <br> －＂Valid from＂ <br> －＂Valid until＂ <br> －＂Thumbprint＂：SHA1 fingerprint <br> Double－clicking an entry opens the＂Certificate＂dialog with the＂General＂tab， ＂Details＂tab，and＂Certification Path＂tab．In that tab，you will find a reference to Windows help with more information about the dialog． |
| :---: | :---: |
| 里 | Adds the selected available certificate to the list of selected certificates． |
| 区 | Deletes the certificate selected in the list． |
| ＂Available certificates in the local Windows Certificate Store＂ | Double－clicking an entry opens the＂Certificate＂dialog with the＂General＂tab， ＂Details＂tab，and＂Certification Path＂tab．In that tab，you will find a reference to Windows help with more information about the dialog． |
| Certificate icons | - 圆：Certificate with a private key <br> - 漛：Untrusted certificate |

See also
－出 Chapter 6．4．1．21．3．4．18＂Command＇Security Screen＂＂on page 2592

### 6.4.2 Fieldbus Support

$\qquad$
6.4.2.2 Fieldbus Devices and I/O Drivers....................................................... 2812
6.4.2.3 Bus Cycle Task................................................................................... 2814
6.4.2.4 Symbolic Access to I/O Channels....................................................... 2815
6.4.2.5 EtherNet/IP........................................................................................ 2816

### 6.4.2.1 Device Diagnosis

CODESYS provides general and fieldbus-specific function blocks for performing a diagnosis on the connected devices.

General diag- You can perform a diagnosis on devices regardless of the fieldbus. The function blocks from the nosis CAA Device Diagnosis library are provided for this purpose.

Before you can work with these function blocks, you have to select the "Enable diagnosis for devices" option in the PLC settings. This causes CODESYS to create instances of the diagnosis functions blocks automatically. These function blocks can be used for your diagnosis.


Work exclusively with the automatically generated instances of the diagnosis function blocks. Do not create your own instances.

## See also

- ${ }^{4}$ Chapter 6.4.1.21.2.8.10 "Tab 'PLC Settings'" on page 2439
- Library CAA DeviceDiagnosis

Bus-specific For bus-specific diagnosis options, see the diagnosis chapters of the individual fieldbuses. diagnosis

### 6.4.2.2 Fieldbus Devices and I/O Drivers

The technical basis for each fieldbus device, which is configured in the device tree, is the CODESYS I/O driver.
The I/O driver is the link between the fieldbus stack, the IEC application, and the CODESYS IDE. The driver configures the fieldbus stack from the data of the device configuration. It shows the diagnosis, provides an API for the IEC application, and is responsible for the I/O mapping (see chapter "I/O Mapping").
This chapter provides a brief overview of the basic functionality of CODESYS I/O driver devices, without discussing the details of specific bus systems. In addition, some recommendations for the configuration are provided.

Bus cycle task The bus cycle task is the IEC task in whose context the I/O driver is executed. Some I/O drivers use multiple tasks: usually one real-time critical task (with high priority), which is used for the transfer of I/O data, and another task with low priority for tasks such as evaluating diagnostics and executing acyclic services of the bus system.
With real-time critical bus systems, it has to be ensured that no operations are executed in the context of this bus task that would interrupt the bus clock due to the execution time.
The bus task can be configured in the I/O mapping dialog of the I/O driver device. Note that the settings of the parent device are inherited by default. If this device is the PLC, then its PLC setting applies in the bus cycle task.

## NOTICE!

If this above setting is not set, then the task with the shortest cycle time is used. In this way, a non-real-time I/O driver can be executed unintentionally in the task context of a real-time critical driver, thus interrupting its communication. To diagnose these communication problems, it is recommended to check the task monitoring.

See also

- 出 Table 418 ""Bus Cycle Options"" on page 2440


## I/O mapping

An essential function of an CODESYS I/O driver is to update the I/O mapping. This means the mapping of the I/O data of the bus system to variables of the IEC application (and vice versa).

The input/output data is mapped cyclically by copy and conversion operations in both directions from the internal memory image of the bus system to IEC variables assigned to \%I and \%Q addresses.

For the I/O driver, there is no internal difference whether symbolic names or "direct" access to the \%I and $\%$ Q addresses are used for this I/O mapping. For the maintainability of the application, it is recommended to always use descriptive variable names (example: variable "TemperatureReactor" instead of "\%IW117" access).
The updating of the I/O mapping can be set with "Always update variables" (globally in the "PLC Settings" or individually for each device in the I/O mapping dialog):

- Disabled:

Only I/O data used in the application is mapped.
This may improve performance by avoiding the copy operations, but may cause confusion if the I/O data in the I/O mapping dialog is not updated (the values are then grayed out). This setting is recommended for an application whose development has been completed.

- Enabled 1:

All data is updated.

- Enabled 2:

Caution: For productive use in special cases only.
As a result, inconsistent I/O data may occur, because the bus cycle task reads/writes this data while the application code uses it in other tasks.
See "Consistency of I/O data".
See also

- ${ }^{*}$ ) Chapter 6.4.1.21.2.8.12 "Tab '<device name> I/O Mapping'" on page 2444

Consistency of The CODESYS programming system allows the IEC application to use multiple tasks executed I/O data in parallel (for visualization, field buses, or other POUs). The application code can access I/O data from the context of these tasks via the mapped IEC variables. By accessing the same data from different tasks, inconsistent or corrupt data could occur (for example, due to interrupted write access).
The I/O driver ensures data consistency by providing each task executing a task cycle with a consistent mapping - a snapshot, so to speak - of all I/O data used.
So a code like in the following example cannot cause problems: (Note "DIV by ZERO")

```
IF(inputData <> 0) THEN // inputData is mapped to %I
    x := y / inputData; // This will never result in
DIV_BY_ZERO Exception
END_IF-
bus cycle during execution of POU
```


## NOTICE!

With the "Always update variables" option set to "Enabled 2 - always in bus cycle task", this mechanism is overridden. Accordingly, the application code has to take this into account.

## Services In addition to the basic functionality, some I/O drivers provide services that can be called from

 the CODESYS IDE, such as the device scan function or the setting of device addresses.
## General recom- Settings:

- "PLC Settings": $\checkmark$ I/O updates in stop:

The bus cycle continues even when the application is stopped, for example when the application is on a debug breakpoint. In this way, communication with the field devices is maintained and can be continued immediately without interruption.

- "PLC Settings": "Always update variables" is set to "Enabled 1 - use bus cycle task if not used in any task":
During the development of the application, it is useful to see the values of all I/O data.
Task configuration:
- Especially for real-time critical fieldbus systems such as Profinet, EtherCAT, or CAN, which depend on maintaining an exact send/receive clock, it is recommended to use a separate bus cycle task with high priority. For less real-time-critical tasks (for example, visualization) a significantly lower priority should be selected than for the bus cycle task.
- In order to achieve maximum I/O throughput with as little offset as possible, separate POUs can be executed in the bus task of the fieldbus system. However, these then have to meet the real-time requirements: for example, no file access or blocking socket functions may be executed, but for example only the calculation of the output data.

Multiple I/O If consistent access to I/O data from multiple tasks and possibly across multiple I/O driver drivers and tasks (troubleshooting) instances has to be synchronized, then undesired reciprocal interference between the bus and application task may occur under certain circumstances.

This is the case, for example, when the general system load is high or when the I/O data of the real-time critical fieldbus system is used together with I/O data of a slow and blocking local bus system in the same task.

In case of unexpected interference of the communication, with the particularly real-time-critical fieldbuses (EtherCAT, Profinet, CAN), the task monitoring should therefore first be examined for very large jitter or outliers in the cycle time (maximum value compared to average value). The task list provides detailed information about the use of I/O data in different tasks.
It may be possible to avoid using I/O data from different bus systems in one and the same task or to reduce the number of I/O tasks.

See also

- 留 Chapter 6.4.1.21.2.8.18 "Tab 'Task deployment'" on page 2459


### 6.4.2.3 Bus Cycle Task

Generally, for each IEC task, the used input data is read at the start of each task (1) and the written output data is transferred to the I/O driver at the end of the task (3). The implementation in the I/O driver is decisive for additional transfer of the I/O data. It is responsible for the time frame and time point that the actual transfer to the corresponding bus system occurs.
The bus cycle task of the PLC can be defined globally for all fieldbuses in the PLC settings. For some fieldbuses, however, you can change this independent of the global setting. The task with the shortest cycle time is used as the bus cycle task (setting: "unspecified" in the PLC settings). The messages are normally sent on the bus in this task.

Other tasks copy only the I/O data from an internal buffer that is exchanged only with the physical hardware in the bus cycle task.

(1) Read inputs from input buffer
(3) Write outputs to output buffer
(5) Input buffer
(7) Copy data to/from bus
(9) Bus cycle task, priority 1, 1 ms
(10) Bus cycle task, priority 5
(11) Bus cycle task, priority 10, interrupted by task 5

## Task usage

The "Task Deployment" tab provides an overview of used I/O channels, the set bus cycle task, and the usage of channels.

## WARNING!

If an output is written in various tasks, then the status is undefined, as this can be overwritten in each case.
If the same inputs are used in various tasks, then it is possible for the input to change during the processing of a task. This happens when the task is interrupted by a task with a higher priority and causes the process image to be read again. Solution: At the beginning of the IEC task, copy the input variables to variables and then work only with the local variables in the rest of the code.
Conclusion: Using the same inputs and outputs in several tasks does not make any sense and can lead to unexpected reactions in some cases.

### 6.4.2.4 Symbolic Access to I/O Channels

### 6.4.2.4.1 General

You get direct access to the I/O channels of your device by means of specially generated function blocks, without having to actively set up a mapping manually for each device and for each channel.

Automated generation

- The inputs and outputs are combined into a function block under the name of the I/O channel. If possible, the basic block defined in the device description is extended.
- The channel name corresponds to the name as specified in the device editor on the I/O Mapping tab in the "Channel" column.
- The input and output variables created in this way are also displayed in the Input Assistant.
- If the I/O channels are structured or enumerations, then their function blocks are also structured accordingly. As such, the generated function blocks contain structured inputs and outputs or enumerations.
- For channels with identical names, the suffix _ $\langle X\rangle$ is appended to the input or output of the function block.
- If a device does not have its own I/O channels (example: EK1100 Slave), then no function block is created.


### 6.4.2.4.2 Enable automatic mapping

1. In the Devices view, open the top device. In the device editor, select the PLC Settings tab.

Under Additional Settings, there is the Enable symbolic access for I/Os option.
2. Select the check box.

The generated function block instance is displayed on the IEC Objects tab of the device.
For each I/O channel, a (usually extended) function block is now automatically created and instantiated. This provides you with direct access immediately. You can access the I/O channel by means of the variables of the function block.

## Example: EtherCAT Slave

The new function block extends the existing function block. The EL1008 function block instance is listed in the device editor below the EtherCAT IEC Objects tab.

Function block: EL1008_FB
The DrvEthercatLib.ETCSlave_Diag function block is extended to EL1008_FB and gets the output variables Input_<n>.

Example: If no function block is defined in the device description which can be extended, then a new Device without a function block is generated. For example, the Generic_XN1AI function block instance is listed in defined basic block

Example: The BK5120_FB function block, which extends the existing _3SCOS.CANRemoteDevice funcCANopen tion block according to the device description, is generated. The BK5120 function block instance is listed in the device editor on the CANopen IEC Objects tab.

### 6.4.2.4.3 Online Mode

In online mode, the data is updated by means of the existing functions. However, only those of the generated variables which are actually used are updated. Or if the Always update variables option is enabled.

If a channel is simultaneously accessed symbolically and via I/O mapping, then compiler warnings are issued.

### 6.4.2.5 EtherNet/IP

### 6.4.2.5.1 Introduction

EtherNet/IP is used to connect any number of different terminals in a network, using a scanneradapter relationship. The scanner can be thought of as a teacher and the adapter as a student. The scanner gives commands to the adapter in the form of input data and the adapter responds to the request with output data. For example, a PLC might send input to a motor to determine the speed of the machine. The motor, acting as an adapter, would respond with an output containing the speed of the machine.

The difference of EtherNet/IP from other protocols, is that the adapters can accept connections from more than one scanner. With other protocols, such as Modbus RTU, slave devices can only connect to a master device. A scanner can connect to any number of adapters and access their data. However, only one scanner can control the outputs of an adapter. Other scanners can monitor the output of an adapter.
② Chapter 6.8.2.6.1.2 "Ethernet protocols and ports for AC500 V3 products" on page 4480
(4) "Pin assignment" on page 381

### 6.4.2.5.2 Configuration EtherNet/IP adapter

## Setting up the communication gateway

Setting up of To set up the communication between the PC and the PLC, e.g., for downloading the compiled communication program, you have to set up the communication parameters. parameters The IP address of your PC must be in the same class as the IP address of the CPU.
The factory setting of the IP address of the CPU is 192.168.0.10.
The IP address of your PC should be 192.168.0.X. Avoid X = 10 in order to prevent an IP conflict with the CPU.

Subnet mask should be 255.255.255.0.

## Changing of the IP address

1. Open Windows control panel. Click "Network and Internet
2. Click "Change adapter settings"
$\Rightarrow$
If using existing network with several devices, please pay attention on given network rules or contact your system administrator.
3. Right-click "Local Area Connection (Ethernet)" and select [Properties].

4. Double-click "Internet Protocol Version 4 (TCP/IPv4)".

5. Enter your desired IP address and subnet mask.

## Setting up the communication gateway

$\checkmark \quad$ CPU and PC are connected with an Ethernet cable．

| Devices |  | $\square \times$ |
| :---: | :---: | :---: |
| ```~省 Project -1) PLC_AC500_V3``` |  | $\cdots$ |
|  |  | Communication Settings．．． |
|  |  | Communication Settings．．． |
|  |  | Source Download．．． |
|  | 䟿 | Copy |
|  | 㫛 | Paste |
|  | \％ | Cut |
|  | $\times$ | Delete |
|  |  | Rename |
|  | 酉 | Properties．．． |
|  |  | Add object |
|  |  | Update objects |
|  | $\square$ | Add Folder．．． |
|  |  | Edit Object |
|  |  | Edit Object with．．． |
|  |  | Compare Objects |
|  |  | Export |
|  |  | Runtime Licensing |
|  |  | Simulation［PLC＿AC500＿V3］ |
|  |  | Reset Origin Device［PLC＿AC500＿V3］ |
|  |  | Check configuration |

1．In the Automation Builder device tree right－click＂PLC＿AC500＿V3＂．
2．Select＂Communication Settings＂．

3. Keep the default value in the IP address of the CPU or type in the current IP address, if differs.

The standard (default) IP address of the port ETH1 is: 192.168.0.10
4. Select $[O K]$ to implement the IP address.

Network scan If you need to scan the network for the CPU or if you have multiple CPUs on the same network.

1. Right-click "PLC_AC500_V3" in the device tree.
2. Select "Communication Settings".

3. Select "..."
$\Rightarrow$ "Pick IP Address for PLC_AC500_V3" opens.

| Pick IP Address for 'PLC_AC500_V3'.... | - |  |
| :--- | :--- | :--- | :--- | :--- |
| Abort scan $\square$ Extended Scan     <br> MAC address Device name Port Serial number Device ID IP Address <br>  Config. IP Address Device Type    <br>       <br>       |  |  |
| Scarning |  |  |

The automatic scan runs.
The results will appear in this field.
4. Select the CPU in the field and select [OK] to implement the needed communications gateway.

Checking the communication settings

If you need to check the communications settings or if you want to see more information about the current selected CPU.


1. Double-click "PLC_AC500_V3" in the device tree.
2. Select "Communication Settings".
$\Rightarrow$ The selected IP address is shown.
3. If the IP address is not visible, enter the IP address manually.
4. To test the connection and/or to see the CPU information press [Enter] or click on the black dot next to the PLC figure.

### 6.4.2.5.3 EtherNet/IP - Bus Cycle Task

General infor- The term "bus" includes all fieldbuses as well as the ${ }^{*} / \mathrm{I} / \mathrm{O}$ bus. Consider that there is no bus mation cycle task for Modbus as Modbus does not provide I/O mapping and is controlled by POUs.

It's recommended to define a dedicated bus cycle task for each fieldbus configured in the project. It's strongly recommended not to use "unspecified" in the ""PLC Settings" to avoid unexpected behavior. The task defined in "PLC Settings" determines the bus cycle task of I/O bus and, depending on the configuration, of the additional fieldbuses (the setting is by default inherited).

Especially in case of EtherCAT, a dedicated bus cycle task should be used which is not shared with other fieldbuses. If [unspecified] is set in the "PLC Settings", the EtherCAT task might be automatically used by other fieldbuses, potentially causing the EtherCAT task processing to fail. This should be avoided by specifying a task different to the EtherCAT task in the "PLC Settings".

As a rule, for each IEC task the used input data is read at the start of each task and the written output data is transferred to the I/O driver at the end of the task. The implementation in the I/O driver is decisive for further transfer of the I/O data. The implementation is therefore responsible for the timeframe and the specific time when the actual transmission occurs on the respective bus system.
Other tasks copy only the I/O data from an internal buffer that is exchanged only with the physical hardware in the bus cycle task.


```
(1) Read inputs from input buffer
(2) IEC task
(3) Write outputs to output buffer
(4) Bus cycle
(5) Input buffer
(6) Output buffer
(7) Copy data to/from bus
(9) Bus cycle task, priority 1, 1 ms
(10) Bus cycle task, priority 5
(11) Bus cycle task, priority 10, interrupted by task 5
```


## Using tasks

The "Task Deployment" provides an overview of used I/O channels, the set bus cycle task, and the usage of channels.

WARNING!
If an output is written in various tasks, then the status is undefined, as this can be overwritten in each case.
When the same inputs are used in various tasks, the input could change when a task is processed. This happens if the task is interrupted by a task with a higher priority and causes the process map to be read again. Solution: At the beginning of the IEC task, copy the input variables to variables and then work only with the local variables in the rest of the code.

Conclusion: Using the same inputs and outputs in several tasks does not make any sense and can lead to unexpected reactions in some cases.

### 6.4.2.5.4 Tabs EtherNet/IP Scanner

Tab EtherNet/IP Scanner - "General"
This tab in the configurator of the EtherNet/IP scanner includes the basic settings. The network interface used by the scanner is configured in the settings of the Ethernet adapter.

Table 598: Options

| Auto-reestablish of connec- <br> tions | The scanner always attempts to automatically reestablish an interrupted <br> connection. <br> For example, if a timeout is detected for UDP I/O messages or the TCP connec- <br> tion to the adapter is interrupted. If the option is activated, then the scanner <br> reconnects to the adapters with the lost connection. |
| :--- | :--- |

## Tab "NetX Configuration"

This tab is only displayed if Ethernet communication modules (e. g. CI50xPNIO) based on the highly integrated netX100 microcontroller is used.

As an alternative to the general EtherNet/IP node in case of a NetX field bus the EtherNet/IP node ( $\operatorname{NetX}$ ) can be added to the device tree. This node provides the additional NetX configuration dialog to select the NetX chip (slot for the card) and the communication channel of this chip. The name of the setting used in the dialog and the possible settings provided by the selection lists are defined by the device description.

Table 599: NetX settings

| Slot | Slot to be used. In case of PCI cards with NetX chip, the slot numbers usually <br> correspond to the PCI card numbers. |
| :--- | :--- |
| NetX Com channel | Channel on the card to be used for the communication. A NetX board may have <br> up to four communication channels for different fieldbuses. |
| Auto-initialize bus | Ve user is asked to determine if the bus should be reinitialized when down- <br> loading or when resetting the application. A new initialization will interrupt the <br> bus and may lead to unwanted behavior of the machine. |

## Tab EtherNet/IP Scanner NetX - "General"

This tab in the configurator of the EtherNet/IP scanner contains the basic settings for communication in the network.

Table 600: Address Settings

| Use static IP address |  |
| :--- | :--- |
| IP address <br> Subnet mask <br> Gateway address | These entries each occupy four bytes and serve to identify the scanner within <br> the network environment |
| Optain IP address automati- <br> cally | This option is available only for the NetX scanner |
| BOOTP | Assignment of the IP address by a server by means of Bootstrap Protocol <br> (BOOTP) |
| DHCP | Automatic configuration of the network settings by the host by means of Dynamic <br> Host Configuration Protocol (DHCP) |

Table 601: Ethernet Settings

| Speed and duplex: | Bit rate of the transmission. In case of "Auto-negotiation", the highest of the <br> available bit rates is selected automatically. |
| :--- | :--- |

Table 602: Options

| Auto-reestablish connections | $\boxed{\square}:$ The scanner always attempts to automatically reestablish an interrupted <br> connection. <br> For example, if a timeout is detected for UDP I/O messages or the TCP connec- <br> tion to the adapter is interrupted. If the option is activated, then the scanner <br> reconnects to the adapters with the lost connection. |
| :--- | :--- |

### 6.4.2.5.5 Tabs EtherNet/IP Adapter

Tab EtherNet/IP Adapter - "General"
The device editor tab shows general information from the device description file. You can modify these values.

Table 603: EDS File

| Vendor name |  |
| :--- | :--- |
| Vendor ID | Provided by the ODVA (Open DeviceNet Vendors Association) |
| Product name |  |
| Product code | Values from the EDS file |
| Major revision |  |
| Minor revision |  |


| Enable ACD | Enables the ACD functionality (Address Conflict Detection) for the EtherNet/IP <br> adapter. <br> Note: The ACD functionality is normally applied by the operating system. There- <br> fore, the user should only use this function very conscientiously. By enabling <br> ACD, complications can result between the controller and the operating system. <br> ACD is a mechanism that EtherNet/IP devices can use to detect and respond to <br> IPv4 address conflicts. The ACD mechanism used in EtherNet/IP complies with <br> the IETF RFC 5227 standard. |
| :--- | :--- |
| Install to Device Repository | If a device with the same device identification has already been installed, then <br> you are asked whether the device should be overwritten. If the device is inserted <br> as a remote adapter below an EtherNet/IP scanner, then you will be asked to <br> automatically update the device. |
| Export EDS File | The EDS file is created and stored on the local computer. In this way, the EDS <br> file can be used in an external configuration file. |

## Tab EtherNet/IP Adapter - "Connections"

The upper part of this tab displays a list of all configured connections. When there is an "Exclusive owner" connection in the EDS file, it is inserted automatically when adding the adapter. The configuration data for these connections can be changed in the lower part of the dialog.
The configuration data is defined in the EDS file. The data is transmitted when the connection to the adapter is established.

| RPI (ms) | Requested Packet Interval: Exchange interval of the input/output data |
| :--- | :--- |
| O --> T Size (Bytes) | Size of the producer data from the scanner to the adapter (Originator --> Target) |
| T --> O Size (Bytes) | Size of the consumer data from the adapter to the scanner (T --> O) |
| Proxy Config Size (Bytes) | Size of proxy configuration data |
| Target Config Size (Bytes) | Size of adapter configuration data |
| Connection Path | Address of the - configuration objects - input objects - output objects |
| Add Connection | Opens the "New Connection" dialog. The parameters for the new connection are <br> determined here. |
| Delete Connection | Deletes the selected connection from the list |
| Edit Connection | Opens the "Edit Connection" dialog. The parameters for the existing connection <br> are modified here. |

Table 604: "Configuration Data"
The table shows the connections with the configuration parameters from the EDS file. The connections are divided into configuration groups.

| Raw data values | If the scaling parameters are defined in the EDS file for the data, then you can show the values as raw data or converted data. The data is displayed without conversion. In the case of enum data types, the index of the enumeration value is shown. $\square$ : The data is displayed with conversion. In the case of enum data types, the enumeration value is shown. |
| :---: | :---: |
| Show parameter groups | If groups are defined in the EDS file, then the parameters that are defined in these groups are displayed in a sorted list. |


| Defaults | Resets to the default values |
| :--- | :--- |
| Value | Double-click to change the value. Depending of the data type, you can specify <br> the value directly in the input field or select from a list box. <br> In the case of bit field data types and deactivated raw data values, a dialog <br> opens for you to choose the individual bits. Only those bits can be selected <br> which fall within defined minimum and maximum values. If bit field data types <br> contain enumerations in the associated EDS file, then only these enumerations <br> are shown with the associated bit positions. <br> If a connection contains a parameterizable connection path in the EDS file, then <br> here you can modify the different parameters of the respective connection. |

## Dialog "New Connection"

Generic connec- The dialog contains the parameters for the new connection. tion (freely con-
figurable)
Table 605: Connection Path Settings

| Automatically generated path | The "Connection Path" is generated automatically from the values for <br> "Configuration assembly", "Consuming assembly", and "Producing assembly". |
| :--- | :--- |
| User-defined path | The "Connection Path" is specified manually in the corresponding input field. |
| Path defined by symbolic name | The path is specified by a symbolic name. <br> Requirement: The device must support symbolic connection paths. |

Table 606: General Parameters

| Connection Path | The connection path is used to address one or more objects in the adapter that <br> provide the input data and receive the output and configuration data. <br> Requirement: The connection path is set to "User-defined path". |
| :--- | :--- |
| Symbolic name | An ANSI string is used instead of the normal connection path. See the manual of <br> the respective EtherNet/IP adapter for permitted ANSI strings. <br> Requirement: The connection path is set to "Path defined by symbolic name". |
| Trigger type | - "Cyclic": Data exchange takes place cyclically at intervals set by the RPI. <br> - "Change of State": Data is exchanged automatically after a change to the <br> scanner outputs or adapter inputs. |
| Transport type "Application": Not implemented |  |

[^42]Table 607: Scanner to Target (Output)

| O-->T size (bytes) | Amount of data from scanner to adapter |
| :--- | :--- |
| Proxy config size (bytes) | Size of proxy configuration data |
| Target config size (bytes) | Size of adapter configuration data |
| Connection type | - "Multicast": A network connection is established. The connection data can be <br> - "Poived by multiple consumers. <br> "Point to Point": A network connection is established. The connection data <br> can be received by exactly one consumer. <br> "Null": A network connection is not established. |
| Connection Priority | Two scanners using different priorities to one adapter can cause conflicts. <br> Adapting the connection priority solves this problem. |
| Fixed/Variable | See the specifications CIP volume 1 and volume 2 for details of the parameters. |
| Transfer format | Requirement: The "Transfer format" is "Heartbeat". <br> Extends the interval at which the scanner sends heartbeat messages to the <br> adapter. This value is multiplied by the "RPl" value. <br> Inhibit time (ms) <br> Heartbeat multiplier <br> sent every 100 ms. |

Table 608: Target to Scanner (Input)

| T-->O size (bytes) | See description for "Scanner to Target". |
| :--- | :--- |
| Connection type |  |
| Connection Priority |  |
| Fixed/Variable |  |
| Transfer format |  |
| Inhibit time (ms) |  |

## Tab EtherNet/IP Adapter - "Assemblies"

The upper part of this tab displays a list of all configured connections. When a connection is selected, the associated assemblies in the lower area of the tab are displayed.

Table 609: Connections
A description of the columns is found on the "Connections" tab.

Table 610: Consuming Assembly O-->T, Producing Assembly T-->0

| Add | Opens the "Select Parameters" dialog. |
| :--- | :--- |
| Delete | Deletes all selected parameters. |
| Move Up <br> Move Down | Moves the selected parameter within the list. The order in the list determines the <br> order in the I/O mapping. |


| Name | You can double-click in the text field to edit the values. |
| :--- | :--- |
| Unit |  |
| Help string |  |$\quad$| Show filling bytes of assem- |
| :--- |
| blies | | helpful in iling bytes of the assemblies are shown in the I/O mapping. This can be |
| :--- |
| the EDS file. |

## Dialog Select <br> Parameters

| Display parameter groups | The dialog displays all parameters from the EDS file by group. <br> $\square$ The dialog displays all parameters from the EDS file in a flat structure. <br> Individual parameters from this list can be selected and added to the list of <br> assemblies by clicking "OK". |
| :--- | :--- |
| Generic parameters | Q You can add generic parameters. Individual values of the parameter can be <br> edited. |

## Tab EtherNet/IP Adapter - "User-Defined Parameters"

The tab displays all additional parameters that are transmitted once only into the bus system during the phase of the starting procedure allotted to this.

## NOTICE!

The user parameters are also transmitted again when a connection is reestablished, for example after the failure of a remote adapter.

| New | Opens the "Select Parameters" dialog for adding a new parameter. The new <br> parameter is inserted before the selected line. |
| :--- | :--- |
| Edit | Opens the "Select Parameters" dialog for changing an existing parameter. |
| Delete | Delete the marked lines. |
| Move Up, Move Down | Changes the order of the user parameters. The order of the parameters in the <br> list corresponds to the order at the initialization. |
| Line | Number of the line |
| Name | Name of the parameter |
| Value | The value of the respective parameter can be changed directly by double- <br> clicking the value. If applicable, a list box opens containing possible values. |
| Unit | Unit |
| Bit Length | Length of the bit |
| Abort on Error | $\boxed{\square}:$ In case of error, the entire transmission of the parameters is aborted. |
| Jump to Line on Error | Line" column. In this way, an entire block can be skipped during the ine initialization, <br> or a return can be defined. <br> Note: A return can lead to an infinite loop if it is never possible to write a certain <br> parameter. |
| Next Line | - |
| Comment | Space for comments |

## Dialog "Select Parameters"

The dialog contains a list of the parameters that are defined in the EDS file. You can define your own generic parameters in addition to the specified parameters.

The values of the selected parameter are displayed in the lower section of the dialog. They can be changed there.

| Show parameter groups | V: Display of the parameters sorted by parameter groups |
| :---: | :---: |
| Generic parameter | : Enables the creation of generic parameters |
| Name | Name of the generic parameter |
| Class | Each object class that can be addressed by the network is identified by an integer value. <br> A class can also be addressed from the class by specifying a special object instance (see "Instance"). |
| Instance | Integer value for the unique identification of an object instance within a class. <br> Example of an object instance: <br> If the value 0 is assigned to the instance, then the class itself is referenced by this special instance. <br> Example - object instance 0: |


| Attribute | Integer value that can belong to a certain class or instance. <br> Example attribute: <br> Object class \#5, Instance \#2, Attribute \#' |
| :--- | :--- |
|  |  |
|  | Different data types (e.g. BOOL, UDINT, WORD, DWORD...) |
| Data type | $8,16,32$ |
| Bit length | Input of different values in different output formats. |
| Value |  |

The values for "Class", "Instance", and "Attribute" are defined in the "CIP Networks Library" (Vol. 1 and 2) or in the manual of the device manufacturer.

## Tab EtherNet/IP Adapter - "General", local

The tab in the device editor of the EtherNet/IP adapter contains the basic settings for network communication.

Table 611: Address Settings

| IP address | Address for the identification of the EtherNet/IP adapter device. |
| :--- | :--- |

Table 612: BOOTP

| Bootstrap Protocol <br> This option is available only for adapters under the NetX scanner. |  |
| :--- | :--- |
| MAC address | Device-specific MAC address of the slave. |
| Save IP address | $\square:$ The address of the slave is saved. The requirement, however, is that the <br> slave supports this function. This option is only available for the CIFX scanner. |

Table 613: Electronic Keying

| Compatibility check | The adapter uses its own keying values to perform a compatibility check of the keying values from the EDS file. All keying values are sent to the device. Then the device decides whether it is compatible with the received values. The adapter uses its own keying values to perform an exact check of the keying values from the EDS file. The user decides which keying information should be checked. <br> - Vendor ID <br> - Device type <br> - Product code <br> - Major revision <br> - Minor revision <br> If the check fails, then an I/O connection is not established to the device and an error message is issued to the status page. |
| :---: | :---: |
| Restore Default Values | For generic devices only. |

### 6.4.2.5.6 Application notes

| Application | How to set up an EtherNet/IP connection and use a PLC as a scanner and set <br> example |
| :--- | :--- |
|  | it up with another PLC as adapter is described in detail in the application example. |
|  | $\underline{C} 500$ V3 EtherNet/IP $-A C 500$ as scanner connected to another $A C 500$ as adapter |

Application How to set up an EtherNet/IP connection and use a PLC as a scanner example and set it up with a drive is described in detail in the application example. AC500 V3 EtherNet/IP - AC500 as scanner connected to a drive.

### 6.4.2.5.7 Command EtherNet/IP - "Scan for Devices"

Function: The command establishes a brief connection to the hardware and determines the devices in the network. Then you can apply the devices found into the device tree of your project.

Call: Menu bar: "Project"; context menu of a device object in the device tree
Requirement: The communication settings to the controller are correct. The gateway and the PLC are started. The device supports the scan function.

The following devices provide the scan function: EtherCAT master, EtherNet/IP Scanner (IEC), Sercos master, CANopen Manager, CANopen Manager SIL2, PROFINET controller and PROFIBUS DP master.


You can perform the device scan immediately if the scan function is permanently implemented in the PLC. When scan function is implemented in a library, you have to log in only one time to download the library to the controller.

The command refers to the master controller selected in the device tree. For example, an already inserted PROFINET IO controller can be selected and the command used to determine the I/O devices and I/O modules assigned to it.

After performing the scan operation, the "Scan Devices" dialog opens and displays the found devices.

Dialog 'Scan
Devices'

Table 614: "Scanned Devices"

| "Device name, Device type, | Data about the scanned device depending on network type. |
| :--- | :--- |
| Address, Station name, etc." | When you change a value in the list of scanned devices, the value is shown <br> in italics. This indicates that the new value has been changed in the editor in <br> CODESYS, but not in the device. When you download the value to the device, it <br> is shown normally. <br> Value that indicate differences between the project and the scanned device are <br> shown in orange. <br> If multiple device descriptions are available for the scanned device, then the <br> name is displayed in bold. The selection of the matching device description is <br> resolved differently for different fieldbuses. For more detailed information, see <br> the corresponding fieldbus chapters. <br> If a device description cannot be found, then the following message is shown: |
|  | "Attention! The device was not found in the repository." Depending on the bus <br> system, additional information is displayed, such as manufacturer number and <br> product number. The device cannot be inserted into the project without the <br> installed device description. |
| "Show differences to project" | V: The table in the dialog also shows additional configured devices (in the <br> device tree of the project). <br> $\square:$ The table shows all scanned devices. The configured devices are not shown. |
| "Scan for Devices" | Starts a new search. |

## NO If $y$ with of be

Table 615: "Configured Devices"
This part of the dialog is visible only when you select the "Show differences to project" option.
Differences between the scanned and configured devices are color-coded. Devices displayed in green are identical on both sides. Devices displayed in red are available only in the view of the scanned or configured devices.

|  | If you have selected a device in both views, then the scanned devices are <br> inserted above the selected configured device. |
| :--- | :--- |
| if you have selected a device in both views, then the scanned devices are |  |
| inserted below the selected configured device. |  |

Scanning an adapter can fail if the PLC is in RUN mode and a connection already exists from the scanning controller to the adapter. Then the scanning causes another connection to be established to the adapter, which interrupts the existing connection in some adapters. Then the scanner restarts the connection to the adapter, which causes the adapter to interrupt the connection to the scanning controller.
For this reason, it makes sense to perform a network scan in STOP mode after a "Reset". If RUN mode cannot be interrupted, then scanning is possible without an projected remote adapter (EtherNet/IP scanners in the device tree only).

For experts When accepting the remote adapter by means of the "Copy to Project" command, the I/O dimensions with which the adapter responded are set for the first "exclusive owner" connection. In order to log all of the detected assembly instances after scanning, the definition IODRVETHERNETIP_PRINT_SCAN_RESULT must be set. By default, it is scanned by the instance ID 100-199. This can be adapted by means of the library parameters ParamScanStartOfInstanceAssem and ParamScanLastofInstanceAssem from the library IoDrvEtherNetIP Library. This might be necessary, for example to scan in another manufacturer-specific range (assembly instance ID ranges).

### 6.4.3 Runtime systems, OPC UA server

### 6.4.3.1 OPC UA server for AC500 V3 products

### 6.4.3.1.1 General

OPC UA server can be added as an object below the Ethernet interfaces ETH1 or ETH2.
The user can access the variable interface of the PLC via a client. At the same time, communication can be protected by means of encryption.
The CODESYS OPC UA server supports the following features:

- Browsing of data types and variables
- Standard read/write services
- Notification for value changes: subscription and monitored item services
- Encrypted communication according to "OPC UA standard (profile: Basic256SHA256)"
- Imaging of the IEC application according to "OPC UA Information Model for IEC 61131-3"
- Supported profile: Micro Embedded Device server Profile
- By default, there is no restriction in the number of sessions, monitored items, and subscriptions. The number depends on the performance of the respective platform.
- Sending of events according to the OPC UA standard.

Application note How to install and configure an OPC V2 or OPC V3 server and how to use it for AC500 PLCs is described in the application note How to use OPC server.

Application The application example How to use OPC server V3-for DA and UA is available to gain a example deeper understanding of the OPC protocol and to configure AC500 V3 accordingly.

### 6.4.3.1.2 Creating a project for OPC UA access

1. Click "File $\rightarrow$ New Project $\rightarrow$ AC500 project" in Automation Builder 2.1 or newer.
2. Choose a PLC - AC500 V3 and click [Add object].
3. Right-click on node ETH1 or ETH2 and "Add object".
4. Choose OPC UA Server in the dialog and click [Add object].
5. Declare some variables of different types in the program.
6. Right-click "Application $\rightarrow$ Add object". Choose "Symbol configuration" and click [Add object].
7. Enable checkbox "Support OPC UA Features" in the dialog "Add symbol configuration".
8. Double-click "Symbol configuration" in the Devices tree to open the editor "Symbol configuration".
9. Click [Build].
$\Rightarrow$ The variables are displayed in a tree structure.
10. Activate the variables that you want to publish to an OPC UA client. Specify the access rights.
11. Download the project to the PLC.

### 6.4.3.1.3 Use node name

1. Double-click node "OPC_UA_Server".
2. Set parameter "Use node name" to TRUE.
3. Double-click node "PLC_AC500_V3 <...>".
4. Click "Device" and "Rename active device..."
5. Enter new device name in the following dialog and click [OK].

### 6.4.3.1.4 Use UaExpert client

The OPC UA client UaExpert is available for download from the Unified Automation website and can be used free of charge (freeware license).
Using this client, you can connect to the AC500 OPC UA server.
The following description refers to this program. Other OPC UA clients work in a similar way.

1. Start the UaExpert program.

2. Click on the "blue cross symbol".
3. Double-click on the "blue cross symbol" in the "Add Server" dialog.
4. Enter URL and click [OK].
$\Rightarrow$ The URL appears in the "Add Server" dialog.
5. Select "Advanced" tab and click [OK].
6. Click [Connect].

7. Expand the project tree in the "Address Space" window.

8. Drag and drop the needed symbols to "Data Access View".

### 6.4.3.1.5 Working with encryption

## Creating a certificate for the OPC UA server

$\checkmark$ Prerequisite: A battery is inserted and the clock is set to actual time.

1. Double-click the Security symbol in the lower right corner of Automation Builder.
2. Select the "Devices" tab.
$\Rightarrow$ The certificate information opens.
3. Select the PLC in the left "Information" view.
$\Rightarrow$ All services of the PLC that require a certificate are displayed in the right Information view.
4. Select the service "OPC UA Server".
5. Click the icon ${ }^{\frac{1}{4} \text { ti }}$ to create a new certificate for the device.
$\Rightarrow$ "Certificate Settings" dialog appears.

6. Define the certificate parameters according the figure above and click [OK].
$\Rightarrow$ The certificate is created on the PLC.

7. Upload the certificate to your PC.
8. Restart the runtime system.

For further information see ${ }^{\mu}$, Chapter 6.3.4.7.3.4 "OPC UA secure" on page 1728.

## Encrypted connection with UaExpert client

1. Start the UaExpert program.

2. Click on the "blue cross symbol".
3. Double-click on the "blue cross symbol" in the "Add Server" dialog.
4. Enter URL and click [OK].
$\Rightarrow$ The URL appears in the "Add Server" dialog.
5. Select "Advanced" tab.

6. Choose option "Basic256ha256" of drop-down list "Security Policy" and "Sign \& Encrypt" of drop-down list "Message Security Mode" and click [OK].

7. Click menu "Settings" and "Manage Certificates"

8. Click [Create new Application Certificate...].
$\Rightarrow$ Dialog "New Application Instance Certificate" opens.

9. Enter the required informations and click [OK].
$\Rightarrow$ Dialog "Manage Certificates" opens

10. Click [Copy Application Certificate To...] your PC.

11. Download the certificate to AC500 via the "Security Screen" view.
12. Click [Connect] in the UaExpert client.

$\Rightarrow$ Dialog "Certificate Validation" opens.

Certificate Validation

## Validating the certificate of server 'OPCUAServer@PM5670' returned an error:

BadCertificateChainIncomplete
Certificate Chain

| Name | Trust Status |
| :---: | :--- |
| 8 OPCUAServer@PM5670 | Trusted |
|  |  |
|  |  |

Certificate Details


Working with a trusted certificate will avoid this error message.
14. Enable checkbox Accept the server certificate temporarily for this session and click [Continue].
$\Rightarrow$ Dialog "Connect Error" opens

15. Click [lgnore]

16. Check settings in dialog "Manage Certificates".

### 6.4.3.1.6 Changing variables via UaExpert client

1. Expand in view "Address Space" "Objects $\rightarrow$ DeviceSet $\rightarrow$ PM5670 $\rightarrow$ Resources $\rightarrow$ Application $\rightarrow$ PLC_PRG".
$\Rightarrow$ The variables of the global variable list are visible.

2. Drag and drop the variables to the Data Access View.
3. Change values in the column Value.

### 6.4.3.1.7 Configuring OPC UA client

## Operating modes

Polling

- Objects will be continuously updated in a defined interval
- Create higher load then Subscription
- Is recommended only for a few Symbols

Pub/Sub Not yet supported

Subscription (recommended mode)

- Updated objects depending on the publishing interval and filters
- Method to reduce load
- Different intervals
- Filter possible (coming in AC500)

| Client defines a group of sym- <br> bols with | Description |
| :--- | :--- |
| Publishing interval | Interval, in which server publish data to client |
| Sampling interval | Interval for sampling and storing data at server and send in <br> each publishing interval |
| Queue size | Array of data to save data if sampling Interval is faster than <br> publishing Interval (At AC500 in the moment only 1) |
| Data change filter | Can be used to reduce traffic from server to client. <br> Criteria: <br> $\bullet \quad$ Change of data, <br> $\bullet \quad$ Change of status <br> $\bullet$ <br> Change of time stamp <br> AC500 is fix configured for change of data and change of <br> status. |



## Using OPC UA with subscription mode

## Recommendations:

- Define only variables you need as symbols
- Do not configure publishing Intervals to short (increase load)
- Use different subscriptions with different publishing intervals in order to decrease load
- Do not use sampling intervals faster then publishing intervals as long as AC500 OPC UA server don't support Queue Size different from 1
- Be careful: Setting „0" at sampling Interval at client will be interpreted in server as „as fast as possible", which is 100 ms at AC500 and create a high load.

Publishing and sampling intervals in UaExpert

1. Right-Click on an Item in "Data Access View" and click "Subscription Settings".

2. Set the recommended values.

Life Time Count: Number of publishing intervals in which client has to send publish requests to the server. After this period without request from client, subscription in server will be deletet.

Max Keep Alive Count: If there are no new data to send, server can skip a publishing interval. After the alive count, server has to send, even if there are no new data.

Click [OK].
3. Right-Click on an Item in "Data Access View" and click "Monitored Item Settings".

4. Set the recommended values.

### 6.4.3.2 Using OPC UA Information Models

The OPC UA information models provide plant information according to the OPC UA standard. This plant data contains both data and metadata, such as data origin, data quality, and data cross-connections. OPC UA-based plants can therefore be connected and analyzed faster.
In plant and mechanical construction, there are several sector-specific OPC UA information models. For example, the Companion Specification EUROMAP77 is an information model for plastics and rubber machinery.

An information model consists of different objects that contain meta-information, runtime information, and user data, such as process variables.

Requirements:

- A CODESYS project is open.
- A controller with integrated OPC UA server functionality has been added to the project.

1. Click "Tools $\rightarrow$ OPC UA Information Model Repository".
$\Rightarrow$ The "OPC UA Information Model" dialog opens.
The information models with the URI http: / /opcfoundation.org/UA/ and http: / /opcfoundation.org/UA/DI / are already installed by default.
2. To install the OPC UA information model that you need in your project, select it and click "Install".
3. In the "Select OPC UA Information Model" dialog, select the information model file *NodeSet $2 . x m 1$ from the file directory and click "Open".
$\Rightarrow$ Now the selected OPC UA information model is displayed in the "OPC UA Information Model Repository".
4. Add a "Communication Manager" object to the application of your CODESYS project.
5. Add an $\triangle$ "OPC UA Information Model" object to the ©iaincomunication Manager".
6. In the "Add OPC UA Information Model" dialog, specify a name for the information model according to IEC 61131-3 and select the desired information model in the "Information Model URI" list box. The list box shows all OPC UA information models from the OPC UA information model repository. Then click "Add".
$\Rightarrow$ The $\bigcirc$ publishing object for which you specified a name in the last step is inserted in the "Devices" view with the selected
The publishing object is opened in the editor.
7. In the next steps, select those objects and data types from the OPC UA information model which you want to use in your CODESYS application. To do this, first open the "Information Model" in the editor.
$\Rightarrow$ The type declarations of the OPC UA information model are displayed on the left side.
8. Add the OPC UA type declarations desired for your CODESYS application by dragging them from the left area to the right area. For all types with the "Modelling Rule" as "Mandatory", a corresponding IEC member is generated for the application later when the IEC declarations are generated. For the "Modelling Rule" as "Optional", in the right area you can activate an IEC member to be generated for this type as well.
9. For OPC UA types with the modelling rule as "Optional Placeholder", in the right area you can drag and drop another corresponding IEC type to insert it.
$\Rightarrow$ Example:

| Data Modes |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  | ¢ Filter by Name, OPC UA Type |  |  |  |
| Types | Element Type | Reference Ty... | Modelling Rule |  | IEC Type | OPC UA Type | Generate member |
| Object Types <br> *- ModileGroupType <br> + MyAddinType <br> * MyInterfaceType <br> - NestedObjectType <br> *- PlaceholderObjectType <br> $=$ TestrootObjectType <br> + Var 1 <br> - Var2 <br> Testlacehoider_N_ <br> + [hasinterfoce] <br> t Menber 1 <br> * Numberinist <br> *. MyAddlin <br> Aundaries <br> - Modiles <br> - Examplemethod <br> * ExampleComplexMethod <br> + ExampleArrayOutputMethod <br> - Data Types <br> *- DataTypeSimple |  |  |  | Name <br> 3.TestRootObjectType | FunctionElock | TestRootObjectiype | [1] |
|  |  |  |  | Var1 | Dint | Int32 | - |
|  |  |  |  | Var2 | NestedObjectType | NestedObjectives | B |
|  |  |  |  | TestPacehoider JIV_ | Placeholder Cbjecttype | PlaceholderObjectivpe | $\square$ |
|  |  |  |  | Audisries | AusilaresFolderType | FolderType | $\square$ |
|  |  |  |  | Modies | ModuesModieGroupType | ModuleGroup Type | $\square$ |
|  |  |  |  | - Examplelvethod | Method | Method | $\square$ |
|  | Int32 | HasComponent | Mandatory | - Inputarguments | None | Argument | $\square$ |
|  | Nestedobiect Tine | HasComponent | Mandatory | * Outputarguments | None | Argument | $\square$ |
|  | PlaceholderObjectType | HasComponent | Optiond Placeholder | - ExampleComplesMethod | Method | Method | 민 |
|  | Myinterfacetrpe |  |  | - InoutArguments | None | Argument | $\square$ |
|  | Int32 | HasComponent | Mendatory | summand1 | Dint | Int32 | g |
|  | UInt16 | HasComponent | Mandatory | summand2 2. | Dint | Int32 | V |
|  | MyAddinType | HasAddln | Mandatory | duttest | DataTypeSimple | DataTypeSimple | $\square$ |
|  | FolderType | HasComponent | Optional | aryTest | ARRAY[I OF DataTypeSimple | DataTypeSimple[] | V |
|  | ModuleGroup Type | HasComponent | Optional | - Outputarguments | None | Argument | $\square$ |
|  | Method | HasComponent | Mandatory | sum | Dint | Int32 | $\square$ |
|  | Method | HasComponent | Mandatory | aryout | ARRAYII OF Dint | Int32] | $\square$ |
|  | Method | HasComponent | Mendatory | - ExampleArrayOutbutMethod | Method | Method | $\square$ |
|  |  |  |  | + IngutArguments | None | Argument | (1) |
|  | Struct |  |  | * Outputioguments | None | Argument | 8 |
|  |  |  |  | * NestedObjectType | FunctionElock | NestedObiect ${ }^{\text {ches }}$ | $\square$ |
|  |  |  |  | * PlocehoidercbjectType | Functionelock | Piaceholderoblectives | $\checkmark$ |

10. When you have mapped all desired OPC UA types to IEC, click the "Generate IEC declarations" button.
$\Rightarrow$ The generated IEC members are visible, for example in the "OPC_UA_Symbols Objects" folder, and can be used in the application. These members are available in the Input Assistant for the implementation of the POUs in your application.
11. When arrays are generated, in the declarations you have to define the upper limits for the arrays because this information is not included in the installed information model file *NodeSet2.xml.
12. In the next steps, you configure the OPC UA types which should be published in an OPC UA Client. To do this, double-click the OPC UA publishing object $\triangle$ in the device view.
13. In the OPC UA publishing editor that just opened, you can generate new instances for the POUs which you have generated in the "Information Model"... Or you can use instances which you have already declared in your application in PRGs and GVLs.
14. To create a new instance, click "Generate New Instance".
15. In the "Generate New Instance" dialog, select the "IEC Type" from the list box and click "OK".
$\Rightarrow$ An instance of the selected IEC type is generated and inserted in the editor. You can double-click the automatically generated name "inst_n" to change it.
16. To search for instances which have already been declared in the application, click "Search for Mapped Instances".
17. In the editor, in the "Root Node" list box, select the directory where the instances should be displayed on the OPC UA Client. The directories that can be selected depend on the OPC UA companion specification.
18. To publish these instances on an OPC UA Client, click "Online $\rightarrow$ Login".
19. When the application has been successfully downloaded to the controller, start the application.
20. Start the OPC UA Client.
$\Rightarrow$ In the OPC UA Client, the instances are displayed which you have selected in the previous steps in the OPC UA publishing editor.

See also - «2y Chapter 6.4.1.21.2.17 "Object 'OPC UA Information Model"" on page 2472

- 若 Chapter 6.4.1.21.3.9.12 "Command 'OPC UA Information Model Repository"" on page 2665
- $\Rightarrow$ Chapter 6.4.3.3 "Mapping of OPC UA Types to IEC Types" on page 2848


### 6.4.3.3 Mapping of OPC UA Types to IEC Types

Mapping of basic data types

Table 616: Basic data types

| OPC UA | IEC | Description |
| :--- | :--- | :--- |
| Basic types |  |  |
| Boolean | BOOL |  |
| Byte | BYTE |  |
| SByte | SINT |  |
| Int16 | INT |  |
| UInt16 | UINT |  |
| INT32 | DINT |  |
| UInt32 | UDINT |  |
| Int64 | LINT |  |


| OPC UA | IEC | Description |
| :---: | :---: | :---: |
| UInt64 | ULINT |  |
| FLOAT | REAL |  |
| Double | LREAL |  |
| Duration | LREAL |  |
| DateTime | LDATEANDTIME |  |
| UtcTime | LTIME |  |
| String | STRING | Simple strings are converted to IEC strings. <br> The length of the IEC string can be changed afterwards and can be chosen without restriction. |
| Special types (examples: NodeId, Bytestring) | Not currently supported | The special data types are ignored when generating the IEC mappings. Members are not generated. |
| Inheritance | Inheritance is allowed for all OPC UA types. For example, new types can also be derived from String and Int32. | Note: In the case of derivations of base types, the base type is used as the basis for the mapping. As a result, the derived OPC UA type is no longer available in IEC. |

## Mapping of object types

All declarations are declared altogether as local variables between VAR and END_VAR. The user can change the declarations as needed in VAR_INPUT and VAR_OUTPUT.

Table 617: Object types

| OPC UA | IEC | Description |
| :--- | :--- | :--- |
| OPC UA <br> object types | Function blocks |  |
|  |  |  |
| Interfaces and <br> add-ins | Function block. The members <br> of the interface are members <br> of the function block. | Example: |


| OPC UA | IEC | Description |
| :---: | :---: | :---: |
| Inheritance | Instead of generating multiple function blocks with "Extends", a flat hierarchy is generated. | Example: |
| Folder | A separate type for each instance of a folder in an OPC UA object type. <br> The user may add the instances on his own by editing the declaration of the IEC POUs. However, function blocks have to be used which originate from an OPC UA companion. <br> All instances of function blocks below the folder are exported. Semantic checks based on NodeSet2.xml are not possible. | Initially, a folder is set as an object type in OPC UA. However, it is not enough to generate a folder type. <br> The user is responsible for adding appropriate elements to the folder. |

## Mapping of structured data types

Table 618: Structured data types

| OPC UA | IEC |
| :--- | :--- |
| Structure | DUT |
| Union | Not currently supported |
| Optional member | Not currently supported |
| Inheritance | Implementation as for the object types |

Mapping of OPC Table 619: OPC UA reference types

## UA reference

 types| OPC UA | Meaning in OPC UA | Mapping in IEC |
| :--- | :--- | :--- |
| Organizes |  | Normally only the derivations <br> of this type are relevant. The <br> exception is when folders are <br> mapped directly to IEC. <br> See n Chapter 6.4.3.3 "Map- <br> ping of OPC UA Types to IEC <br> Types" on page 2848 |
| HasSubtype "Map- |  |  |
|  |  | See „ Chapter 6.4.3.3 "Ma <br> ping of OPC UA Types to IEC <br> Types" on page 2848 |


| OPC UA | Meaning in OPC UA | Mapping in IEC |
| :---: | :---: | :---: |
| HasTypeDefinition |  | - Used to collect all elements of a type when the UA types are compiled according to IEC. <br> - Declarations of instances in IEC are then referenced again by the OPC UA Server with this reference to the type in the information model. <br> - HasTypeDefinition for OPC UA variables is currently supported only for the special case when the variable type has the same members as the data type. Then the single access to the individual variables of the DUT instance is allowed in IEC. |
| HasComponent |  | Variables and objects are mapped in IEC as variables. Therefore, each HasComponent reference becomes an variable declaration in IEC. Methods also become methods of the function block in IEC. <br> The user has to apply the modelling rules in the information model editor before the IEC POUs are generated. Optional members can be selected or deselected and concrete members can be generated for placeholders. |
| HasProperty | In OPC UA, properties have the character of additional meta-information for process data. They can be of static nature, for example engineering units. But they can also change when the server is running. | In IEC, this reference is handled exactly like HasComponent. As a result, variables are also created for this at the function block. An unresolved issue is the handling of properties at variables in UA (by means of the detour of HasTypeDefinition and VariableType). In IEC, the variable of type INT would have to be structured once again and have children. Currently, properties at variables in OPC UA are ignored and cannot be reached from IEC. |

See also • $y^{2}$ Chapter 6.4.3.2 "Using OPC UA Information Models" on page 2846

- $y^{y}$ Chapter 6.4.1.21.2.17 "Object 'OPC UA Information Model'" on page 2472


### 6.4.4 Libraries

### 6.4.4.1 General

Libraries are used for preparing POUs and functions for use in CODESYS applications. In addition to the descriptions presented here in the help, always see the documentation included in the library as well.
For using libraries in your CODESYS project, see the "Managing Libraries" chapter.
To create your own CODESYS libraries, follow the guidelines for library developers.
y Chapter 6.4.1.17 "Using Libraries" on page 2034

### 6.4.4.2 Guidelines for creating libraries

Libraries must be created according to specific rules to avoid compatibility issues.
The main items include the following:

- Select a meaningful library name (required)
- Use templates to ensure consistency (optional)
- Use a familiar and uniform project structure, when possible (optional)
- Register a unique library namespace (required)
- Enter all project information (required)
- Apply the correct method for referencing other libraries correctly (required)
- Design smart external and internal interfaces (required)
- Implement a user-friendly error handling (required)
- Apply the correct method (protection) for deployment (required)
- Apply a consistent naming convention to get clean code (optional)
- When revising an existing library, consider the interface compatibility with previous versions.

Please follow these guidelines when developing libraries in CODESYS: "Library Development Summary". You will find this document as a CHM file (LibDevSummary.chm) in the installation directory of CODESYS, or in the online help.
See also

- $\Rightarrow$ Chapter 6.4.1.17 "Using Libraries" on page 2034


### 6.4.5 CODESYS Visualization

### 6.4.5.1 Introduction

## Everything in one project

In the same CODESYS project, you use CODESYS Visualization to create the suitable user interface for your application. You link the visualization to the application variables and in this way they can animate and display data. When creating a visualization and an application, you use common functions, for example, as library and source code management or find/replace throughout the project.

## Overview of

functionality

- Display variant depending on the target platform

You can execute the same visualization on various target platforms. Possible display variants are CODESYS WebVisu, CODESYS TargetVisu. In addition, there is a display integrated in the development system.

- Visualization editor

In the graphic editor you design the desired user interface from visualization elements. The visualization elements are provided via libraries in a "ToolBox". You drag them into the editor area and adapt them with the help of a property configurator.

- Referenceable visualizations

A visualization can be referenced in other visualizations. This enables the creation of user interfaces with a complex structure. For this purpose CODESYS Visualization also provides predefined visualizations, e.g. for dialogs.

- Simple design change

The simple change of the look \& feel of a visualization is possible in one place by creating a different visualization style.

- Multilingualism

You can conveniently prepare visualization texts in several languages with the help of text lists. You can configure a user input element for switching to a different language in online mode.

- User management

You can set up the visualization's own user management for access control up to individual element level.

- Other useful features

Function block instances of visualizations, array accesses to the visualization, real-time data logging, extendability of the pool of visualization elements, provision of graphic objects via symbol libraries, calls of PLC functions from the visualization, reusability of visualizations by depositing them in libraries.

Table 620: Overview of the objects, editors, repositories, etc. relevant for the visualization in the CODESYS Development System
$\left.\begin{array}{|l|l|}\hline & \\ \hline \text { Visualization } & \begin{array}{l}\text { Object below an application in the device tree or in the POUs } \\ \text { pool that contains a visualization image. A visualization can } \\ \text { reference other visualizations. }\end{array} \\ \hline \begin{array}{l}\text { Visualization editor and addi- } \\ \text { tional views }\end{array} & \begin{array}{l}\text { In this IEC 61131-3-compliant editor you can create the } \\ \text { desired graphical user interfaces, panels, dialogs, etc. from } \\ \text { visualization elements. The editor is made up of the following } \\ \text { components: }\end{array} \\ & \begin{array}{l}\text { - Graphic editor area for arranging the elements } \\ \text { - Interface Editor": for the parameterization of the visualiza- } \\ \text { tion } \\ \text { "Hotkey Configuration": editor for defining keys for online } \\ \text { operation } \\ \text { "Elementlist": overview of all visualization elements used, } \\ \text { editor for the position of the elements on the z-axis }\end{array} \\ \text { The following views are also available: } \\ \text { "ToolBox": view for the provision of visualization elements } \\ \text { - "Properties": view with editor for the configuration of the } \\ \text { element that currently has the focus in the graphic editor }\end{array}\right\}$

|  |  |
| :--- | :--- |
| Visualization Manager | Each application has a visualization manager of its own <br> for its visualizations with various settings such as user man- <br> agement, style, language, input type, etc. The "Visualization <br> Manager" object is suspended in the device tree below the <br> application. |
| Display variant | A visualization can be displayed in online mode in the fol- <br> lowing variants, which are created as objects under the visual- <br> ization manager: <br> - CODESYS TargetVisu (target visualization and remote |
|  | -CODGet visualization on PLC devices) <br> browser) <br> - Visualization integrated in the development system |
| Visualization library | Collection of visualization elements that are provided in the <br> toolbox. |
| Symbol library | Collection of images and graphics that you can use in visuali- <br> zations. When inserting a visualization object you can choose <br> whether the installed system libraries should be available in <br> the project. |
| Visualization Element Reposi- <br> tory | Repository for the management of the visualization profiles <br> and the visualization element libraries. |
| Visualization Styles Repository | Repository for the management of visualization styles. |
| VISU_TASK | This task is automatically present as an object in the task <br> configuration of an application as long as an object for a dis- <br> play variant of the type WebVisu or TargetVisu is also inserted <br> under the Visualization Manager. |

## System overview and mechanism, display variants

The user interfaces created in CODESYS can be used in different display variants, depending on which ones the controller employed supports.
The display variants

- Visualization ("diagnostic visualization") integrated in the CODESYS Development System: The integrated visualization in the development system is ideal for application tests, for service or diagnostic purposes and for the commissioning of a system. As soon as a connection to the controller has been established, the visualization editor switches over and animates the elements displayed. This variant is part of the free CODESYS Development System and can always be used, irrespective of the controller employed.
- CODESYS WebVisu:

This variant means web-based display of the user interface in a standard browser (PCs, tablets, smartphones), enabling remote access, remote monitoring and service and diagnosis of a system via the Internet. A standard web browser communicates by Java Script (optionally with SSL encryption) with the web server in the controller and displays the visualization by means of HTML5. This technology is supported by virtually all browsers and is thus also available on terminal devices with iOS or Android.

- CODESYS TargetVisu:

This variant runs independent of the platform on control systems with an integrated display. Logic application and user interface run on the same device; the user interface is displayed directly on the controller. This variant is suitable for the operation and monitoring of machines and plants. An optional extension of the runtime system is required for the use of CODESYS TargetVisu.

### 6.4.5.2 Tutorial

### 6.4.5.2.1 CODESYS visualization - Getting started

If you are not yet familiar with CODESYS visualization, we recommend you to start with the application example First steps with CODESYS visualization. The example demonstrates the main features of visualization and provides insights into possible use cases.

To use the Edge browser for CODESYS visualization of AC500 PLCs, follow the steps described in the application note Usage in the Edge browser.

Visualization More examples and step by step instructions are available in the Automation Builder. basics

Chapter 3.4.10 "Setting up a visualization" on page 53
y Chapter 3.4.11 "Creation of a visualization" on page 57
*y Chapter 3.4.12 "Enabling a web visualization" on page 65

### 6.4.5.2.2 Show instance names

For complex visualizations, it can be helpful if the instance names are displayed within the visualization. How to show instance names is described in the application example Visualization - Instance names.

### 6.4.5.2.3 Visualizing a Refrigerator Controller

This tutorial demonstrates how to add visualizations to the project and link the elements of the visualization to the variables of the control program.

Preparation This tutorial is based on the sample program RefigeratorControl, which was created in the "Your First Program in CODESYS" chapter. The finished program can also be found in the installation directory of CODESYS, in the "Projects" subfolder.

See also

- Your First CODESYS Program

Creating the vis- The visualization consists of the following three visualization screens:

## ualizations

- Visualization: Control elements and display of the refrigerator
- Diagnosis: History of the set and actual temperature, parameter settings
- Live Visu: Animation with refrigerator

1. Select the "Application" object in the device tree.
2. Click "Project $\rightarrow$ Add Object $\rightarrow$ Visualization".
3. Specify Live_Visu as the name.
4. Create two more visualizations with the names Diagnosis and Visualization.

Structure of the This screen consists of control and display elements that control the refrigerator. visualization Visualization


- (1) Numeric display of the actual temperature
- (2) Pointer to display of the actual temperature
- (3) Numeric display of the set temperature
- (4) Potentiometer for setting the set temperature
- (5) Label for compressor lamp
- (6) Lamp for compressor on
- (7) Label for signal lamp
- (8) Lamp for signal "Close doors"
- (9) Switch for opening and closing the refrigerator door

1. Open the visualization Visualization in the editor.
2. Drag a "Rectangle" visualization element to the editor.

Change the following properties

- "Texts $\rightarrow$ Text": Actual temperature: $\% 2.1 \mathrm{f}{ }^{\circ} \mathrm{C}$
- "Text variables $\rightarrow$ Text variable": Glob_Var.rTempActual

3. Drag a "Meter $180^{\circ}$ " visualization element to the editor.

Change the following properties

- "Value": Glob_Var.rTempActual
- "Scale $\rightarrow$ Scale end": 20
- "Scale $\rightarrow$ Main scale": 5
- "Scale $\rightarrow$ Subscale": 1

4. Drag a "Rectangle" visualization element to the editor.

Change the following properties

- "Texts $\rightarrow$ Text": Temperature presetting: \%.1f ${ }^{\circ} \mathrm{C}$
- "Text variables $\rightarrow$ Text variable": Glob_Var.rTempSet

5. Drag a "Potentiometer" visualization element to the editor.

Change the following properties

- "Variable": Glob_Var.rTempSet
- "Background $\rightarrow$ Background color": "yellow"
- "Pointer $\rightarrow$ Color": "red"
- "Scale $\rightarrow$ Subscale position": "Outward"
- "Scale $\rightarrow$ Scale start": 3
- "Scale $\rightarrow$ Scale end": 13
- "Scale $\rightarrow$ Subscale": 1
- "Scale $\rightarrow$ Main scale": 1
- "Label $\rightarrow$ Unit": ${ }^{\circ} \mathrm{C}$
- "Label $\rightarrow$ Scale format (C syntax)": \%.Of
- "Label $\rightarrow$ Max. text width of labels": 21
- "Label $\rightarrow$ Height of labels":15

6. Drag a "Label" visualization element to the editor.

Change the following properties

- "Texts $\rightarrow$ Text": Cooling compressor

7. Drag a "Lamp" visualization element to the editor. Position it behind the text Cooling compressor.

Change the following properties

- "Variable": Glob_Var.bCompressor

8. Drag a "Label" visualization element to the editor.

Change the following properties

- "Texts $\rightarrow$ Text": Signal (beep)

9. Drag a "Lamp" visualization element to the editor. Position it behind the text "Signal (beep)".
Change the following properties

- "Variable": Glob_Var.bSignal
- "Background $\rightarrow$ Image": Red

10. Drag a "Rectangle" visualization element to the editor.

Change the following properties

- "Texts $\rightarrow$ Text":Door open

11. Drag a "Rocker Switch" visualization element to the editor.

Change the following properties

- "Variable": Glob_Var.rDoorOpen

Structure of the In this screen, you can monitor the temperature curve and optimize the parameters. visualization
Diagnosis


- (1) "Label" elements for the heading
- (2) "Trace" element for displaying the temperature curve
- (3) "Rectangle" elements for displaying the values

1. Open the visualization Diagnosis in the editor.
2. Drag a "Label" visualization element to the editor.

Change the following properties

- "Texts $\rightarrow$ Text": Refrigerator Diagnosis \& Service Menu
- "Text properties $\rightarrow$ Font": Arial, Standard, 18

3. Drag a "Trace" visualization element to the editor.
4. Click the Diagnosis_Trace1 value of the "Trace" property.
$\Rightarrow$ The "Trace Configuration" dialog opens.
5. Select the "MainTask" in "Task".
6. Click the "Add Variable" link.
$\Rightarrow$ A variable is added to the trace. The variable settings are displayed in the dialog.
7. Select Glob_Var.bCompressor for the variable.
8. Add the Glob_Var.rTempSet and Glob_Var.rTempActual variables to the trace. For the other settings, you can use the default values.
9. Click "OK" to exit the dialog.
10. Drag a "Rectangle" visualization element to the editor. Position it on he right next to the trace element.
Change the following properties

- "Texts $\rightarrow$ Text": \%s
- "Text variables $\rightarrow$ Text variable": PLC_PRG.rHysteresis

11. Configure the "OnMouseDown" input configuration of the element. Click "Input configuration $\rightarrow$ OnMouseDown $\rightarrow$ Configure".
$\Rightarrow$ The "Input Configuration" dialog opens.
12. Assign the "Write Variable" command to the action. Accept the default values and click "OK".
13. Drag a "Label" visualization element to the editor. Position it over the first rectangle.

Change the following properties

- "Texts $\rightarrow$ Text": Hysteresis Regulator

14. Adjust the size and position of both elements.
15. Select both of the "Rectangle" and "Label" elements and duplicate them by means of copy and paste.
16. Adjust the labels and variables of the copied elements.

- "Text": Compressor Efficiency, "Text variable": Simulation.P_Cooling
- "Text": Environment Efficiency, "Text variable": Simulation.P_Environment
- "Text": Environ. Efficiency DoorOpen Sim, "Text variable": Simulation.P_EnvironmentDoorOpen
- "Text": Time until Beep for DoorOpen, "Text variable": Glob_Var.timDoorOpenThreshold
- "Text": Time until Beep for Compressor On, "Text variable": Glob_Var.timAlarmThreshold

See also

- ② Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration'" on page 3370

Structure of the This screen includes the representation of a refrigerator. The refrigerator consists of several visualization 'Live Visu' polygon type visualization elements. The doors of the refrigerator are drawn in both the closed and open states. Both doors consist of a group of single elements.


1. Open the Live_Visu visualization in the editor.
2. Select the "Polygon" visualization element in the "Visualization Toolbox" view.
3. Click several times in the editor to create a surface. Right-click to stop adding corner marks.
4. Move the corner marks to the required position so that the element (1) is formed.

5. Select the element.

Change the following properties:

- "Colors $\rightarrow$ Use gradient color":
- "Appearance $\rightarrow$ Line style": "Invisible"

6. Click the "Colors $\rightarrow$ Use gradient color" property.
7. Select the color "Gray" for "Color 1" in the "Gradient Editor" dialog.

8. Create all other elements with the "Polygon" visualization element.
9. Group the elements of the closed doors $(2+3+4)$ and the open doors $(5+6+7+8)$. To do this, press the [Shift] key and click "Visualization $\rightarrow$ Group" to select the elements.
10. Move the elements together so that the completed refrigerator is formed. Position the open doors precisely on the closed doors.
11. Select the "Open doors" group.
12. In the properties, double-click the input field "State variable $\rightarrow$ Invisible".
13. Press [F2] to open the Input Assistant.
14. Select the rDoorOpen variable in the "Variables" category (below "Application $\rightarrow$ Glob_Var").
15. Negate the variable with NOT (--> NOT Glob_Var.rDoorOpen).
$\Rightarrow$ If the rDoorOpen variable is FALSE (door is closed), then the element is invisible. Then the underlying doors are visible.
16. Copy the following elements from the Visualization screen:

- Potentiometer for setting the temperature
- Rectangle for displaying the set temperature
- Door open switch
- Cooling compressor lamp
- Signal (beep) lamp

17. Insert the elements from the clipboard to the Live_Visu visualization screen.
18. Reduce the elements and position them on the refrigerator.


Visualization in When the visualization is complete, test it in simulation mode. online mode (simulation)

1. Click "Online $\rightarrow$ Simulation".
2. Click "Online $\rightarrow$ Login".
$\Rightarrow$ A dialog opens and prompts you to create and download the application.
3. Click "Yes" to confirm the dialog.
4. Click "Debug $\rightarrow$ Start".
5. Open the visualization Live_Visu in the editor.
$\Rightarrow$ The refrigerator is in online mode.
6. Open the doors with the switch and monitor the temperature and the alarms. Change the parameters in the screen Diagnosis and watch the reaction in the temperature curve.

### 6.4.5.2.4 Displaying Array Data in a Histogram

## Setting element properties for the histogram

## Requirements

- A project contains a visualization object and a program.
- A one-dimensional array is declared in the program (example: histogram : ARRAY[1...10] OF INT;).
- In the program, histogram data is assigned to the array (example: within the range from 0 to 50).

1. Double-click the "Visualization" object in the device tree.
2. In the "Visualization Toolbox" view, click "Measurement Controls" and drag the "Histogram" element to the visualization editor.
3. In the visualization editor, click the inserted "Histogram".
$\Rightarrow$ The "Properties" view opens.
4. In the "Properties" view, double-click the "Value" input field in the "Data array" element property. Then click... .
5. In the "Input Assistant" dialog in the "Variables" category of the "PLC (PRG)" program, select the array (example here: histogram : ARRAY[1..10] OF INT; ) and click "OK".
6. To display only part of the array as a histogram, activate the "Use subrange" option and specify the index values of the array in "Start index" and "End index" to define the subrange.
7. Select the "Display type" (example: "Bar").
8. Specify a value between 1 and 100 (example: 30) for the "Relative bar width".
9. Click the histogram in the visualization editor and change the size and position as desired.
$\Rightarrow$ The "Position" property changes its values accordingly.
10. Specify the values for the "Scale" element property. Select the values for "Scale start" and "Scale end" so that the array is displayed completely. For the example: "Scale start" 0, "Scale end" 50.
For the distance between values on the main scale, specify the value 10 , for example, in "Main scale".
11. In the "Label" element property, specify the "Unit" for the display values.
12. Click "Build $\rightarrow$ Generate Code".
13. If the project has been compiled without errors, then click "Online $\rightarrow$ Login" and click "Debug $\rightarrow$ Start" to start the application.
$\Rightarrow$ The histogram is displayed in the visualization as follows:

$\checkmark$ The visualization displays a histogram with bars all the same color (example: green).
Now you want the bars with values less than 30, for example, to be displayed in another color (example: red).
14. Click the element property "Colors $\rightarrow$ Alarm color".
15. Specify the limiting value in "Alarm value" above or below which the bars should be displayed in another color.
16. Select "More" from the list box in "Alarm condition" if all values greater than the "Alarm value" should be displayed in another color. Otherwise, select "Less".
17. Select an "Alarm color" (example: "Red").
18. Click "Build $\rightarrow$ Generate Code".
19. If the project has been compiled without errors, then click "Online $\rightarrow$ Login" and click "Debug $\rightarrow$ Start" to start the application.
$\Rightarrow$ In the example histogram, all bars with values greater than 30 are displayed in red.


See also

- ${ }^{2} \gg$ Chapter 6.4.5.20.1.29 "Visualization Element 'Histogram"" on page 3216


### 6.4.5.2.5 Displaying Array Variables in Tables

A frequently required function of a user interface is the display of data arrays. CODESYS Visualization provides the element "Table" for this.
In the configuration of the element "Table", enter an array variable in the property "Data array". The array components are displayed in the rows and columns of the table.

Subsequent instructions describe an example of how an array of a structure is displayed in a table. As a preparation, create the MYSTRUCT DUT and the declarations in the PLC_PRG program.

```
TYPE MYSTRUCT :
    STRUCT
        iNo : INT;
        bOnStock : BOOL;
        strPartNumber : STRING;
        END_STRUCT
END_TYPE
PROGRAM PLC_PRG
VAR
    arrStruct : ARRAY[0..6] OF MYSTRUCT;
    iSelectedColumn : INT;
END_VAR
```

1. Drag the "Table" visualization element to the visualization editor.
2. Assign the array variable arrStruct to the "Data array" property.
$\Rightarrow$ The structure members are displayed as column headings and the array index as row headings.
3. Change the "Columns $\rightarrow$ Column $\rightarrow$ [0] $\rightarrow$ Column header" property to an informative heading (example: Number).
4. Change the heading of column [1] to in stock and column [2] to Part number. Adjust the column width.
5. Assign a color to the "Selection $\rightarrow$ Selection color" property.
6. Define the "Selection $\rightarrow$ Selection type" property as Row selection.
7. In the "Selection $\rightarrow$ Variable for selected row" property, define the PLC_PRG.iSelectedColumn variable.
$\Rightarrow$ The following display results in online mode:

|  | Number | in Stock | PartNumber |
| :---: | :---: | :---: | :---: |
| 0 | 0 | FALSE |  |
| 1 | 0 | FALSE |  |
| 2 | 0 | FALSE |  |
| 3 | 0 | FALSE |  |

See also

- ${ }^{\text { }} \boldsymbol{y}$ Chapter 6.4.5.20.1.13 "Visualization Element 'Table"" on page 3106


### 6.4.5.2.6 Displaying Web Contents

Displaying websites in a visualization
$\checkmark$ Requirement: A visualization open in a CODESYS project. The "Visualization Toolbox" and "Properties" views of the visualization are also open.

1. Drag the "Web Browser" element from "Special Controls" to the visualization editor.
2. Select the element in the editor.
$\Rightarrow$ In the "Properties" view, the element properties are listed for the "Web Browser" element.
3. In the "Position" property, specify the size (in pixels) for the "Width" and the "Height" (example: 600).
4. In "Control variables $\rightarrow$ URL", specify the URL for the website (example: ' http: / / de.wikipedia.org'). You can also specify a variable here (STRING or WSTRING) where the URL is assigned in the project.
5. In "Control variables $\rightarrow$ Display", specify a Boolean variable (example: bSetURL).
$\Rightarrow$ If the variable bSetURL has the value TRUE, then the website 'http:// en.wikipedia.org' is displayed at runtime.

## Configuring the

 buttons for forward and back navigation of the website$\checkmark$ Requirement: The "Web Browser" element of your visualization is configured as described above.

1. In a POU, declare both Boolean variables bGoBack and bGoForward.
2. In the visualization editor, click the "Web browser".
3. For the property "Control variables $\rightarrow$ Back", select the variable bGoBack from the Input Assistant. For the property "Control variables $\rightarrow$ Forward", select the variable bGoForward.
4. In "General Controls", add the "Button" element to your visualization two times.
5. Click a "Button" in the visualization editor and drag the "Button" to the required position (for example above the "Web Browser" element).
6. In the property "Texts $\rightarrow$ Text", specify the character $>$. In "Text properties $\rightarrow$ Font", select a font from the Input Assistant (example: Arial, Bold, 14).
7. Configure the property "Input configuration $\rightarrow$ OnMouseClick" so that the variable bGoForward switches.
8. Configure the second button for back navigation in the same way as described in Steps 5 to 7.
$\Rightarrow$ If the variable bSetURL has the value TRUE, then the website 'http:// de.wikipedia.org' is displayed with the forward and back buttons. When you click the buttons, navigation to the previous and next websites is successful.
(3) Chapter 6.4.5.20.1.38 "Visualization Element 'Web Browser'" on page 3262
"Input action 'Toggle Variable"' on page 3379

### 6.4.5.2.7 Using Client Animation

The example shows a visualization with 3 screens. A menu controls the navigation of the screens. The menu is hidden until it moves into view by means of a hamburger button. During the movement, the transparency of the menu is changed. After the screen is selected, the menu moves back out of view. The animation is computed entirely on the target system. The CODESYS visualization only defines the target values (positions, transparency).

## 1. Preparation

1. Create a new standard project with the CODESYS Control Win V3 controller.
2. Add a "Visualization" object below the "Application". Choose the name Visu_Main.
3. Open the Visualization Manager in the editor and select the "Support client animations and overlay of native elements" option.
4. Creating the PLC_PRG program

The program checks whether the menu button has been pressed. If the menu bar is not visible (position -300), then the position is moved to the visible area (0). If the menu bar is already visible (position 0 ), then the position is moved to the hidden area.

1. Open the "PLC_PRG" program in the editor.
2. Input the following code into the declaration editor:
```
PROGRAM PLC_PRG
VAR
    iSelection : INT; // to switch the
referenced visualization page
    xVisible: BOOL; // auxiliary variable to
toggle the menu bar
    iMenuPos : INT := -300; // position of the menu bar
    xToggle: BOOL; // button variable to
toggle the menu bar
END_VAR
```

3. Input the following code into the implementation:
```
IF xToggle THEN
    xToggle := FALSE;
    IF xVisible THEN
        xVisible := FALSE;
        iMenuPos := -300;
    ELSE
        xVisible := TRUE;
        iMenuPos := 0;
    END_IF
END
    IF
``` menu bar
3. Creating the The menu bar has 3 menu items. A visualization screen is displayed by clicking the corresponding menu item.
1. Insert a "Visu_Menu" visualization below the application.
2. Open the object properties. In the "Visualization" tab, set the "Visualization size" to a "Width" of 300 and a "Height" of 180.
3. Open the visualization in the editor.
4. Select the "Advanced" option in the "Properties" view.
5. In the upper left corner, add a button with a "Width" of 300 and a "Height" of 60.
6. Label the button as "Visu 1". Set the font size to 24.
7. Open the "Input configuration \(\rightarrow\) OnMouseClick" property.
8. Select the "Execute ST code" action.
9. Input the following ST code:
```

PLC_PRG.iSelection := 0;
PLC_PRG.xToggle := TRUE;

```
10. Set the "Button state variable \(\rightarrow\) Digital variable" property to PLC_PRG.iSelection \(=0\)
11. Add two more buttons named "Visu 2" and "Visu 3".
12. Edit the button properties of "Visu2" (PLC_PRG.iSelection = 1) and "Visu3" (PLC_PRG.iSelection = 2).
\(\Rightarrow\) Result:
Visu1
Visu2
Visu3

\section*{4. Creating more visualization screens}
1. Insert the "Visu1" visualization below the application.
2. Open the object properties. In the "Visualization" tab, set the "Visualization size" to a "Width" of 800 and a "Height" of 600.
3. Change the background color of the screen (for example, light gray).
4. Insert a "Label" object into the visualization screen and name the element (example: "Visu 1").
5. Insert two more visualizations "Visu2" and "Visu3" below the application. Edit the properties in the same way as for "Visu1".
5. Creating the main visualization screen

On this screen, you can see the menu bar and a button to show or hide the menu bar. The different visualization screens are navigated in a "Frame" visualization element.
1. Open the properties of the "Visu_Main" visualization. In the "Visualization" tab, set the "Visualization size" to a "Width" of 800 and a "Height" of 600.
2. Open the visualization in the editor.
3. Insert a "Frame" element into the visualization.
\(\Rightarrow\) The "Frame Configuration" dialog opens.
4. Add the "Visu1" (Index 0), "Visu2" (Index 1), and "Visu3" (Index 2) visualizations.
5. Set the property values of "Position" as follows: " \(X\) " \(=0, " Y\) " \(=0\), "Width" \(=800\), and "Height" = 600 .
6. Set the property value of "Switch frame variable \(\rightarrow\) Variable" to PLC_PRG.iSelection.
7. Insert a "Button" element into the visualization.
8. Set the property values of "Position" as follows: " \(X\) " \(=0, " Y\) " \(=0\), "Width" \(=800\), and "Height" = 600.
9. Set the property value of "Texts \(\rightarrow\) Text" to \(=\).
10. Set the property value of "Text properties \(\rightarrow\) Font" to Arial; 36.
11. Open the "Input configuration \(\rightarrow\) OnMouseClick" property.
12. Select the "Execute ST code" action.
13. Input the following ST code:

PLC_PRG.xToggle := TRUE;
14. Set the property value of "Button state variable \(\rightarrow\) Digital variable" to PLC_PRG.xVisible.
15. Insert the "Visu_Menu" visualization element from the "Current Project" category into the visualization.
16. Set the property values of "Position" as follows: "X" = 0, "Y" \(=0\), "Width" \(=300\), and "Height" \(=180\).
17. Set the property value "Absolute movement \(\rightarrow\) Movement \(\rightarrow \boldsymbol{X}\) " to PLC_PRG. iMenuPos.
18. Set the property value of "State variables \(\rightarrow\) Invisible" to not (PLC_PRG.xVisible).
19. Set the property value of "Animation duration" to 2000.
\(\Rightarrow\) Result:


See also
- \({ }^{*}>\) Chapter 6.4.5.20.1.6 "Visualization Element 'Frame'" on page 3053
6. Downloading the project to the controller and starting the WebVisu
1. Build the project and download it to the PLC.
2. Start the project.
3. In the browser, connect to the visualization (http: / /localhost:8080).
\(\Rightarrow\) The WebVisu connects to the controller and the visualization opens.
4. In the visualization, click the menu button.
\(\Rightarrow\) The menu moves into view.
5. Select a menu item.
\(\Rightarrow\) The visualization screen is selected and the menu moves back out of view.

\subsection*{6.4.5.3 Preparing CODESYS and projects}

The following provides details of the presets that exist for visualizations and the steps that are necessary for creating a visualization in a project.

Presets When you create a visualization in a project, you should know that the following presets apply:
\begin{tabular}{|c|c|c|}
\hline Scope & Location & Setting \\
\hline Throughout CODESYS & \begin{tabular}{l}
"Tools \(\rightarrow\) Options" \\
Categories "Visualization" and "Visualization styles"
\end{tabular} & \begin{tabular}{l}
- Visualization editor: display, handling \\
- Paths of the basic text and image files \\
- Visualization styles
\end{tabular} \\
\hline Throughout the project & \begin{tabular}{l}
"Project \\
\(\rightarrow\) Project settings" \\
Categories "Visualization" and "Visualization profiles"
\end{tabular} & \begin{tabular}{l}
- "Properties handling" for the visualization elements \\
- Paths of the basic text and image files \\
- Symbol libraries with ready content \\
- Visualization profile
\end{tabular} \\
\hline Throughout the application & "Visualization Manager": & \begin{tabular}{l}
- Unicode, CurrentVisu variable, multitouch, semitransparency, memory size, data transmission, number of clients \\
- Visualization styles \\
- Language setting, language-specific font \\
- Default keyboard configuration \\
- Visualizations and visualization references \\
- Font for each language \\
- User management
\end{tabular} \\
\hline Single visualization & \begin{tabular}{l}
"Properties" of the visualization object \\
Category "Visualization"
\end{tabular} & \begin{tabular}{l}
- Purpose and scope of use \\
- Size definition
\end{tabular} \\
\hline Display variant of a single visualization & Editor of the WebVisu or TargetVisu object & \begin{tabular}{l}
- Start visualization, refresh rate, buffer size, html file name \\
- Scaling options \\
- Display options \\
- Default text input
\end{tabular} \\
\hline
\end{tabular}

Project-specific updates of the visualization profile, the visualization styles, and the visualization symbol libraries are possible in "Project \(\rightarrow\) Project environment" of the respective tabs.
Customization of the visualization menu is performed in "Tools \(\rightarrow\) Customize".

\section*{See also}
- \(y^{*}\) Chapter 6.4.5.21.3.9 "Dialog Box 'Options' - 'Visualization'" on page 3384
- \(乡\) Chapter 6.4.5.21.3.7 "Dialog 'Options' - 'Visualization Styles'" on page 3382
- \({ }^{\star}\) Chapter 6.4.5.21.3.13 "Dialog 'Project Settings' - 'Visualization"' on page 3387
- \({ }^{4}\) Chapter 6.4.5.21.3.14 "Dialog 'Project Settings' - 'Visualization Profile'" on page 3388
- \({ }^{\text {• }}\) Chapter 6.4.5.21.3.10 "Dialog 'Project Environment' - 'Visualization Profile"' on page 3385
- \(\boldsymbol{y y}^{\boldsymbol{y}}\) Chapter 6.4.5.21.3.11 "Dialog 'Project Environment' - 'Visualization Styles"' on page 3386
- Chapter 6.4.5.21.3.12 "Dialog 'Project Environment' - 'Visualization Symbols'" on page 3386
- Chapter 6.4.5.21.3.15 "Dialog 'Properties' of Visualization Objects" on page 3388

Creating visuali- For each visualization, you insert a "Visualization" object into your project like any other object. zation objects in This also applies to visualizations that should be used later only within other visualizations. You the project can insert the new visualization object directly below an application, or below the root node of the "Devices" view (for availability throughout the entire project).
The required base libraries and other objects, such as the Visualization Manager, are inserted automatically. When you insert the visualization object below an application, the subordinate objects for the display variants supported by the device are also displayed.
Every visualization object can be edited separately in the visualization editor.
The following steps describe a simply example for creating an object for an application-specific visualization.
Requirement: A project is open. An application is created in the device tree.
1. Select the application in the device tree. Click "Add object \(\boldsymbol{\rightarrow}\) Visualization" in the context menu.
\(\Rightarrow\) The "Add visualization" dialog box opens. In the "Symbol libraries" table, there is at least the standard entry "VisuSymbols Vx.x.x. (System)", and possibly other installed symbol libraries.
2. Accept the default name Visualization. Activate the "VisuSymbols"option. Then the visualization symbols (graphical objects) are contained in the library in the visualization project. Click "Add" to close the dialog box.
\(\Rightarrow\) In the device tree, the "Visualization manager" and "Visualization" objects are inserted below the application. Depending on the device in use, the "TargetVisu" and/or "WebVisu" objects are also created below the visualization manager.
If a "TargetVisu" object or "WebVisu" object is created, then a "VISU_TASK" object is also created below the task configuration with an implicit program call.
The required visualization libraries are added automatically in the "Library Manager" of the application.
The visualization editor opens with the "Visualization" editor window and the "ToolBox" and "Properties" views.
In the "ToolBox" view, there is a "Symbols" button for viewing the symbols from the library VisuSymbols.library.
3. Now you can create the required visualization in the visualization editor.
4. Note: You can create structured visualizations by using a frame element to reference one visualization in another visualization. Dialog visualizations are a special option for this. In this case, the input configuration of a visualization element is used for referencing.


For creating an application-dependent visualization, insert the visualization object directly below the root node of the device tree. This corresponds to insertion in the "POUs" view. In this case, the visualization manager is not created with objects for the display variants.
(4) Chapter 6.4.5.21.3.7 "Dialog 'Options' - 'Visualization Styles'" on page 3382
«2 Chapter 6.4.5.21.3.13 "Dialog 'Project Settings' - 'Visualization'" on page 3387
* \({ }^{4}\) Chapter 6.4.5.21.3.15 "Dialog 'Properties' of Visualization Objects" on page 3388

\subsection*{6.4.5.4 Limitation of the number of usable web pages on AC500 V3 PLCs}

Automation Builder will get all the available visualizations in the project and count those reachable from the visualization client objects (WebVisu and RemoteTargetVisualization).
In case the predefined number of visualizations is exceeded an error is shown in the message window, preventing the user from compiling the project.
The error will be shown under "Build" category when the user executes the build command. The PLC program won't download to the PLC until this error is solved (like with any other build errors). In the image below, there are 5 visualizations being used (3 of them added directly into the Automation Builder project and the other 2 referenced from a library that was added to the project).


The error will look like this when build command is executed:
```

Messages - Total }3\mathrm{ error(s), 2 warning(s), 2 message(s)

```

```

Description
----- Build started: Application: PLC_AC500_V3.Application ------
Typify code...
(%)There are 5 used visualizations. The limit of used visualizations is 4.
Compile complete --1 errors, 1 warnings

```

If the visualizations are in the project but not being referenced (e.g. not reachable from the "Start Visualization" in the WebVisu) they are not taken into account for this limitation. If the error condition is solved, the error will disappear when the user executes the build command again.

\subsection*{6.4.5.5 Designing a visualization with elements}

\subsection*{6.4.5.5.1 General}

The visualization editor provides the visualization elements for designing a user interface in the "Visualization Toolbox" view.

Drag the desired element into the editor view and adapt it in the "Properties" view: purely visual design, labeling, display of data, reaction to user inputs, possibility to input values, etc.
Static or dynamic configuration of the properties is possible. This means the assignment of fixed values or the assignment of application variables. A dynamic configuration allows for an animation which is executed at runtime.
See also
- \(\Rightarrow\) Chapter 6.4.5.21.4.1.1 "Visualization Editor" on page 3393
- « Chapter 6.4.5.10 "Animating visualization elements" on page 2912

\subsection*{6.4.5.5.2 Select Element}

The "Visualization Toolbox" view provides the following elements for selection:
- All visualization elements which the set visualization profile defines.
- Image elements for all images of the project from the integrated libraries or symbol libraries.
- Frame elements for all visualizations of the project or from the libraries.


The elements are combined into specific categories, each of which has its own button in the "Visualization Toolbox" view. You can create new categories and assign its elements.
The elements of the categories are displayed in the "Visualization Toolbox" view as preview images. It is also possible to search for an element name.
Simply drag the preview image of the element to the desired position in the editor window. Then the configurable properties of the element are displayed automatically in the "Properties" view of the visualization editor.
(4) Chapter 6.4.5.21.4.1.1 "Visualization Editor" on page 3393

Chapter 6.4.5.21.4.1.2 "View 'Visualization Toolbox"' on page 3394
« Chapter 6.4.5.20.1.5 "Visualization Element 'Image"" on page 3038
« Chapter 6.4.5.20.1.6 "Visualization Element 'Frame'" on page 3053

Create or Requirement: The visualization editor is open. remove new element category
1. In the "Visualization Toolbox" view, click the 国 button.
\(\Rightarrow\) the "Configure Categories and Items" dialog opens.
2. In the dialog, click the + symbol to open the "Add Category" dialog. Note: Click the symbol or press the [Del] key to delete the definition of a category.
3. In the "Name" field, specify a name (example: tagA) and click "OK" to close the dialog.
\(\Rightarrow\) In the "Configure Categories and Items" dialog, the new custom category tagA is inserted below in the tree view. It is provided with the es symbol.
4. Click the "Enable" option for the new category, and click "OK" to close the dialog.
\(\Rightarrow\) CODESYS adds a "tagA" button in the "Visualization Toolbox" view. When you click the button, all elements that are assigned to this category are displayed.
* \({ }^{2}\) Chapter 6.4.5.21.3.4 "Dialog 'Configure Categories and Items"' on page 3368

Assigning a vis- Requirement: The visualization editor is open. You have already created a custom category ualization ele- tagA. A button labeled tagA is visible in the "Visualization Toolbox" view. ment of an element category
1. In the "Visualization Toolbox" view, right-click an element to open its context menu.
\(\Rightarrow\) A context menu opens. It contains the "Add Item to Category 'tagA"" and "Add to Categories" commands.
2. Click "Add to Category 'tagA"" and click "OK" to close the dialog.
3. Click the "tagA" button.
\(\Rightarrow\) All elements are displayed which are assigned to this category, below it also the currently assigned element.
« Chapter 6.4.5.21.4.1.2 "View 'Visualization Toolbox'" on page 3394

\subsection*{6.4.5.5.3 Positioning the Element, Adapting Size and Layer}

A visualization is a raster image in pixels. The pixel position is specified in \(X / Y\)-coordinates. The origin \((0,0)\) is located at the upper left corner of the window. The positive \(X\)-values run to the right, and the positive Y -values run downwards. The position of an element on the Z-axis of the visualization is controlled by the position in the element list (see below).

Configuring the The size and position of an element are specified as pixel coordinates in the "Properties" view. size and posiThese settings are displayed graphically in the editor view at the same time.
When you drag a visualization element from the "Visualization Toolbox" view to the editor view, it is shown as selected, as in the following example of a rectangle element:


The possible positions depend on the set grid. You can change its settings CODESYS options. Commands in the context menu are available for alignment and grouping.

Now you can move or resize the element directly in the editor. As an alternative, you configure the "Position" property in the properties editor, which opens automatically for the selected element. See the description for this, for example in the help page for the "Button" element. The changes are also updated in the other editor.

Changing the element size and position in the editor
1. Focus the element so that the shape of the mouse pointer indicates movement (example:

解)
2. Drag the element to any position.
\(\Rightarrow\) The position of the element is also updated in the properties "Position \(\rightarrow X\) " and "Position \(\rightarrow Y\) ".
3. Focus on a blue box.
\(\Rightarrow\) The shape of the mouse pointer is a double arrow that indicates the direction you can drag the box in order to resize the box: 野.
4. Drag the blue box to resize the element.
\(\Rightarrow\) The position of the element is also updated in the properties "Position \(\rightarrow X\) " and "Position \(\rightarrow Y\) ".

Moreover, you can rotate the "Rectangle", "Line", "Polygon", and "Pie" elements.

Static rotation of rectangle, line, polygon, pie, or image
1. Select the element for static rotation. Example: Rectangle
\(\Rightarrow\) The rectangle is displayed with a handle next to the movable position boxes.

(1) Handle
2. Drag the mouse pointer over the handle.
\(\Rightarrow\) The cursor is displayed as a rotating arrow \(\boldsymbol{A}\).
3. Rotate the element to any position.

\(\Rightarrow\) In the property "Position \(\rightarrow\) Angle", the set angle is displayed in degrees.

๕ Chapter 6.4.5.10.2 "Configuring rotations and offsets" on page 2912

Moving the visu- Each visualization element is in its own layer of the visualization (Z-axis). It can be hidden alization element forward and back
by other elements in the foreground and hide other elements in the background. The order of layers is visible on the "Element List" tab above the editor view. The order of elements from front to back specifies the order of visualization layers from back to front.
Use the commands from the "Order" context menu to move a selected element.
Example of an element list (1):

«y Chapter 6.4.5.21.4.1.1 "Visualization Editor" on page 3393
๕ Chapter 6.4.5.20.1.11 "Visualization Element 'Button'" on page 3089
(4) Chapter 6.4.5.21.2.3 "Command 'Visualization Element List"' on page 3342
y Chapter 6.4.5.21.2.5 "Command 'Order'" on page 3344
\& Chapter 6.4.5.21.2.6 "Command 'Alignment"' on page 3344
(y) Chapter 6.4.5.21.2.7 "Command 'Group'" on page 3347

\subsection*{6.4.5.5.4 Assigning a color}

You configure the color of a visualization element either statically by means of the "Color" property, or dynamically by assigning an application variable by means of the "Color variables" property. Depending on the element, color assignments are also available in other properties. For example, for the font color, this is provided in the "Text" property of a labeled element.
For the static assignment of a color value, you can always use the color dialog in the properties editor, which provides color palettes to choose from.
You can specify the color as a style color. Style colors are color names for color definitions from the actively applied style. When configuring an corresponding property, you are provided with a list of available style colors. We recommend that you use style colors because then you can change colors centrally by means of a style selection or a style customization. You can also open the "Color" dialog to select a value from color palettes.

In addition, you can define the fill color of an element as a "Gradient". Then the color changes linearly, radially, or axially from the initial color to the final color. You configure the "Gradient setting" in the "Gradient Editor" dialog.

Designing a visualization element with a style color or a fixed color value

\section*{NOTICE!}

A color assignment with style color allows for easy global color changes.

Requirement: The visualization editor is open.
1. Insert some Rectangle elements.
2. Select an element.
\(\Rightarrow\) The "Properties" view is active.
3. Click in the "Colors \(\rightarrow\) Normal state \(\boldsymbol{\rightarrow}\) Fill color" property.
\(\Rightarrow\) A list box and the... button appear.
4. Assign a style color to the rectangle. For example, select "Elementfillcolor" from the list box.
5. Define the degree of transparency in the "Colors \(\boldsymbol{\rightarrow}\) Normal state \(\boldsymbol{\rightarrow}\) Fill color \(\rightarrow\) Transparency" property. Use the slider to select the value " 136 ".
6. Select another rectangle. Click in the "Colors \(\boldsymbol{\rightarrow}\) Normal state \(\rightarrow\) Fill color" property. \(\Rightarrow\) A list box and the \(\ldots\) button appear.
7. Assign a fixed color value to the rectangle. Click \(\square .\). to do this.
\(\Rightarrow\) The "Color" dialog opens.
8. Select a standard color or "Define Custom Colors" to fine-tune your selection. Then click "OK".
\(\Rightarrow\) The color is set as a fixed value. The color is displayed as a small rectangle. The RGB values are also indicated next to it.
9. Click in the "Colors \(\rightarrow\) Normal state \(\boldsymbol{\rightarrow}\) Fill color \(\boldsymbol{\rightarrow}\) Transparency" property.
10. Use the slider to select the value " 136 ".
\(\Rightarrow\) The color is semitransparent.
*) Chapter 6.4.5.19 "Applying Visualization Styles" on page 2979

Designing a visualization element with a color gradient
\(\square\) Requirement: The visualization editor is open.
1. Drag a "Rectangle" element to the visualization.
2. Select the "Colors \(\boldsymbol{\rightarrow}\) Use gradient color" property.
3. Click in the "Colors \(\rightarrow\) Gradient setting" property.
\(\Rightarrow\) The "Gradient Editor" dialog opens.
4. Define the color gradient for the element:
- "Gradient type": "Radial"
- "Standard radial": "Center"
\(\Rightarrow\) The fill color of the element changes radially from white to black.

ⓨ Chapter 6.4.5.21.3.5 "Dialog 'Gradient Editor'" on page 3369

\section*{Configuring a visualization element for color animation}

The "Color variables" property, which certain elements may have, is used for the color animation of the element. If you assign a variable there, then you can program color changes in the application code or configure a user input that results in a color change.
You can see an example in the "Animating Visualization Elements" chapter.
«y Chapter 6.4.5.10 "Animating visualization elements" on page 2912
苂 Chapter 6.4.5.21.4.2 "Object 'Visualization manager'" on page 3398
\({ }^{\star}\) Chapter 6.4.5.19 "Applying Visualization Styles" on page 2979
\(\boldsymbol{\wedge}\) Chapter 6.4.5.10.4 "Animating a color display" on page 2914

\subsection*{6.4.5.5.5 Using texts}

You can get displayed text in an element by assigning a string in the element property "Texts \(\rightarrow\) Text". For example all base elements have this property. Also, you can get displayed a text as a tooltip (element property "Texts \(\rightarrow\) Tooltip"). Texts assigned in this way are static. They are managed in the object "GlobalTextList" in view "POUs" and they cannot be modified during runtime, neither programmatically nor via an user input.
However, you can extend a static text by (exactly) 1 placeholder containing a formatting specification, in order to output the content of a variable at this place. At runtime the current value of the variable, which you have assigned to the element via property "Textvariable", will be output.

On the possible formatting specifications please see: \(\#\) Chapter 6.4.5.20.2
"Placeholders with Format Definition in the Output Text" on page 3329

By dynamic configuration you can animate the optical representation of the text.
You can localize the static texts, if you have set up multilingualism in your project.
See some examples for the text configuration of visualization elements in the following chapters.
丞 Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329
Chapter 6.4.5.8 "Setting Up Multiple Languages" on page 2906

\section*{Labeling an image element with a static text}

Precondition: A project containing a visualization is opened. You have an image file representing a stop symbol.
1. Below the Application object insert an object "Image Pool" named ImagePool_A.
2. In the image pool ImagePool_A add your stop symbol image file with ID Stop.

3. Open the visualization and from the ToolBox draw an element "Image" into the editor. \(\Rightarrow\) The input assistant opens. In tab Category you see the image poolImagePool_A.
4. Select the image Stop and close the dialog with "OK".
5. Configure the property "Text" of the image: ImagePool_A, Stop
6. Configure the property "Text properties \(\rightarrow\) Horizontal alignment": Left.
7. Configure the property "Text properties \(\rightarrow\) Vertical alignment": Bottom.

\section*{STOP}

ImagePool_A, Stop

Text output: Element outputs the result of ST code which is executed on a mouse-click

Precondition: A project containing a visualization is opened.
1. Open the visualization and insert a "Button" element.
\(\Rightarrow\) The "Properties" view opens for the new element.
2. Configure property "Text": Number of clicks: \%I
\(\Rightarrow\) The string contains the placeholder \(\% I\).
3. In POU PLC_PRG of the application declare a type-conform variable: iClicks : INT;
4. Configure the property "Text variable" of the button element with PLC_PRG.iClicks.
\(\Rightarrow\) At runtime the variable value will be output instead of the placeholder.
5. Below property "Inputconfiguration", in the cell containing the input event OnMouseClick, click on "Configure".
6. From the list of possible actions choose Execute ST-Code.
7. Enter the code for the action in the editor "Execute ST-Code":

PLC_PRG.iClicks := PLC_PRG.iClicks + 1;
8. Close the dialog with "OK".
\(\Rightarrow\) The user input is configured.
9. Build, download and start the application.
\(\Rightarrow\) The application is running. The visualization opens. The element is labeled and the number of clicks will be output. If you as user click on the button, the number will be increased.


Text output: Using the "Text field" element you can produce a dynamic text output. The text output can be Dynamic output using a textlist effected via an user input or via the application program.
\(\checkmark\) Precondition: A project containing a visualization is opened.
1. Open the visualization and insert a "Text field" element.
\(\Rightarrow\) The "Properties" view shows the configuration of the element.
2. Below the application add a Text List with the following entries: Textlist_A.
\begin{tabular}{|c|c|c|}
\hline \multicolumn{2}{|r|}{1 TextList_A \(\times\)} & - \\
\hline ID & Standard & \\
\hline 0 & None & \\
\hline 1 & Dynamic: F & \\
\hline 2 & Dynamic: F & \\
\hline 3 & Dynamic: F & \\
\hline 4 & Information & \\
\hline 5 & Information & \\
\hline 6 & Information & - \\
\hline & III & , \\
\hline
\end{tabular}
3. In POU PLC_PRG of the application declare the text variable:strTextID : STRING := '0';
4. Also declare the variable strTooltipID : STRING := '0';
5. Also declare the variable iText : INT;
6. Configure the property "Dynamic texts \(\rightarrow\) Text List" with 'Textlist_A'.
7. Configure the property "Dynamic texts \(\rightarrow\) Text index" with PLC_PRG.strTextID.
8. Configure the property "Dynamic texts \(\rightarrow\) Tooltip index" with PLC_PRG.strTooltipID.
9. In POU PLC_PRG implement the CASE instruction as shown below.
\(\Rightarrow\) The variables in property "Dynamic Texts" are programmed.
10. Configure the property "Inputconfiguration \(\rightarrow\) OnMouseclick" for Execute ST-Code with PLC_PRG.iText := (PLC_PRG.iText + 1) MOD 4;
\(\Rightarrow\) For element "Text field" an user input is configured.
11. Build, download and start the application.
\(\Rightarrow\) The application is running. The visualization opens. In the text field the text None is output. When you as user click on the element, the text changes to Dynamic File_A. And the matching tooltip is available: Information A. With each click the text changes according to the CASE instruction.

\section*{Dynamic: File_A}

Sh

> Information A
```

CASE instruc- CASE iText OF
tion 0: strTextID := '0';
strToolTipID := '0';
1: strTextID := '1';
strToolTipID := '4';
2: strTextID := '2';
strToolTipID := '5';
3: strTextID := '3';
strToolTipID := '6';
ELSE
strTextID := '0';
strToolTipID := '0';
END_CASE;

```

\section*{Text output: \\ Configuring a static + dynamic text output}

In property "Texts \(\rightarrow\) Text" you can define a text in order to get a static text output. A text in "Texts \(\rightarrow\) Tooltip" will be displayed as tooltip. You can configure the text in a way, that the content of a variable is additionally output.

You can extend a static text by (exactly) 1 placeholder including a formatting definition, in order to output the content of a variable at this place at runtime. The variable must be assigned in property "Text variable". When the variable value changes in the application code, then at the same time the output in the visualization changes.
\(\checkmark\) Precondition: A project containing a visualization is opened.
1. Open the visualization and insert an element "Text field".
\(\Rightarrow\) The "Properties" view shows the element configuration.
2. Configure the property "Texte \(\rightarrow\) Text": File name: \%s
\(\Rightarrow\) The text contains the placeholder \(\% s\).
3. In POU PLC_PRG of the application declare a type-conform variable strFileName : STRING :='File_A';
4. Configure the property "Text variable" of the text field with PLC_PRG.strFileName. \(\Rightarrow\) At runtime the variable value will be output instead of the placeholder.
5. Build, download and start the application.
\(\Rightarrow\) The application is running. The visualization opens. The text field element displays the text: File name: File_A
* \({ }^{4}\) Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329

Configuring text You can use the "Text field" element in order to output the text given by a variable, or to provide input in a text a place, where the user can give input on the variable.
field

Additionally you can configure a text input. In this case on an user input an input field in the element "Text field" will appear. As a precondition you must have configured an user input action in the property "Inputconfiguration".

Chapter 6.4.5.8 "Setting Up Multiple Languages" on page 2906

\section*{Showing text as a tooltip}
\(\checkmark\) Precondition: A project containing a visualization is opened.
1. Open the visualization and insert an element "Text field".
\(\Rightarrow\) The "Properties" view shows the element configuration.
2. Configure the property "Texts \(\rightarrow\) Text": File name: \%s
\(\Rightarrow\) The text contains the placeholder \(\% \mathrm{~s}\)
3. Configure the property "Texts \(\rightarrow\) Tooltip": Storage location: \%s
4. In POU PLC_PRG of the application declare a type-conform variable strFileName : STRING :=-'File_A';
5. In POU PLC_PRG of the application declare also the variable strFileDir : STRING :=-'D:/Data';
6. Configure the property "Text variable" of the text field with PLC_PRG. strFileName. \(\Rightarrow\) At runtime the variable value will be output instead of the placeholder.
7. Configure the property "Tooltip" of the text field with PLC_PRG.strFileDir.
8. Build, download and start the application.
\(\Rightarrow\) The application is running. The visualization opens. The text field element shows the text File name: File_A. When the mouse cursor is moved above the text field, the tooltip will be displayed: Storage location : D/Data.

\section*{File name : File_A}

Storage location: D:/Data

\section*{Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329}

Configuring ele- The user should be able to enter text in a text field. For this configure an input of type "Write ment "Text field" for text input variable" on a text output variable. This text output variable will store the text input of the user and will display this text instead of the placeholder (this is \%s in the example below). You specify the text output variable in the property "Text variables \(\rightarrow\) Text variable".
\(\checkmark\) Precondition: A project containing a visualization is opened.
1. In POU PLC_PRG of the application declare a string variable: strInput : STRING;
2. Open the visualization and insert an element "Text field".
\(\Rightarrow\) The "Properties" view shows the configuration of the element.
3. In property "Texts \(\rightarrow\) Text" enter Input: \%s.
4. In property "Inputconfiguration" for mouse action "OnMouseClick" click on "Configure" to open the "Input Configuration" dialog box. There choose action "Write a Variable" and activate option "Use text output variable". Close with "OK".
5. In the element property "Text variables \(\rightarrow\) Text variable" assign the text output variable PLC_PRG.strInput.
6. Build, download and start the application.
\(\Rightarrow\) The application is running. The visualization opens. The element outputs the text: Input: . Click in the element to open an input field, where you can enter a string. After having terminated the input by [Enter], the text will be adopted.

Input: My current input.

\section*{Animating the text display}

Configure the property "Font variables" in order to animate the display of a text. All base elements have this property, additionally the table, scrollbar and text field element.

Chapter 6.4.5.10.3 "Animating a text display" on page 2914

\section*{Configuring the}

\section*{'Label' element}

Making an element invisible
\(\checkmark \quad\) Precondition: A project containing a visualization is opened.
1. Open a visualization and insert an element "Label".
\(\Rightarrow\) The "Properties" view with the pre-set property configuration opens.
2. Configure the property "Texts \(\rightarrow\) Text": Visualization A.

You can configure the property "State variables \(\rightarrow\) Invisible" in order to hide an element in the visualization.
\(\checkmark\) Precondition: In the visualization you have configured a text field, which gets visible only, if a certain application variable gets TRUE. For example in order to show certain instructions or descriptions only in a certain state of the machine.
1. For the text field element configure the property "Texts \(\rightarrow\) Text" with Error detected:

Do the following....
Configure the property "Text properties \(\rightarrow\) Font color" with "dark red".
2. In PLC PRG declare the variable bIsInvisible : BOOL : TRUE; (this is the initialization for the current example; normally the variable should be set to TRUE by the application program under certain conditions.
3. Configure the property "State variables \(\boldsymbol{\rightarrow}\) Invisible" with PLC_PRG.bIsInvisible.
4. Build, download and start the application.
\(\Rightarrow\) The application is running. The visualization opens and the text field is not visible. When you set bIsInvisible to TRUE, the textfield will be displayed.

\subsection*{6.4.5.5.6 How to display variable values in the visualization}

There are simple to very specialized visualization elements for displaying data from a running application.

\section*{Examples:}
- Simple output of variable values: For example, you can configure a purely formatting specification for a "Rectangle" element in the "Text" property and the variable whose value is to be displayed in the element in the "Text variable" property.
- Display of structured variable values (structure, array, function block): You use the "Table" element and specify an array variable in its configuration in the "Data array" property whose values are to be displayed in the table. One-dimensional arrays can also be displayed in a "Histogram".
- Display of values by image switching. Example: A specific screen is displayed depending on the error message that occurs. You do this by configuring an "Image" element with a variable for the "Bitmap ID variable" property.
- Display a variable value as a bar or with a pointer on a scale: You specify a variable in the "Value" property of the "Bar Display" element or "Meter" element to display its value as a bar on a horizontal or circular scale.
- Display of alarms: The alarms configured in the alarm management of the application can be made visible by means of the "Alarm Table" and "Alarm Banner" elements in the user interface.
- "Trace" and "Trend": For graphical recording of variable values over a period of time.

For details, see the descriptions of the element properties.
See also
- \({ }^{\text {® }}\) Chapter 6.4.5.20.1 "Visualization Elements" on page 2987
- \({ }^{\text {• }}\) Chapter 6.4.5.5.5 "Using texts" on page 2878
- \(\quad\) Chapter 6.4.5.2.4 "Displaying Array Data in a Histogram" on page 2862

\subsection*{6.4.5.5.7 How to Change Variable Values via the Visualization}

In addition to displaying values from the controller, a user interface is also used to enter and change values.

In general, you can configure user input for each element in its "Input configuration" properties. Moreover, elements have been developed especially for specific input.
Examples:
- A "Button" element (or "Rectangle" element, and so on) that is clicked to open a predefined dialog visualization for easily specifying a value.
- A "Slider" element for changing the value of a variable by moving visual element parts, for example with the mouse. In the case of the slider: The element adjusts the value of a variable, depending on the position of the slider within the slider. You define the value range of the slider bar by means of the scale start and scale end.
- A switch element (example; "Power Switch") for setting a Boolean value.
- A "Spin Box" element for incrementing or decrementing the value of a variable in defined intervals.
- A "Button" element for writing a recipe, executing a specific ST code, writing a specific variable, and so on (definition in the input configuration).
See also
- \({ }^{\text {® }}\) Chapter 6.4.5.21.5 "Visualization Elements" on page 3412
- \({ }^{\bullet}\) Chapter 6.4.5.6 "Configuring user inputs" on page 2885

\subsection*{6.4.5.5.8 Designing a background}

You can design the background of your visualization in color or with an image. To do this, use the command "Visualization \(\rightarrow\) Background".
See also
- \(\xi^{*}\) Chapter 6.4.5.21.2.10 "Command 'Background'" on page 3349
- \(\quad\) \& Chapter 6.4.5.21.3.15 "Dialog 'Properties' of Visualization Objects" on page 3388

Configuring an image as a background

In addition, you can use the property "Integrate background" in the dialog "Properties" of a visualization object to specify whether the background image should always be displayed in its entirety or whether it should be truncated.

Requirement: A project with a visualization is open.
1. Open the visualization and select the command "Visualization \(\rightarrow\) Background".
2. Activate the option "Image" and open the input assistant.
3. Select an image in the dialog "Input Assistant".
\(\Rightarrow\) The image serves as a background image.

See also
- Chapter 6.4.5.21.3.15 "Dialog 'Properties' of Visualization Objects" on page 3388

\section*{Configuring a colored background}
\(\square\) Requirement: A project with a visualization is open.
1. Open the visualization and select the command "Visualization \(\rightarrow\) Background".
2. Activate the option "Color".
3. Select a style color such as "Element background color" from the selection list.
\(\Rightarrow\) The background of the visualization is colored.

\subsection*{6.4.5.6 Configuring user inputs}

\subsection*{6.4.5.6.1 General}

User inputs for a visualization are configured in order to operate the visualization.
For this purpose, you configure input events on visualization elements where follow-up actions are triggered. The combination of user inputs and follow-up actions are defined in the "Input configuration" of an element. For example, you can select a mouse click on an element as the input event and opening a dialog box as the input action.
Keyboard events can also be configured that trigger actions in a specific visualization window when the events occur. You program this kind of input configuration for a visualization in its "Keyboard configuration" editor.

In addition, keyboard events can be configured that occur in all visualizations programmed in the application. You configure this kind of input configuration per application below the visualization manager in the "Standard keyboard shortcuts" tab.
Input is usually performed with the mouse and keyboard as controlling device. You can also configure a user operation by means of gestures.
If a visualization device is not equipped with a mouse, then you can activate default keyboard usage. Then a user can operate the visualization with the keyboard only by navigating with the arrow keys and triggering events by pressing the [Enter] key.
If a visualization device is not equipped with a keyboard, then you can call a virtual keyboard or a virtual numeric keypad.

\section*{NOTICE!}

Configure keyboard events only for keys that the visualization device supports.
« Chapter 6.4.5.21.2.2 "Command 'Keyboard Configuration"' on page 3341
* Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration"' on page 3370
\# Chapter 6.4.5.21.4.3 "Tab 'Visualization Manager' - 'Default Hotkeys"' on page 3402
1. Event handler of the application. Requirement: The event handler is activated.
2. Events of the default keyboard usage
3. Events of the keyboard usage are configured in the tab "Visualization manager" - "Default hotkeys".
4. Events of keyboard usage are configured in the tab "Keyboard configuration" for the currently visible visualization.
« Chapter 6.4.5.6.6 "Capturing user input events" on page \(2895 \Longleftrightarrow\) Chapter 6.4.5.21.1 "Keyboard Shortcuts for Default Keyboard Action" on page 3338
«乡 "Tab 'Keyboard configuration"' on page \(3341 \Leftrightarrow\) Chapter 6.4.5.21.2.2 "Command 'Keyboard Configuration'" on page 3341
Chapter 6.4.5.21.4.3 "Tab 'Visualization Manager' - 'Default Hotkeys'" on page 3402
« \(\boldsymbol{y}^{2}\) Chapter 6.4.5.21.4.3 "Tab 'Visualization Manager' - 'Default Hotkeys'" on page 3402

\subsection*{6.4.5.6.2 Configuring user inputs for visualization elements}

All base elements and some common control elements have the "Input configuration" property. This is where you can configure a user input for an element. For this purpose, you select an input event and an input action.

\section*{Configuring \\ user inputs}
\(\checkmark\) Requirement: A project is open with a visualization.
1. Open the visualization and added a "Button" element.
\(\Rightarrow\) The "Properties" view opens for the new button.
2. Configure the property "Text" with Number of clicks: \%i.
3. Declare a variable iClicks : INT; in the application in the PLC_PRG POU.
4. Configure the "Text variable" property of the button as PLC_PRG.iClicks.
\(\Rightarrow\) At runtime, its variable value will replace the placeholder in the "Text" property.
5. In the "Input configuration" property, click the "Configure" button in the OnMouseClick line.
6. Select the Execute ST code action from the list of possible actions and click the \(\square\) symbol.
\(\Rightarrow\) The action appears in the list of actions to be executed. The blank implementation of the action appears in the window area to the right of the list.
\begin{tabular}{|c|c|c|}
\hline \(F\) Execute ST-Code & \multicolumn{2}{|l|}{Execute ST-Code} \\
\hline & 1 & \\
\hline
\end{tabular}
7. Program the action in the editor at "Execute ST code":
```

PLC_PRG.iClicks := PLC_PRG.iClicks + 1;

```
8. Click "OK" to close the dialog box.
\(\Rightarrow\) The user input is configured.
9. Compile, download, and start the application.
\(\Rightarrow\) The application runs. The visualization opens. If the user clicks the button, then the action is executed, the variable PLC_PRG.iClicks is incremented, and the number of clicks is printed.

« Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration"' on page 3370

\subsection*{6.4.5.6.3 Configuring gesture recognition}

You can execute a visualization on a device that is operated by means of gestures. The visualization retains its user input configuration for mouse and keyboard operation and also recognizes gestures and multi-touch events. Gesture events are recognized and interpreted as mouse events.

For this purpose, activate the "Activate multi-touch" setting in the visualization manager.
Elements of the type "Frame" or "Tab control element" display contents that a user should be able to move. Therefore, configure their "Scaling type" property with "Fixed and scrollable".

Gesture recognition for:
- Tapping

A quick tap on the element is interpreted as a mouse click.
- Panning

Pressing, moving, and releasing with one finger in a frame or with a tab control element (in the window area of the element) will move the contents.
- Multi-finger touch detection

Touching several elements at the same time will input for all elements. These touch events are interpreted as the respective mouse events.
Example:
Two-hand operation in order to trigger an action with two simultaneous inputs on two different elements.
Virtual mixing console where multiple sliders can be operated at the same time.
In addition, the IGestureEventHandler interface is available in the
VisuElems. VisuElemBase library. You can use this to implement application code that recognizes gestures and executes follow-up actions.

The following display variants can execute a visualization on a multi-touch device
- CODESYS WebVisu
(4) "Implementing event handling with multi-touch" on page 2888

\section*{Using gestures to control visualizations}
\(\checkmark\) Requirement: A project is open with a visualization and a user input configuration. It contains one button. The visualization device is a display with multi-touch support.
1. Double-click the "Visualization manager" object.
\(\Rightarrow\) The editor opens.
2. Click the "Settings" tab.
3. In the "Additional settings" group, activate the "Activate multi-touch" option.
4. Compile, download, and start the application.
\(\Rightarrow\) The application runs. The visualization opens. When a user touches the display of the visualization device, the visualization responds. Elements that respond to mouse events also respond to touch events Several buttons can be pressed at the same time. Scrollable frames or tab control elements are displayed without scrollbars and can be moved by panning.

Note: The "Scaling type" property of elements type "Frame" or "Tab control element" must be set to "Fixed and scrollable".
※ Chapter 6.4.5.20.1.6 "Visualization Element 'Frame"' on page 3053
* Chapter 6.4.5.20.1.10 "Visualization Element 'Tabs'" on page 3084

Implementing event handling with multi-touch
\(\checkmark\) Requirement: The device is multi-touch capable
1. Implement and register a function block that receives the gesture events.
```

A FUNCTION_BLOCK GesturesHandler IMPLEMENTS
VisuElems.VisuElemBase.IGestureEventHandler2
VisuElems.g_VisuEventManager.SetGestureEventHandler(THIS^);

```
2. Implement and register a function block that sets the touch areas.
```

A FUNCTION_BLOCK RectProvider IMPLEMENTS
VisuElem\overline{s.VisuElemBase.IApplicationRectangleProvider}
VisuElems.g_VisuRectangleProvider := THIS^;

```
3. Implement actions as application code that are executed when a gesture event occurs

\subsection*{6.4.5.6.4 Configuring text input with the virtual keyboard}

A visualization is usually configured so that it calls a virtual keyboard for a text input event when an input device is not available. For this purpose, the follow-up action "Write variable" is preset accordingly in the user input: The value "Standard" is selected for the "Input type" setting.

However, you can also configure especially how text is input. For this purpose, more input types are available in the user input, such as Text input or the listed visualizations. These visualizations have the visualization type "Numpad/Keypad" and display virtual keyboards or numeric keypads.
In the "Settings for default text input" setting of the visualization manager you can preset a keyboard visualization that is called from all visualizations in the application when required. This is possible without having to customize the user inputs of the visualizations.
Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration"" on page 3370

\section*{Configuring text input especially for virtual keyboards}
\(\square\) Requirement: A project is open with a visualization.
1. Declare an input variable in the PLC_PRG program.
\(\Rightarrow\) VAR_INPUT stInput : STRING; VAR_END
2. Add a button to the visualization and select the element.
3. Configure the property "Texts \(\boldsymbol{\rightarrow}\) Text" with Text input: \%s.
4. Configure the property "Text variables \(\rightarrow\) Text variable" with PLC_PRG.stInput.
5. Click auf "Configure" in the property "Input configuration \(\rightarrow\) OnMouseClick".
\(\Rightarrow\) The "Input Configuration" dialog box opens. The selected input event is printed below the caption.
6. Select the "Write variable" action.
7. Select the visualization Visudialogs. Keypad in "Input type" of the implementation of the action.
\(\Rightarrow\) The virtual keyboard Visudialogs. Keypad is selected as the input device.
8. Compile, download, and start the application.
\(\Rightarrow\) The visualization opens.
9. Click the button as a visualization user.
\(\Rightarrow\) The virtual keyboard appears and allows text input by means of the mouse.
```

Text input:My user defined text

```


\section*{Configuring numeric input especially for virtual numeric keypads}
\(\checkmark\) Requirement: A project is open with a visualization.
1. Declare an input variable in the PLC_PRG program.
```

| VAR_INPUT iInput : INT; VAR_END

```
2. Open the visualization and added a "Rectangle" element.
3. Select the element in the editor.
\(\Rightarrow\) The properties are visible in the "Properties" view.
4. Configure the property "Texts \(\boldsymbol{\rightarrow}\) Text" with Number input: \% i.
5. Configure the property "Text variables \(\boldsymbol{\rightarrow}\) Text variable" with PLC_PRG.iInput.
6. In the "Input configuration" property, click the "Configure" button in the onMouseClick line.
\(\Rightarrow\) The "Input Configuration" dialog box opens. The selected input event is printed below the caption.
7. Select the Write variable action from the list of possible actions and click the symbol.
\(\Rightarrow\) The action appears in the list of actions to be executed. The blank implementation of the action appears in the window area to the right of the list.
8. Select the following settings:
"Input type" set to VisuDialogs. Numpad.
"Choose variable to edit" set to "Use text output variable".
"Dialog title" set to 'My virtual numpad'.
9. Click "OK" to close the dialog box.
\(\Rightarrow\) The user input is configured.
10. Compile, download, and start the application.
\(\Rightarrow\) The application runs. The visualization opens. When a user clicks the rectangle, the numeric keypad opens.

\section*{Input number: 154}

\section*{My virtual numpad}
\[
154
\]


\section*{Defining standard text input}
\(\checkmark\) Requirement: A project is open with a visualization and a user input configuration. For all "Write variable" follow-up actions, the value "Default" is selected for the "Input type" setting.
1. Double-click the visualization manager.
2. Click in the default text input in the "Settings" tab ("Default text input" group) and assign visualizations.
\(\Rightarrow\) These visualizations are defined as default text input. If a display variant does not have a keyboard, then these visualizations are called without you having to adapt the user input.

\subsection*{6.4.5.6.5 Configuring Keyboard Shortcuts}

You can define keyboard shortcuts and assign specific actions to them. At runtime, a visualization detects the keyboard input event and executes the action.
There are different locations where you can configure a keyboard input event.
The options include the following:
- Configure keyboard input for a specific element.
- Configure keyboard input for a specific visualization.
- Configure keyboard input that is valid for all visualizations.
- Select the default hotkeys.

If the visualization integrated in CODESYS is executed, then you can deactivate the keyboard input of the visualization in order to use the keyboard shortcut from CODESYS in this state.

Configuring You can define a keyboard shortcut that triggers an action for an element. The elekeyboard short- ment has to be visible and operable. For this purpose, the property "Input configuration cuts for ele- \(\quad \rightarrow\) Keyboard shortcuts" is available in the "Properties" view of the visualization editor. ments
\(\square\) Requirement: A CODESYS project is open with the existing visualizations visEllipse and visRectangle.
1. Select the application in the device tree and add a visualization named visMain.
\(\Rightarrow\) The visualization editor opens.
2. In the "Visualization Toolbox" view, select and drag the "Frame" element to the editor.
\(\Rightarrow\) The "Configuration of Frame Visualizations" dialog opens.
3. Double-click in succession the visEllipse and visRectangle visualizations in "Available Visualizations". \(\Rightarrow\) The visualizations appear in "Selected Visualizations".
4. Click "OK" to exit the dialog.
\(\Rightarrow\) The visualization contains a new element type "Frame". The 2 selected visualizations appear under its property "References".
In the editor, the frame shows the visualization with the index 0 .
5. Add a button and configure its properties:

Select Rectangle in the property "Texts \(\rightarrow\) Text".
In the "Input configuration \(\rightarrow\) OnMouseDown" property, select "Toggle frame visualization" for the visualization visRectangle.
Specify the value R in the property "Input configuration \(\rightarrow\) Keyboard shortcuts \(\rightarrow\) Key".
\(\Rightarrow\) The button has a user input and a keyboard shortcut.
6. Add a button and configure its properties:

Select Ellipse in the property "Texts \(\rightarrow\) Text".
In the "Input configuration \(\rightarrow\) OnMouseDown" property, select "Toggle frame visualization" for the visualization visEllipse.
Specify the value E in the property "Input configuration \(\rightarrow\) Keyboard shortcuts \(\rightarrow\) Key".
\(\Rightarrow\) The button has a user input and a keyboard shortcut.
7. Click "Online \(\rightarrow\) Login" for the device and start the application.
\(\Rightarrow\) The visualization starts. It has a frame where one of the referenced visualizations runs. Focus on the visEllipse visualization and press [E]. The visualization switches the contents in the frame to the visEllipse visualization. When you press \([R]\), the visualization switches the contents in the frame to the visRectangle visualization.


Chapter 6.4.5.20.1.6 "Visualization Element 'Frame"' on page 3053
ⓨ "Input action 'Switch Frame Visualization"" on page 3377

Configuring You can define keyboard shortcuts that trigger an input action on a specific visualization. The keyboard short- "Keyboard Configuration" tab in the editor of the visualization is used for this purpose. cuts for a specific visualization

\section*{\(\checkmark \quad\) Requirement: A CODESYS project is open with the visualizations visEllipse and visRectangle.}
1. Open the CODESYS TargetVisu object and select visEllipse as the start visualization.
2. Open the visEllipse visualization and click the "Keyboard Configuration" tab.
3. Click "Visualizations \(\rightarrow\) Keyboard Configuration".
\(\Rightarrow\) The "Keyboard Configuration" tab opens.
4. Select the value C in the "Key" column.
5. Activate the "Press key" option.
6. Select the "Change shown visualization" value in the "Action Type" column.
7. Select visRectangle in the "Action" column.
\(\Rightarrow\) The user input is configured for [C].
8. Open the visRectangle visualization and click the "Keyboard Configuration" tab.
9. Select the value C in the "Key" column.
10. Activate the "Press key" option.
11. Select the "Change shown visualization" value in the "Action Type" column.
12. Select visEllipse in the "Action" column.
\(\Rightarrow\) The user input for [C] is also configured for this visualization.
13. Build the application.
14. Click "Online \(\rightarrow\) Login" for the device and start the application.
\(\Rightarrow\) The visualization starts and displays an ellipse. Focus on the visEllipse visualization and press [C]. The visRectangle visualization is displayed. Focus on the visualization and press [C] again. Now the visualization is switched again to visRectangle.

4. "Tab 'Keyboard configuration"' on page 3341
ⓨ "Input action 'Change Shown Visualization'" on page 3373

Configuring keyboard shortcuts for all visualizations in the application

You can define keyboard shortcuts that trigger the same input action for all visualizations of the application. The "Default Hotkeys" tab in the Visualization Manager is available for this purpose.


Requirement: A project is open with a visualization.
1. Open the visualization.
2. Add a rectangle.
3. Configure the property "Texts \(\rightarrow\) Text" with Keyboard shortcut.
4. Double-click the "GlobalTextList" object.
5. Click in the table, "Add Language", and then specify de.
\(\Rightarrow\) The language de is configured.
6. Click in the table, "Add Language", and then specify en.
\(\Rightarrow\) The language en is configured.
7. Configure translations for de and en for the text Keyboard shortcut.
\(\Rightarrow\) Hotkey Keyboard Shortcut Hotkey
8. Open the Visualization Manager and select the "Default Hotkeys" tab.
9. Specify D in the "Key" column.
10. Activate the "Press key" option.
11. Select the "Change language" value in the "Action Type" column.
12. Select the language de in the "Action" column.
\(\Rightarrow\) The keyboard event for \([D]\) is configured.
13. Specify \(D\) in the "Key" column.
14. Activate the "Press key" option.
15. Select the "Alt" option.
16. Select the "Change language" value in the "Action Type" column.
17. Select the language en in the "Action" column.
\(\Rightarrow\) The keyboard event for \([A I t]+[D]\) is configured.
18. Compile, download, and start the application.
\(\Rightarrow\) The visualization opens.
19. As the visualization user, press [D].
\(\Rightarrow\) The text is displayed in the language de.
« Chapter 6.4.5.21.4.3 "Tab 'Visualization Manager' - 'Default Hotkeys'" on page 3402
步 "Input action 'Change Language'" on page 3372

\section*{Activating standard keyboard handling}

When you activate the universal keyboard shortcuts for standard keyboard handling, the user can operate the visualization without a mouse. Elements that respond to user input can process a keyboard event instead of a mouse event without adapting its input configuration.
\(\checkmark\) Requirement: A project with a visualization is open.
1. Click the "Visualization Manager" object.
2. Activate the "Activate standard keyboard handling" option.
\(\Rightarrow\) The universal keyboard shortcuts are activated.
3. Download the application to a device and start the application.
\(\Rightarrow\) The visualization starts. Now operation can proceed without the mouse. You can navigate in the window by means of the [Arrow] and [Tab] keys and press [Enter] instead of the mouse button.

③ Chapter 6.4.5.21.1 "Keyboard Shortcuts for Default Keyboard Action" on page 3338

Activating and deactivating keyboard shortcuts for integrated visualizations

If you execute the visualization as an integrated visualization, then the "Visualization \(\rightarrow\) Activate Keyboard Usage" command is available in order to deactivate the capturing of keyboard events. It is actually possible for the same keyboard shortcuts to be defined in the visualization and in CODESYS
When you activate the command, the visualization executes the configured keyboard events.
When you deactivate the command, CODESYS executes the keyboard events. Capturing keyboard events is then deactivated for the visualization.
出 Chapter 6.4.5.21.2.4 "Command 'Activate Keyboard Usage'" on page 3343

\subsection*{6.4.5.6.6 Capturing user input events}

You can capture user input events in the application. For this purpose, you can implement a function block that is executed when user events occur.

Capturing the When the user completes the input of a value (in an input field), an edit control event is closed. writing of varia- You can capture this event in the application as follows. bles
1. Create a function block that implements the VisuElems.IEditBoxInputHandler interface from the VisuElemBase library.
2. Pass the instance to the global event manager VisuElems.Visu_Globals.g_VisuEventManager by calling the SetEditBoxEven \(\bar{t} H a n d l e r ~ m e ̀ t h o d . ~\)

```

control field;
String gets composed by method 'VariableWritten;
Result is displayed in the lower rectangle of the
visualization *)
END_VAR
VAR
inst : POU;
bFirst : BOOL := TRUE;
END_VAR
IF bFirst THEN
bFirst := FALSE;
VisuElems.Visu_Globals.g_VisuEventManager.SetEditBoxEventHandler(ins
t);
(* Call of method VariableWritten *)
END_IF
POU implemen- FUNCTION_BLOCK POU IMPLEMENTS VisuElems.IEditBoxInputHandler
tation (* no fuřther declarations, no implementation code *)
Method METHOD VariableWritten : BOOL
VariableWrit (* provides some information always when an edit control field is
ten assigned closed in the visualization, that is a variable gets written by
to POU user input in one of the upper rectangles *)
VAR_INPUT
pVar : POINTER TO BYTE;
varType : VisuElems.Visu_Types;
iMaxSize : INT;
pClient : POINTER TO VisuElems.VisuStructClientData;
END_VAR
// String stInfo, which will be displayed in the lower rectangle,
is composed here
PLC_PRG.stInfo := 'Variable written; type: ';
PLC_PRG.stInfo := CONCAT(PLC_PRG.stInfo, INT_TO_STRING(varType));
PLC_PRG.stInfo := CONCAT(PLC_PRG.stInfo, ', \overline{adr': ');}
PLC_PRG.stInfo := CONCAT(PLC_PRG.stInfo, DWORD_TO_STRING(pVar));
PLC_PRG.stInfo := CONCAT(PLC_PRG.stInfo, ', by : 'j);
PLC_PRG.stInfo := CONCAT(PLC_PRG.stInfo,
SEL(pClient^.globaldata.clienttype =
VisuElems.Visu ClientType.Targetvisualization,'other visu',
'targetvisu'));

```

Capturing key- When the user presses and releases the key, a keyboard event is triggered in the visualization. board events You can capture this event in the application as follows.
1. Create a function block that implements VisuElems.IVisuUserEventManager from the VisuElemBase library.
2. Pass the instance to the global event manager

VisuElems.Visu_Globals.g_VisuEventManager by calling the SetKeyEventHandler method.
\begin{tabular}{|c|c|}
\hline Example & A visualization has one text output element. The text output element is a rectangle where the contents of the text variable PLC_PRG. stInfo are printed. The text variable contains information about the last key pressed by the user. \\
\hline & Properties of the rectangle for the text output \\
\hline & "Texts \(\rightarrow\) Text" \({ }^{\circ} \mathrm{Os}\) \\
\hline & "Text variables \(\boldsymbol{\rightarrow}\) Text variable" \({ }^{\text {a }}\) " PLC_PRG.stInfo \\
\hline Implementation of the PLC_PRG program & ```
PROGRAM PLC_PRG
VAR_INPUT
    stInfo : STRING;
END_VAR
VAR
    inst : POU;
    bFirst : BOOL := TRUE;
END_VAR
IF bFirst THEN
    bFirst := FALSE;
VisuElems.Visu_Globals.g_VisuEventManager.SetKeyEventHandler(inst);
END_IF
``` \\
\hline Implementation of the POU function block & FUNCTION BLOCK POU IMPLEMENTS VisuElems.IKeyEventHandler (* no further declarations, no implementation code *) \\
\hline Implementation of the VariableWrit ten method of the POU function block & ```
/// This method will be called after a key event is released.
/// RETURN:
/// TRUE - When the handler has handled this event and it should
not be handled by someone else
/// FALSE - When the event is not handled by this handler
METHOD HandleKeyEvent : BOOL
VAR_INPUT
    -/// Event type. The value is true if a key-up event was
released.
    bKeyUpEvent : BOOL;
    /// Key code
    dwKey : DWORD;
    /// Modifier. Possible values:
    /// VISU_KEYMOD_SHIFT : DWORD := 1;
    /// VISU_KEYMOD_ALT : DWORD := 2;
    /// VISU_REYMOD_CTRL : DWORD := 4;
    dwModifiers : DW\overline{ORD;}
    /// Pointer to the client structure were the event was released
    pClient : POINTER TO VisuStructClientData;
END VAR
VAR-
END_VAR
PLC_PRG.stInfo := 'KeyEvent up: ';
PLC_PRG.stInfo := CONCAT(PLC_PRG.stInfo,
BOOL_TO_STRING (bKeyUpEvent));
PLC_\overline{PRG.stInfo := CONCAT(PLC_PRG.stInfo, ', key: ');}
PLC_PRG.stInfo := CONCAT(PLC_PRG.stInfo, DWORD_TO_STRING(dwKey));
PLC_PRG.stInfo := CONCAT(PLC_PRG.stInfo, ', mo\overline{difieqer: ');}
PLC-PRG.stInfo := CONCAT(PLC_PRG.stInfo,
DWORD_TO_STRING(dwModifiers));
PLC_P\overline{RG.}.\overline{StInfo := CONCAT(PLC_PRG.stInfo, ', by: ');}
PLC_PRG.stInfo := CONCAT(PLC_PRG.stInfo,
``` \\
\hline
\end{tabular}
```

SEL(pClient^.globaldata.clienttype =
VisuElems.Visu_ClientType.Targetvisualization,
'other visu',
'targetvisu'));

```
Recording vari-
able value
changes trig-
gered by input
events

All visualization elements that change the value of a variable by user input call the IValueChangedListener interface. With this interface, the value changes can be recorded and then processed programmatically.
1. Implement a function block (example: POU) that implements the IValueChangedListener interface.

FUNCTION_BLOCK POU IMPLEMENTS VisuElems.IValueChangedListener
\(\Rightarrow\) In the device tree, the "ValueChanged" method is inserted below the function block.
2. In a program (example: "PLC_PRG"), implement the IEC code that registers the interface.

VisuElems.g_itfValueChangedListenerManager.AddValueChangedListene r(itfValueChangedListener)
\(\Rightarrow\) "PLC_PRG" receives all value changes by means of the "ValueChanged" method. Now you can record and process the value changes.

\subsection*{6.4.5.7 Setting Up User Management}
6.4.5.7.1 General ..... 2899
6.4.5.7.2 Creating runtime-based user management initially ..... 2900
6.4.5.7.3 Switch to legacy user management. ..... 2902
6.4.5.7.4 Creating a group with administrator rights for visualizations. ..... 2902
6.4.5.7.5 Setting up access control and login operation ..... 2903
6.4.5.7.6 Group-dependent restriction of functionality. ..... 2905

\subsection*{6.4.5.7.1 General}

If you want to deploy a web visualization and securely protect it from third-party access, it is advantageous to set up runtime-based user management for it.
Runtime-based user management makes sure that users will log in securely with current technology. The credentials are managed using standard web technologies. In addition, user accounts can be created at runtime via secure channels, user credentials can be managed, and user permissions can be restricted according to the role of the user.

For this, it is necessary that the visualization user management is coupled with the device user management. You need to map the groups with their rights in the visualization to the groups in the device. You configure the user accounts centrally on the "Users and Groups" tab of the device editor.

If you do not want this because you value independent user management for your visualization and accept the outdated technology for this, then you can alternatively switch to legacy user management. This is not recommended if your controller supports runtime-based user management.
«2 Chapter 6.4.1.11.4 "Handling of Device User Management" on page 1971
² Chapter 6.4.1.6.5 "Setting up a user management" on page 1825

\section*{Runtime-Based User Management}

User management linked to device user management for higher security.
Whenever possible, always use runtime-based user management.
If you want to convert to your existing user management, then open
the "User Management \(\rightarrow\) Groups" tab in the "Visualization Manager" and click
the "Convert to Runtime-Based" button. At this time, the user groups are taken
over. The credentials of the users will be lost because these then have to be set
up in the controller.

The users are configured in the device user management on the "Users and Groups" tab of the device editor. If you want a user to have access only to a visualization but not to any device, then assign them a group membership and restrict the group permissions so that they are only allowed access to a visualization but not to any other device functionality.
\({ }^{\mu}\) Chapter 6.4.5.7.2 "Creating runtime-based user management initially" on page 2900
« Chapter 6.4.1.21.2.8.14 "Tab 'Users and Groups"" on page 2450

Legacy User Management

Deprecated but still available user management.
Legacy user management runs independently of device user management.

In the deprecated visualization user management, you define user groups and visualization users in the "Visualization Manager". In the user management dialogs, which request the credentials at runtime, authorized users log in to their visualization with their group permissions. In the visualization, elements can have group-dependent permissions and therefore be visible only to the group, for example.
In a project with several applications, you can configure user management for each application.

\section*{NOTICE!}

When a visualization user management exists, an unregistered user automatically receives the access rights from the None group.
(4) "Tab 'Groups'" on page 3404

Login operation on the display variants

The login operation is always the same for all display variants. This means it is independent of which user management is set up.
Login operation on the display variants:
- The integrated display ignores the user management functionality and starts with the user who logged in to the controller using CODESYS.
- When a TargetVisu is started, no user is initially logged in and the operator then has to log in if this is required by the designed visualization
- When starting a remote TargetVisu, the user from the communication settings of the remote TargetVisu itself is initially logged in. This can be changed via a new login within the visualization.
- Web visualizations start with the secure login operation of runtime-based user management

\subsection*{6.4.5.7.2 Creating runtime-based user management initially}
Whenever possible, use a runtime version >= 3.5 SP18 Patch 1 and CODESYS
Visualization version >= 4.2.0.0 for a new project with a visualization.

\section*{With standard groups}

\section*{Empty with None group}

First create an empty user management if you want new created groups with self-determined permissions.

Requirement: You have inserted a WebVisu below the Visualization Manager.
1. Double-click the "Visualization Manager" object in the device tree.
2. Select the "User Management" tab.
3. Click the [Create Runtime Based Empty User Management] button.
\(\Rightarrow\) The "Groups" and "Settings" tabs are displayed.
\(\Rightarrow\) The "Group" tab is open.
4. To create new user groups, click in the empty field in the "Group Name" column and specify the name of the new group.
5. Connect the application to the controller and click the [Upload device groups names] button.
\(\Rightarrow\) The user groups of the device user management are uploaded and are now selectable in the "Mapping in Runtime Group" column.
6. On the "User Management" tab: For the user groups of the visualization of the "Group Name" column, select a group of the device user management from the list box in the "Mapping in Runtime Group" column to which the user group of the visualization should be assigned.
\(\Rightarrow\) By assigning the visualization user groups to the device user groups, the visualization groups get the same permissions on the controller as the respective groups of the device user management.

The newly created user management groups are now coupled with the device user management and are mapped to the device user management groups. You get their group permissions.

To configure the login operation, see: \({ }^{\sharp}\) Chapter 6.4.5.7.5 "Setting up access control and login operation" on page 2903

\subsection*{6.4.5.7.3 Switch to legacy user management}

Whenever possible, use a runtime version >= 3.5 SP18 Patch 1 and CODESYS Visualization version \(>=4.2 .0 .0\) for a new project with a visualization.

You have initially created a runtime-based user management. Now you want to decouple this from the device user management and have a separate management only for the visualization users.
1. In the Visualization Manager, open the "User Management" tab and click the "Groups" tab.
2. Click the [Convert to Legacy] button.

User management is decoupled from device user management and converted to legacy. The "Users" tab opens.

\subsection*{6.4.5.7.4 Creating a group with administrator rights for visualizations}

Only with runtime-based user management.

\section*{Configuration of user groups with restricted administrator permissions for the visualization.}

Typically, operators of the visualization are a separate user group from application developers. Therefore, the following steps can also be used to configure an administrator for the visualization, who has access only to the groups of the visualization.
This configuration is possible with runtime version 3.5.18.10 and higher.
For detailed information about CODESYS device user management and its operation, see: Handling of Device User Management.
« Chapter 6.4.1.11.4 "Handling of Device User Management" on page 1971
Requirement: A visualization group (example: VISU_ADMIN) has been assigned to a runtime group (example: RTS_GRP_RESTR) in the Visualization Manager on the User Management tab. In the device user management, the User_Restr user has been assigned to the RTS_GRP_RESTR group. The RTS_GRP_RESTR group has full View permissions for the controller.
1. n the device user management, on the "Access Rights" tab, grant the "View" permission for the "UserManagement" object to the "RTS_GRP_RESTR" group. To do this, select the "UserManagement" object, and in the "Rights" area on the right side, double-click the "View" field of the "RTS_GRP_RESTR" group. This opens a dialog prompting whether or not the changed permissions should also be changed for all child objects. Click [No] to refuse this action. Double-click the field again and click [No] again in the next dialog.
\(\Rightarrow\) The \(\bumpeq\) symbol is displayed for the View permission. For all other groups (except Administrator), the permission should have the \(=\) symbol.
2. To the "RTS_GRP_RESTR" group, grant the "View" and "Modify" permissions for the "UserManagement \(\rightarrow\) Users" object.
\(\Rightarrow\) The』 symbol is displayed for both permissions.
3. To the "RTS_GRP_RESTR" group, grant the "View" and "Modify" permissions for those objects of the "UserManagement \(\rightarrow\) Groups" object which are relevant for the visualization.
\(\Rightarrow\) Access to the groups which are relevant for the visualization is granted to the RTS_GRP_RESTR group. For the respective objects, the 亿 symbol is displayed for both permissions.
4. For the "RTS_GRP_RESTR" group, deny the "View" permission to the "Administrator" group. As a result, users of the "RTS_GRP_RESTR" group cannot extend their permissions.
Moreover, select the "UserManagement \(\rightarrow\) Groups \(\rightarrow\) Administrator" object and deny the "View" permission to the "RTS_GRP_RESTR" group.
\(\Rightarrow\) The \(=\) symbol is displayed.
\(\Rightarrow\) Now when you are logged in as a user with restricted administrator permissions on the application, you can create new users or modify existing users for the visualization, but not for groups with additional permissions on the controller.

\subsection*{6.4.5.7.5 Setting up access control and login operation}

In the user management dialogs, you define the login, logout, changing of the user password, and editing of the user management in the visualization at runtime.
Editing and Selecting User Management Dialogs

\section*{NOTICE!}

If you create your own dialog as a user management dialog, then you should use the visualizations from the included library project VisuUserMgmtDialogs as the basis, because it uses the required interfaces. Your own user management dialog is listed then in "Visualization Manager \(\rightarrow\) Settings", "Settings for User Management Dialogs".


For information about the login visualization in runtime-based user management, see: "Creating runtime-based user management initially".

²) Chapter 6.4.5.7.2 "Creating runtime-based user management initially" on page 2900

\section*{Creating Login Pages}
1. If you do not want to configure your own login page, then a preconfigured login page from a library is automatically used when logging in to the visualization. This visualization is automatically entered in the editor of the "WebVisu" as the "Login visualization" when the user administration is created, but it can be changed if necessary.
2. If you want to create your own login page, then follow the steps below:
3. Add a "Visualization" object (for example, named "Login_Visu") to the application.
4. In the context menu of this object, click "Properties". In the properties dialog, open the "Visualization" tab.
5. In the context menu of this object, click "Properties". In the properties dialog, open the "Visualization" tab.
6. Open the "Login_Visu" visualization and design your login page with the input fields for user name, password, and buttons for logging in and canceling the dialog.
7. To do this, only certain visualization elements are available to you. In these visualization elements, you can use placeholders to configure the input fields for the user name and password, as well as the buttons for logging in and canceling the dialog.
8. Compile the application and perform a download to the controller.

When you open the web visualization, the login page which you created for the visualization opens. Type in your user name and the password for the controller which you have stored in the device user management and click the [LOGIN]button.

Editing user Requirement: The library project VisuUserMgmtDialogs.library exists in the installation management dialogs directory.
1. Click "File \(\rightarrow\) Open Project".
2. Select the project VisuUserMgmtDialogs.library from the Projects folder of the installation directory.
3. Click "View \(\rightarrow\) POUs".

In the "POUs" view, the project is displayed with the visualizations "UserMgmtChangePassword", "UserMgmtConfig", and UserMgmtChangePassword.
4. Double-click a visualization (example: "UserMgmtLogin").
5. Change the visualization as you like and save the project.
6. Then, reinstall the library and add it to the "Library Manager" of your application.

Selecting user A user management already exists in your application in the "Visualization Manager" object on management dialogs
the "User Management" tab.
The "VisuUserManagement" library is in the "Library Manager".
1. In the device tree, click the "Visualization Manager" object.
2. Select the "Dialog Settings" tab.
3. In "Settings for User Management Dialogs", select the dialogs for "Login dialog", "Change password dialog", and "Change configuration dialog".

> If no entries can be seen in "Settings for User Management Dialogs" in the dialog lists, then close the "Visualization Manager" and reopen it.

Customizing th
dialogs of the user management
agement dialogs as follows:
Requirement: A visualization is open.
1. Drag a "Button" element from the "Visualization Toolbox" view ("Common Controls" category) to the visualization.
2. In the "Properties" view, click the "Input configuration" node.
3. In the "Input Configuration \(\rightarrow\) OnMouseClick" property, click "Configure".
4. In the "Input Configuration" dialog, click "User management" and
\(\Rightarrow\) The following "Dialogs and actions" are listed on the right: "Login, Logout, Change User Password", and "Open User Configuration".
5. elect the dialog or action to assign to the button and click [OK].
\(\Rightarrow\) When the button is clicked at runtime, the selected dialog opens or the selected action is executed.

If you want to open and edit the user management in the visualization at runtime, you have to be a member of a group that has P"ermission to Change User Data".

\subsection*{6.4.5.7.6 Group-dependent restriction of functionality}

Configuring per- Permissions for visualization-elements are not granted to individual users, but to groups with missions for an assigned users. element of the visualization
1. Click a visualization element in the editor.
2. Click the "Value" field of the "Permissions" element property in the "Properties" view. \(\Rightarrow\) The "Permissions" dialog opens.
3. Select the permissions that the respective user group should have for the visualization element.

If the option "Group hierarchy is used" is activated, the groups lower in the hierarchy cannot be granted more permissions than groups higher in the hierarchy.
\begin{tabular}{ll}
\begin{tabular}{l} 
Updating \\
groups and \\
group permis-
\end{tabular} & \begin{tabular}{l} 
In the project below the application, a visualization is stored which has at least one element with \\
restricted permissions.
\end{tabular} \\
sions
\end{tabular}
1. Open the Visualization Manager and click the "User Management" tab.
2. Click the "List Usage of Groups" button.
\(\Rightarrow\) In "Messages", all elements with restricted permissions are listed which exist in the visualizations that are used across the application. This will give you an overview of functional limitations.

\section*{3. Click the [Update Visualizations/Hotkeys] button.}
\(\Rightarrow\) The "Update Visualizations and Hotkeys" dialog opens. Here you can centrally rename groups, delete groups, or change group permissions. The affected visualization elements are updated. This means that the "Access Rights" property is updated for each of the affected elements: There, the group names are removed or renamed or the permissions are reset.

\subsection*{6.4.5.8 Setting Up Multiple Languages}

Texts and tooltip texts for visualizations are managed in text lists and can be displayed in different languages. To switch a visualization between the available languages, configure a visualization element with the corresponding input configuration for changing the language.
There are static texts that are managed in "GlobalTextList" (generated automatically) and dynamic texts from created text lists. A dynamic text can be changed at runtime with a variable that defines the index of the text list entry. Static texts are fixed labels within a visualization; dynamic texts are often used for displaying variable values or error messages.
For creating and using text lists, see: \({ }^{\Downarrow}\) Chapter 6.4.1.9.10 "Managing text in text lists" on page 1891.


You can modify the appearance and formatting of texts and tooltips with the element properties "Text properties" and "Font variables".

See also
- \({ }^{\text {® }}\) Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329
- \(\xi^{2}\) Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration'" on page 3370

Configuring language switching for texts from text lists

Requirement: An empty visualization object is inserted into the project and it is open for editing in the visualization editor. There is also a "Visualization Manager" object. User management is not created for the visualization.
The following instructions provide a simplified example:
- By means of two buttons, the user should be able to toggle the visualization texts between English and German.
- Static texts in the visualization include the labels "State, Machine 01", "State, Machine 02", "English", and "German". These texts are located in the "GlobalTextList" in English and German.
Dynamic texts will describe the state of both machines. The texts are provided in the text list "Status_Texts" in English (en) and German (de).
1. Drag a "Text Field" from the "Visualization Toolbox" view ("Common Controls" category) to the editor view. Specify the value State, Machine M01 in the properties editor for the element property "Texts \(\rightarrow\) Text".
2. Copy the element and change the copy label to State, Machine M02.
3. See also the figure in step 14 for the following steps.

Insert two elements of type "Button" from the "Visualization Toolbox" view ("Common Controls" category) in the visualization editor. With these elements, the user should be able to toggle the language of the visualization. Specify the text German or English in the properties editor for element property "Texts \(\rightarrow\) Text" (4).
4. Double-click and open "GlobalTextList" in the "POUs" view.
\(\Rightarrow\) The texts are entered in the "Standard" (1) column, and the "ID"s 0 and 1 are assigned automatically as additional information.
5. Add the languages "de" and "en" with the texts shown in the following figure.
\begin{tabular}{|c|c|c|c|c|}
\hline \multicolumn{4}{|l|}{国 GlobalTextList \(\mathbf{x}\)} & - \\
\hline & Default (1) & de & en & \\
\hline 3 & English & Englisch & English & \\
\hline 2 & German & Deutsch & German & \\
\hline 0 & State, Machine M01: & Zustand Maschine M01: & State, Machine M01: & \\
\hline 1 & State, Machine M02: & Zustand Maschine M02: & State, Machine M02: & \\
\hline
\end{tabular}
6. Close the "GlobalTextList".
7. Add two elements of type "Rectangle" from the "Visualization Toolbox" view ("Basic" category) in the visualization editor. The current state of each machine should be displayed.
8. For managing the texts for describing the states, add an object of type "Text List" below the application. Name the list Status_Texts.
9. Specify the texts shown in the figure for the standard language (1) and the target languages "en" and "de" in the editor of "Status_Texts".
\begin{tabular}{|l|l|l|l|}
\hline \multicolumn{8}{|c|}{ Stas_Texts \(\mathbf{x}\)} & & \\
\hline ID & Default & de & en \\
\hline 0 & running & läuft & running \\
\hline 1 & stopped & lauft nicht & not running \\
\hline 2 & starting & läuft gerade hoch & starting \\
\hline 3 & error & Fehlermeldung! & error message! \\
\hline
\end{tabular}
10. Close the text list "Status_Texts".
11. Select the rectangle element for displaying the state of machine M01. Select the text list Status_Texts from the combo box in the properties editor (2) for the "Dynamic texts" element property (5). Specify an application variable for "Text index" that shows the appropriate text index for the state of the machine at runtime. Example: PLC_PRG.ivar_status_m01.
12. Now configure the user input for both buttons for toggling the language in the visualization.

Select the "German" button. Double-click "Configure" of the property "Input configuration" (6), "OnMouseClick".
\(\Rightarrow\) The "Input configuration / OnMouseClick" dialog opens.
13. Select "Change the language" on the left. Click the arrows to accept the setting to the right. Select "de" in "Language" to the right of the dialog in the ... input assistance. Click "OK" to confirm.
14. In the same way, configure the entry for the second button: English: "Text list": Status_Texts, "Text index": 3, "Change the language": English).
\(\Rightarrow\) The following figure shows the performed properties configurations for the four visualization elements.

15. When the application is compiled without errors, you can test the visualization in simulation mode. Activate the option "Online \(\rightarrow\) Simulation". Click "Online \(\rightarrow\) Login".
\(\Rightarrow\) The visualization appears in the visualization editor view in online mode:

16. Click the "German" button.
\(\Rightarrow\) The language changes to German:


\footnotetext{
«» "Input action 'Change Language"" on page 3372
}
Setting up fonts The font for a visualization element is defined in the properties editor. If a language switch is for a language provided, you can overwrite this basic font with another font for each language in the visualization manager.
Requirement: A visualization is set up with at least one language in addition to the default language.

\section*{. Example}
1. Double-click and open the "Visualization Manager" object and select the "Font" tab.
2. Double-click the field in the "Font" line for a particular language. Select a font from the combo box.
3. In the "Font size" line, replace the value 1 with a value greater than 1 (example: 2 ) in order to increase the size of the font as defined by the visualization style; or replace it with a value less than 1 in order to decrease is (example: 0.5 ).
\(\Rightarrow\) In online mode, the font changes depending on the set language.

③ Chapter 6.4.5.21.4.6 "Tab 'Visualization Manager' - 'Font"' on page 3407

\subsection*{6.4.5.9 Visualizing alarm management}

In CODESYS, the alarm management is a powerful object for creating and managing alarms. You can group alarms and set the acknowledgement behavior individually. The alarm display in the visualization can also be customized.

The "Alarm Table" and "Alarm Banner" visualization elements are available for displaying and processing alarms. The alarm table lists the alarm texts. The alarm banner is a simplified version of the alarm table. It visualizes a single alarm only. However, by adding scroll elements you can allow for switching the display from one active alarm to another active alarm.
\& Chapter 6.4.1.9.20 "Alarm Management" on page 1928

Creating an Requirement: In your project, alarms are defined in alarm groups and they are assigned to alarm table an alarm class. The following instructions are based on the example that is described in the "Configuring alarm management" chapter.
1. Open the visualization editor.
2. Drag the "Alarm Table" element from the "Alarm Manager" group to the visualization editor.
\(\Rightarrow\) The "Alarm Table" visualization element is visible in the editor.
3. In the "Alarm configuration" / "Alarm groups" property, define the alarm groups that you want to visualize. Click into the value field.
\(\Rightarrow\) The "Select Alarm Group" dialog opens.
4. Clear the "All" check box and select the "PartsDeficit" alarm group. Add the group to the selected alarm groups by clicking the \(>\) button.
5. In the "Alarm configuration"/ "Alarm classes" property, define the alarm classes that you want to visualize. Click into the value field.
\(\Rightarrow\) The "Select Alarm Class" dialog opens.
6. Clear the "All" check box and select the "PartsDeficit" alarm class. Add the alarm class to the selected alarm classes by clicking the \(\triangle\) button.
7. Add an additional column. Click the "Columns" / "Create New" button.
\(\Rightarrow\) CODESYS adds the column "[2]" to the properties. The "Symbol" column is added to the table.
8. Select data type "State" for column [2].
\(\Rightarrow\) The default column heading "State" is shown in the table.
9. Name the "Column heading" column "Status".
10. Specify the appearance of the selected table cell. Set the "Selection" / "Selection color" to "Green".
11. In the "Control variables" / "Confirm selection" property, specify the variable bQuitAlarm for confirming messages.
12. Adjust the other properties to your requirements. See the "Alarm table" visualization element for a complete description of the properties.
« Chapter 6.4.5.20.1.22 "Visualization Element 'Alarm Table"" on page 3166

Inserting elements for acknowledging alarms

In CODESYS, predefined buttons are available for controlling the alarms in an alarm table.
Requirement: An "Alarm table" element exists in the visualization.
1. Select the visualization element in the editor.
2. Click "Visualization \(\rightarrow\) Insert elements for acknowledging alarms".
\(\Rightarrow\) The "Alarm Table Wizard" dialog opens.
3. Click "OK" to accept all settings.
\(\Rightarrow\) Four buttons are added for controlling the alarm table.


\section*{③ Chapter 6.4.5.20.1.22 "Visualization Element 'Alarm Table"" on page 3166}

\section*{Creating an alarm banner}

Requirement: In your project, alarms are defined in alarm groups and they are assigned to an alarm class. The following statement is based on the example that is described in the "Configuring alarm management" chapter.
The alarm banner displays an active alarm in online mode. If there are multiple active alarms, filtering takes place by means of the filter criteria set in the alarm banner (newest for filter criterion "Priority" and most important for filter criterion "Newest"). See the instructions below for adding scroll elements in order to switch the display between multiple alarms.
1. Open the visualization editor.
2. Drag the "Alarm banner" element from the "Alarm manager" group to the visualization editor.
\(\Rightarrow\) The "Alarm banner" visualization element is visible in the editor.
3. In the "Alarm configuration" / "Alarm groups" property, define the alarm groups that you want to visualize. Click into the value field.
\(\Rightarrow\) The "Select Alarm Group" dialog opens.
4. Clear the "All" check box and select the "PartsDeficit" alarm group. Add the group to the selected alarm groups by clicking the \(\triangle\) button.
5. In the "Alarm configuration"/ "Alarm classes" property, define the alarm classes that you want to visualize. Click into the value field.
\(\Rightarrow\) The "Select Alarm Class" dialog opens.
6. Clear the "All" check box and select the "PartsDeficit" alarm class. Add the alarm class to the selected alarm classes by clicking the \(\square\) button.
7. Set the "Alarm configuration" / "Filter criterion" property to "Newest".
\(\Rightarrow\) In online mode, the newest alarm message is always shown.
8. Add an additional column. Click the "Columns"/ "Create new" button.
\(\Rightarrow\) CODESYS adds the column "[2]" to the properties. The "Symbol" column is added to the table.
9. Select data type "State" for column [2].
\(\Rightarrow\) The default column heading "State" is shown in the table.
10. In the "Confirmation variable" property, specify the variable bQuitAlarm for confirming messages.
* \({ }^{4}\) Chapter 6.4.5.20.1.23 "Visualization Element 'Alarm Banner"" on page 3175


Elements can be added to an alarm banner for switching the display between the individual active alarms. You can control the scrolling with visu-local variables or application variables.
1. Select the added "Alarm banner" visualization element. Click "Insert Elements for Scrolling Alarms" in the context menu.
\(\Rightarrow\) The "Alarm Banner Wizard" opens.
2. Select the element type for the scroll elements: "Button" or "Rectangle".
3. Activate the action(s) for which a control should be inserted: "Scroll to next alarm", "Scroll to previous alarm".
4. Specify a Boolean variable that gets the value TRUE when multiple active alarms are present. If you have already configured a project variable in the element properties, then it is also specified here in the wizard. Otherwise CODESYS automatically creates the visu-local variable "xMultipleAlarmsActive".
5. In the next step, check the configuration of the element properties of the extended alarm banner.
6. Select the alarm banner element and look at the section "Handling of multiple active alarms" in the "Properties" view. You have two options:
7. Option 1: The display should switch automatically. Activate the "Switch automatically" property.
\(\Rightarrow\) Now, in "Every \(N\) seconds" you define the time interval after which the display in the alarm banner in online mode should switch to the next alarm.
8. Option 2: The display should be controlled by means of the application. Deactivate the "Switch automatically" property.
\(\Rightarrow\) Switching between the active alarms can be controlled by two variables. By default, xNext and xPrev are created for scrolling to the next or previous alarm. You can replace these variables with custom your own defined application variables.

Filter alarm Filtering by the contents of a latch variable can be useful when there are a lot of alarm events events by the contents of the latch variable displayed. If the latch variable assigned to an alarm in the alarm group definition contains, for example, the error number or the name of a device instance, then you can filter the alarms in the visualization by it.

For this purpose, you configure an input option in the alarm visualization for the contents of the latch variable to be filtered by. For example, insert an input field which writes to the variable that is specified in the "Alarm configuration" - "Filter by latch 1" - "Filter variable" property of the configuration of the "Alarm table" element or "Alarm banner" element.
In addition, you configure an input option for the type of filtering. The type determines whether a numeric value (typed literal, LINT literal) or the string value of the latch variable is used for filtering. Filtering can also be switched off by means of type setting 0 . For example, in the visualization, insert another input field which writes to the variable that is specified in the "Filter type" property of the configuration of the alarm table or alarm banner.

For more information, see the "Alarm Filter Latch Example" sample project in the CODESYS Store.

\subsection*{6.4.5.10 Animating visualization elements}
\(\qquad\)
6.4.5.10.2 Configuring rotations and offsets.................................................. 2912
6.4.5.10.3 Animating a text display............................................................... 2914
6.4.5.10.4 Animating a color display............................................................. 2914

\subsection*{6.4.5.10.1 General}

The animation of a visualization element at runtime can serve to visualize value curves in addition to serving purely visual purposes. Animation is possible through a dynamic configuration of certain element properties, i.e. by controlling these properties with a variable. See the following examples of possible animations.

\subsection*{6.4.5.10.2 Configuring rotations and offsets}

You can animate a visualization element and have it shifted or rotated at runtime. To do this you assign variables in its property "Absolute movement" and then program the animation in the application code.

Configuring an You can configure an offset of the element by programming the variables in
offset
\(\checkmark\) Requirement: A project with a visualization is open.
1. Open the visualization and add an element "Rectangle".
\(\Rightarrow\) The view "Properties" displays the configuration of the element.
2. In the application in the POU PLC_PRG, declare type-compliant variables: diOffsetX : DINT; and diOffsetY : DINT;
3. Configure the property "Absolute movement \(\rightarrow\) Movement \(\rightarrow X\) " with PLC_PRG.diOffsetX and "Y" with PLC_PRG.diOffsetY.
4. Implement a shift of the element, for example by means of a modulo division of the value:
diOffsetX := diOffsetX MOD 100;
diOffsetY := diOffsetY MOD 100;
5. Compile, load and start the application.
\(\Rightarrow\) The application runs. The visualization opens. The rectangle moves.

Configuring a When an element rotates, then the center point of the element rotates precisely around its rotating element center. The center is defined in the property "Center". The center point of an element is calculated internally. If the center point and center coincide, then there is no rotation.
You can configure a clockwise rotation of the element by increasing the value of the variable "Absolute movement \(\rightarrow\) Rotation".
\(\checkmark \quad\) Requirement: A project with a visualization is open.
1. Open the visualization and add an element "Rectangle".
\(\Rightarrow\) The view "Properties" displays the configuration of the element.
2. In the application in the POU PLC_PRG, declare a type-compliant variable:rValue : REAL;
3. Configure the property "Absolute movement \(\rightarrow\) Rotation" with PLC_PRG.rValue.
4. Implement the clockwise rotation of the element by increasing the value of the variable:
```

rValue := rValue + 0.1;

```
5. Compile, load and start the application.
\(\Rightarrow\) The application runs. The visualization opens. The rectangle rotates about the center. The alignment of the element with respect to the coordinate system is fixed.


Configuring a When an element performs an inner rotation and rotates, then the center point of the element rotating element rotates precisely around its center. This is the point defined in the property "Center". The alignment of the element also rotates relative to the coordinate system. If the center point of the element and the center coincide, this produces a rotation on the spot.
You can configure a clockwise rotation of the element by increasing the value of the variable "Absolute movement \(\rightarrow\) Inner rotation".
If the visualization is In runtime, you can see that the element rotates (also relative to the coordinate system of the visualization).
\(\square\) Requirement: A project with a visualization is open.
1. Open the visualization and add an element "Polygon", which you form into a pointer. \(\Rightarrow\) The view "Properties" displays the configuration of the element.
2. Drag the center point of the element to the base of the pointer.
3. In the application in the POU PLC_PRG, declare a type-compliant variable:
```

rValue : REAL;

```
4. Configure the property "Absolute movement \(\rightarrow\) Inner rotation" with PLC_PRG.rValue.
5. Implement the clockwise rotation of the element by increasing the value of the variable:
rValue := rValue + 0.1;
6. Compile, load and start the application.
\(\Rightarrow\) The application runs. The visualization opens. The pointer rotates about its base.

\(\boldsymbol{y}^{\text {C }}\) Chapter 6.4.5.20.1.1 "Visualization Element 'Rectangle', 'Rounded Rectangle', 'Ellipse"" on page 2988

\subsection*{6.4.5.10.3 Animating a text display}

An animation of the text display can be configured in the property "Font variables". All basic elements have this property as well as tables, scrollbars and text fields.

\section*{Example: animating the font size}
\(\square\) Requirement: A project with a visualization is open.
1. Open the visualization and add an element "Rectangle".
\(\Rightarrow\) The view "Properties" displays the configuration of the element.
2. Configure the property "Texts \(\boldsymbol{\rightarrow}\) Text" with Important:
3. In the application in the POU PLC_PRG, declare a type-compliant variable:
```

| iFontHeight : INT;

```
4. Configure the property "Font variables \(\boldsymbol{\rightarrow}\) Size" with PLC_PRG. iFontHeight.
5. Implement a change of the font size.
\(\Rightarrow\) iFontHeight:= iFontHeight + 1) MOD 20;
6. Compile, load and start the application.
\(\Rightarrow\) The application runs. The visualization opens. The rectangle is labelled with Important. The font size grows from 1 to 20 .

See also
- «s Chapter 6.4.5.20.1.1 "Visualization Element 'Rectangle', 'Rounded Rectangle', 'Ellipse"' on page 2988

\subsection*{6.4.5.10.4 Animating a color display}

The colors of an element are specified in the "Colors" properties of the element properties. There you can select either a predefined style color from the selection list or a color in the color dialog.

The "Color variables" element property is used for the color animation of the element. If you pass variables to the properties, then you can program color changes in the application code or configure a user input that results in a color change. A color constant or color variable in the code has the data type DWORD and is encoded according to the RGB color space or RGBA extension.

\section*{Color definition in RGBA color space}

Animating a visualization element in color
1. Create a standard project in CODESYS.
2. Declare global color constants in the POU tree.
\(\Rightarrow\)
```

{attribute 'qualified_only'}
VAR_GLOBAL CONSTANT
gc_dwRed : DWORD := 16\#FFFF0000;
gc_dwGreen: DWORD := 16\#FFO0FF00;
gc_dwYellow: DWORD := 16\#FFFFFF00;
gc_dwBlue: DWORD := 16\#FF0000FF; // Highly opaque
gc_dwBlack : DWORD := 16\#88000000; // Semitransparent
END_VAR

```
3. In the device tree, declare local color variables in PLC_PRG.
```

=>
VAR
dwFillColor: DWORD := GVL.gc dwGreen;
dwFrameColor : DWORD := GVL.g
dwAlarmColor : DWORD := GVL.gc_dwRed;
END_VAR

```
4. Declare a control variable.
```

b bChangeColor : BOOL;

```
5. Declare an input variable in PLC_PRG.
```

=> bInput : BOOL;

```
6. Enable the visualization editor.
7. Drag a "Rectangle" element to the visualization editor. \(\Rightarrow\) The "Properties" view of the element opens.
8. Configure the properties of the rectangle as follows:
- Property "Color variables", "Normal state", "Filling color": PLC_PRG.dwFillColor
- Property "Color variables", "Normal state", "Frame color": PLC_PRG.dwFrameColor
- Property "Color variables", "Alarm state", "Filling color": PLC_PRG. dwAlarmColor
- Property "Color variables", "Toggle color": <toggle/tap variable>
- Property "Input configuration", "Toggle", "Variable": PLC_PRG.bInput
9. Program the variables as follows:
```

PROGRAM PLC_PRG
VAR
dwFillColor: DWORD := GVL.gc_dwGreen;
dwFrameColor : DWORD := GVL.gc_dwBlack;
dwAlarmColor : DWORD := GVL.gc_dwRed;
bChangeColor : BOOL;
bInput : BOOL;
END_VAR
IF bChangeColor = TRUE THEN
dwFillColor := GVL.gc_dwYellow;
dwFrameColor := GVL.gc_dwBlue;
ELSE
dwFillColor:= GVL.gc_dwGreen;
dwFrameColor := GVL.g}c_dwBlack
END_IF

```
\(\Rightarrow\) The colors are initialized at runtime. If the variable bChangeColor is then forced to TRUE, the color display of the rectangle changes. When the rectangle is clicked in the visualization, the rectangle is displayed in alarm colors.

See also
- ⓨ Chapter 6.4.5.10 "Animating visualization elements" on page 2912
- ⓨ Chapter 6.4.5.21.4.2 "Object 'Visualization manager'" on page 3398
- \(\quad\) H Chapter 6.4.5.19 "Applying Visualization Styles" on page 2979
- \(\boldsymbol{y}^{2}\) Chapter 6.4.5.10.3 "Animating a text display" on page 2914
6.4.5.11 Displaying data arrays in tables
\[
\text { 6.4.5.11.1 General.......................................................................................... } 2917
\]
6.4.5.11.2 Displaying Array Variables in Tables.............................................. 2917
6.4.5.11.3 Configuring and Multiplying Visualization Elements as Templates 2918

\subsection*{6.4.5.11.1 General}

A frequently required function of a user interface is the display of data arrays. CODESYS Visualization provides the element "Table" for this.
In the configuration of the element "Table", enter an array variable in the property "Data array". The array components are displayed in the rows and columns of the table.
A table for displaying data arrays can also be created in the following way. You duplicate a single element having at least one property that is described by a structured variable. The single element is configured as a "template" for this and duplicated with a command.

\subsection*{6.4.5.11.2 Displaying Array Variables in Tables}

A frequently required function of a user interface is the display of data arrays. CODESYS Visualization provides the element "Table" for this.
In the configuration of the element "Table", enter an array variable in the property "Data array". The array components are displayed in the rows and columns of the table.

Subsequent instructions describe an example of how an array of a structure is displayed in a table. As a preparation, create the MYSTRUCT DUT and the declarations in the PLC_PRG program.
```

TYPE MYSTRUCT :
STRUCT
iNo : INT;
bOnStock : BOOL;
strPartNumber : STRING;
END_STRUCT
END_TYP\overline{E}
PROGRAM PLC_PRG
VAR
arrStruct : ARRAY[0..6] OF MYSTRUCT;
iSelectedColumn : INT;
END_VAR

```
1. Drag the "Table" visualization element to the visualization editor.
2. Assign the array variable arrStruct to the "Data array" property.
\(\Rightarrow\) The structure members are displayed as column headings and the array index as row headings.
3. Change the "Columns \(\rightarrow\) Column \(\rightarrow[0] \rightarrow\) Column header" property to an informative heading (example: Number).
4. Change the heading of column [1] to in stock and column [2] to Part number. Adjust the column width.
5. Assign a color to the "Selection \(\rightarrow\) Selection color" property.
6. Define the "Selection \(\rightarrow\) Selection type" property as Row selection.
7. In the "Selection \(\rightarrow\) Variable for selected row" property, define the PLC_PRG.iSelectedColumn variable.
\(\Rightarrow\) The following display results in online mode:
\begin{tabular}{|c|c|c|c||}
\hline & Number & in Stock & PartNumber \\
\hline 0 & 0 & FALSE & \\
\hline 1 & 0 & FALSE & \\
\hline 2 & 0 & FALSE & \\
\hline 3 & 0 & FALSE & \\
\hline
\end{tabular}

See also
- \({ }^{n} \boldsymbol{y}\) Chapter 6.4.5.20.1.13 "Visualization Element 'Table"" on page 3106

\subsection*{6.4.5.11.3 Configuring and Multiplying Visualization Elements as Templates}

A table can also be created to display data arrays in the following way. You multiply a single element that has at least one property which is described by a structured variable. To do this, the single element is configured as a "template" and multiplied by means of a command.

You can use the "Visualization \(\rightarrow\) Multiply Visu Element" command to display array data in a visualization. The command multiplies a template element to create an element of the same type for each array component. The layout of the new elements in the visualization is one-dimensional as a row or column, or two-dimensional as a table.

To do this, drag an applicable element into the visualization editor. Then configure the properties of the element with array variables and specify the index access placeholder \$FIRSTDIM\$ as component access. If you have declared a multidimensional array, then you can use the second index access placeholder \$SECONDDDIM\$ for the additional dimension. Configure the remaining properties as usual with the typical values. The purpose is to create a valid template element. Then execute the "Multiply Visu Element" command on the template element. Now the dialog with the same name opens. There you define in detail how many elements should be created and where they should be located.
After multiplying, the visualization contains as many of the same elements as are indexed using placeholders. In doing so, the settings in the "Multiply Visu Element" dialog are taken into consideration. All new elements in the properties that were preset with placeholders have these replaced with precise indexes. The remaining properties have been applied and copied without changes.

For example, you can have a layout of nine buttons as \(3 \times 3\) tables, which are all the same size or the same color, but vary in the labeling. The labels are declared as a string array (nine components) and are passed as a value to the "Texts"->"Text" property.
Valid template element:
- Declaration of array variables

Example: asText: ARRAY[1..3, 1..3] OF STRING;
- Element with applicable element type
- Configuration of at least one property of the applicable element with array variables with index access placeholders
Example: Property "Texts", "Text" = PLC_PRG. asText[\$FIRSTDIM\$, \$SECONDDIM\$]
This is possible for all properties that permit a variable as a value (for example, also properties from the "Animation" or "Input" categories. To configure multiple properties for an element with arrays and index access placeholders, all arrays must have the same structure with the same dimension. The declarations have to be compatible.
- Configuration of properties that do not vary (and are therefore the same for all generated elements) with the usual values without index access placeholders Example:
```

sButtonTip : STRING := 'This element is created by multiplication'

```
Property "Texts", "Tooltip" = \% s
Property "Text variables", "Tooltip variable" = sButtonTip

You can still use the placeholder \% as usual for the text display of variable values in the properties in "Texts".

Applicable visu- Visualization elements that can be multiplied: alization elements
- "Rectangle"
- "Rounded Rectangle"
- "Ellipse"
- "Line"
- "Polygon"
- "Polyline"
- "Bézier Curve"
- "Image"
- "Frame"
- "Button"
- "Pie"
- "Spin Box"
- "Text Field"
- "Check Box"
- "Image Switcher"
- "Lamp"
- "Dip Switch"
- "Power Switch"
- "Push Switch"
- "Push Switch LED"
- "Rocker Switch"
- "Rotary Switch"

\section*{Configuring and multiplying lamps and buttons as templates}
1. Create a new standard project.
\(\Rightarrow\) A CODESYS Control Win V3 is configured as the device. The MainTask calls PLC_PRG. The implementation language is ST.
2. In PLC_PRG in the program code, declare array variables with basic data type STRING.
\(\Rightarrow\)
PROGRAM PLC_PRG
VAR
axLampIsOn: ARRAY[1..2,1..3] OF BOOL; // For lamp, property 'variable' and button, user input
asButtonText: ARRAY[1..2,1..3] OF STRING := // Output text for button, property 'text variables''text variable'
[
'1A Lamp', '2A Lamp',
'1B Lamp', '2B Lamp',
'1C Lamp', '2C Lamp'
];
END_VAR
3. Select the application in the device tree and click "Add Object \(\rightarrow\) Visualization".
4. In the "Add Visualization" dialog, specify the name VisuMain and click "Add" to close the dialog.
5. Drag a "Lamp" element from the "Visualization Toolbox" view to the visualization.

6．Configure the fixed property values．
\begin{tabular}{|c|c|c|c|c|}
\hline 明 Visualization \(\times\) & ＊ & \multicolumn{2}{|l|}{Properties} & \(-4 \times\) \\
\hline \multirow[t]{16}{*}{} & A & \multicolumn{2}{|l|}{マ Filter＊＊Sort by＊ \(\mathrm{A} \downarrow\) Sort order＊\(\square\) Advanced} & \\
\hline & & Property & Value & \\
\hline & 三 & Element name & Array of lamps & \\
\hline & & Type of element & Lamp & \\
\hline & & －Position & & \\
\hline & & X & 50 & \\
\hline & & Y & 20 & \\
\hline & & Width & 70 & \\
\hline & & Height & 70 & \\
\hline & & Variable & & \\
\hline & & －Texts & & \\
\hline & & Tooltip & & \\
\hline & & －State variables & & \\
\hline & & Invisible & & \\
\hline & & －Background & & \\
\hline & & Image & Yellow & \\
\hline
\end{tabular}

7．Double－click the value field of the＂Variable＂property．
\(\Rightarrow\) The line editor opens．
8．Click ．．．．
\(\Rightarrow\) The Input Assistant opens．
9．Select the array variable PLC＿PRG．axLampIsOn from the variable tree．
10．Extend the string at the end，for example with＂［ f ＂．
\(\Rightarrow\) If you have activated SmartCoding（＂Options＂dialog，＂SmartCoding＂category，＂List components immediately when typing＂option），then the current variable list appears with the placeholders：


11．Select the placeholder \＄FIRSTDIM\＄for the first dimension and confirm the selection．
12．Extend the string at the end，for example with＂，s＂．
\(\Rightarrow\) The variable list appears again．
13．Select the placeholder \＄SECONDDIM\＄for the second dimension and confirm the selection．
14．Complete the string with a closing bracket．
```

A PLC_PRG.axLampIsOn[$FIRSTDIM$, $SECONDDIM$]

```

The lamp is configured as a template．
15. Click "Visualization \(\rightarrow\) Multiply Visu Element".
\(\Rightarrow\) The "Multiply Visu Element" dialog opens. The default values are derived from the array declarations.
"Total number of elements", "Horizontal" = 2
"Total number of elements", "Vertical" = 3
16. Declare the distance between the new elements.
\(\Rightarrow\) "Offset between elements", "Horizontal" = 3
"Offset between elements", "Vertical" = 3
17. Check the advanced settings.
18. Click "OK" to confirm the selection.
\(\Rightarrow\) The new elements appear in the visualization editor. All properties are configured with a precise index and the array variables are indexed.
19. In the "Visualization Toolbox", in the "Common Controls" category, drag the "Button" element to the visualization editor.
\(\Rightarrow\) The "Properties" view of the element opens.
20. Configure the fixed property values.

21. Configure the value for the "Text variables"->"Text variable" property.
```

=> PLC_PRG.asButtonText[$FIRSTDIM$, $SECONDDIM$]

```
22. Configure the value for the "Input configuration"->"Toggle"->"Variable" property.
```

A PLC_PRG.axLampIsOn[$FIRSTDIM$, $SECONDDIM$]

```

The button is configured as a template.
23. Click "Visualization \(\rightarrow\) Multiply Visu Element".
\(\Rightarrow\) The "Multiply Visu Element" dialog opens. The default values are derived from the array declarations.
"Total number of elements", "Horizontal" \(=2\)
"Total number of elements", "Vertical" \(=3\)
24. Declare the distance between the new elements.
\[
\begin{aligned}
\Rightarrow & \text { "Offset between elements", "Horizontal" = } 3 \\
& \text { "Offset between elements", "Vertical" }=3
\end{aligned}
\]
25. Check the advanced settings.
26. Click "OK" to confirm the selection.
\(\Rightarrow\) The new elements appear in the visualization editor. All properties are configured with a precise index and the array variables are indexed.

27. Build, start, and download the application.
\(\Rightarrow\) Visualization at runtime:


Array variable You can also configure the template element with array variables that have more than two with more than dimensions, but you can only assign placeholders to a maximum of two of the dimensions. In two dimensions the additional dimensions, the indexes are fixed.
```

Example Declaration

```
```

PROGRAM PLC_PRG

```
PROGRAM PLC_PRG
VAR
VAR
    asText: ARRAY[1..2, 1..3, 1..6, 1..2] OF STRING;
    asText: ARRAY[1..2, 1..3, 1..6, 1..2] OF STRING;
    END_VAR
    END_VAR
Configure the "Text variables", "Tooltip variable" property for the template element:
PLC_PRG.asText[2, $FIRSTDIM$, $SECONDDIM$, 2]
```

Layout of a one- You can configure the template element with a one-dimensional array by means of the index dimensional array in a table access placeholder \$FIRSTDIM\$. If the number of new elements to be created is greater than five, then a tabular layout is preset in the "Multiply Visu Element" dialog. The layout of the new elements is as quadratic as possible.

## Example

```
PROGRAM PLC_PRG
VAR
    asText: ARRAY[1..100] OF STRING;
END_VAR
```

The default setting in the "Multiply Visu Element" dialog allows for a layout of 100 new elements in a $10 \times 10$ field.

See also

- ${ }^{2} \gg$ Chapter 6.4.5.21.2.11 "Command 'Multiply Visu Element"' on page 3350
- ${ }^{\text {• }}$ Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329
- Options for SmartCoding


### 6.4.5.12 Displaying data curve with trace

### 6.4.5.12.1 General

With this element, you can integrate a trace graph in the visualization that monitors and displays variable values permanently. You configure the displayed trace graph in the element properties. In addition, you can add control elements that control the trace functionality. This is done manually or by using the "Insert Elements for Controlling Trace" command.


Configurations for the 'Trace' visualization element can be taken from the 'Trace' object.


See also

- $\quad$ Chapter 6.4.5.12.2 "Getting started with trace" on page 2926
- $y^{*}$ Chapter 6.4.5.21.2.15 "Command 'Insert Elements for Controlling Trace"' on page 3358
- $\Leftrightarrow$ Chapter 6.4.1.13.3 "Data Recording with Trace" on page 2007


### 6.4.5.12.2 Getting started with trace

## Create a project

with the fol-
PROGRAM PLC_PRG
lowing program VAR
PLC_PRG:
iVar : INT;
rSin : REAL;
rVar : REAL;
END_VAR

```
iVar := iVar + 1;
iVar := iVar MOD 33;
rVar := rVar + 0.1;
rSin := 30 * SIN(rVar);
```

1. In the device tree, select the application and add a new visualization by clicking "Project $\rightarrow$ Add Object $\rightarrow$ Visualization".
$\Rightarrow$ The respective visualization editor opens.
2. Add the "Visualization" object to the device tree below "Application".
$\Rightarrow$ An empty visualization appears.
3. Open "Toolbox $\rightarrow$ Special Controls".
4. Drag the "Trace" element to the visualization editor.
$\Rightarrow$ The element properties are displayed on the right side.
5. Click the symbol in the "Trace" property.
$\Rightarrow$ The "Trace Configuration" dialog opens.
6. Click "Add Variable" to add an entry to the tree view of the trace configuration and select a project variable (for example, PLC_PRG.rSin).
7. Click the top node of the trace configuration.
$\Rightarrow$ The group "Record Settings" is shown on the right.
8. Select the MainTask option for the "Task" setting.

Tip: The trace recording and the corresponding program should be executed in the same task.
9. Click "OK".
$\Rightarrow$ The task configuration is applied.
10. Select the trace element and click "Visualization $\boldsymbol{\rightarrow}$ Add Elements for Trace Control"
$\Rightarrow$ The "Trace Wizard" dialog opens. By default, all control elements are activated there.
11. Click " $O K$ " to close the dialog.
$\Rightarrow$ The control elements are added to the visualization and the control variables are declared. Then the control elements and the trace element are configured with the control variables.
12. Download the application to the controller and start it.

## Example



The PLC PRG program is running on the PLC. When you follow the "Getting Started" instructions, the following interface is displayed:


You can control the trace recording by clicking the buttons.

See also

- ${ }^{*}>$ Chapter 6.4.5.21.2.13 "Command 'Configure Trace"" on page 3355
- ${ }^{*}$ - Chapter 6.4.5.21.3.19 "Dialog 'Display Settings"' on page 3391
- ${ }^{\text {• }} \boldsymbol{y}$ Chapter 6.4.5.20.1.34 "Visualization Element 'Trace"" on page 3240


### 6.4.5.13 <br> Displaying data curve with trend

6.4.5.13.1

General
A trend visualizes data that is used in the database of a trend recording. In contrast to the trace element, the trend element is particularly appropriate for long-term data recording.


The visualization of a trend encompasses the Trend element and the control elements. The three possible control elements can be seen in the illustration.

- Legend (1): Outputs the trend variables with values.
- Time range picker (2): Provides buttons for selecting predefined time ranges.
- Date range picker (3): encompasses control elements for navigation and zooming in the historical and current data on basis of the set date range.
A cursor is optionally available that enables the reading of a value at a certain time.
You can execute a trend visualization in the following clients:
- Target visualization
- Integrated visualization

See also

- \# Chapter 6.4.5.13.2 "Getting Started with Trend Visualization" on page 2928
- ${ }^{\circ}$ Chapter 6.4.5.20.1.35 "Visualization Element 'Trend"" on page 3246
- $\quad 4$ Chapter 6.4.5.20.1.45 "Visualization Element 'Date Range Picker'" on page 3301
- ${ }^{\text {y }}$ Chapter 6.4.5.20.1.46 "Visualization Element 'Time Range Picker'" on page 3306


### 6.4.5.13.2 Getting Started with Trend Visualization

When you execute a Trend, it is best to proceed with user guidance and the help of the trend wizard.

Development of a visualization with trend

1. Create an empty standard project and program at least one variable into PLC_PRG. $\Rightarrow$ PLC_PRG is declared and implemented
2. Add the "Visualization" object to the device tree below "Application".
$\Rightarrow$ An empty visualization appears.
3. Open "Toolbox $\rightarrow$ Special Control".
4. Drag the "Trend" element to the visualization $\Rightarrow$ The "Trend Recording" dialog opens with the "Recording Settings".
5. Select the task in which the trend recording will be executed.


In general the trend recording runs in the same task as the main program, i.e. PLC_PRG.

Therefore, select MainTask.
6. Add a trend variable with "Add Variable" and assign an IEC variable from PLC_PRG to the trend variable.
7. Click "OK" to close "Trend Configuration".
$\Rightarrow$ There is a newly created object of the type Trend recording under "Trend Recording Manager". The active visualization contains a "Trend" element that is selected.
8. Click "Visualization $\rightarrow$ Insert Elements for Controlling Trend Elements".
$\Rightarrow$ The "Trend Wizard" dialog box opens.
9. By default, all three control elements are activated in the dialog. Click "OK" to close the dialog box.
$\Rightarrow$ The active visualization contains a "Trend" with control elements.
10. Set the application containing the trend objects to active.
11. Compile the application with [F11].
12. Click "Online $\rightarrow$ Login".
13. Start the application with [F5].
$\Rightarrow$ The target visualization appears. The visualization contains the trend diagram with the value curve of the variable. The control elements enable user inputs.
\# Chapter 6.4.5.13.3 "Programming a Trend Visualization" on page 2931
« $\boldsymbol{y}^{2}$ Chapter 6.4.5.21.2.18 "Command 'Insert Elements for Controlling the Trend'" on page 3360

```
Example: Visu-
alization of the
sinusoidal trend
of an IEC vari-
able.
- PLC_PRG
- Visualization_Trend1
able.
- VisuWithTrend
```

The following objects are implemented in the project:

```
PLC_PRG PLC_PRG runs as part of the application on the controller.
```

```
    PROGRAM PLC_PRG
VAR
        iVar : INT;
        rSin : REAL;
        rVar : REAL;
END_VAR
iVar := iVar + 1;
iVar := iVar MOD 33;
rVar := rVar + 0.1;
rSin := 30 * SIN(rVar);
```

Visualizatio Visualization_Trend1 is the object that contains the configuration of the trend recording. n_Trend1

VisuWithTren VisuWithTrend is the object that visualizes the trend.
d
The visualization contains four elements: one "Trend" and three control elements. The properties of the trend are defined as follows.

| Properties | Value |
| :--- | :--- |
| "Trend recording" | Visualization_Trend1 |
| "Display cursor" | $\checkmark$ |
| "Display tool tip" | $\boxed{ }$ |
| "Show frame" | $\boxed{ }$ |
| "Date Range Picker" | "㬂 Trend1DateRangeSelector |
| "Time Picker" | (2) Trend1TimeSelector |
| "Legend" | i Trend1Legend |

VisuWithTrend at runtime



### 6.4.5.13.3 Programming a Trend Visualization

To display a trend recording in the visualization, you define which application provides which trend recording. You define the display by means of the "Properties" of the trend element and the controls used.

## Defining the application and data source

The visualization task and the trend recording task generally run under the same application. If this is not the case, then the application containing the visualization task requires a data source manager.

1. Select a trend element in the active visualization editor.
$\Rightarrow$ The properties of the trend element are displayed on the right side.
2. Double-click the value field "Properties $\rightarrow$ Application".
3. Use the Input Assistant (...$)$ to select the application. You can also specify the name of the application directly.

See also

- Trend Recording
- Data Source Manager


## Adding a con-

 trol1. Select a trend element in the active visualization editor.
2. Click "Visualization $\rightarrow$ Insert Elements for Trend Controlling".
$\Rightarrow$ The "Trend Wizard" dialog opens.
3. Select the desired control. Examples: "Date Range Picker", "Time range Range Picker", "Legend". Click "OK" to confirm.
$\Rightarrow$ The selected controls are inserted for the trend element. You can move them to any position you like. In the "Properties" of the trend element, the controls are shown below "Assigned controls".

See also

- ${ }^{\text {® }}$, Chapter 6.4.5.21.2.18 "Command 'Insert Elements for Controlling the Trend"" on page 3360


## Defining the trend recording <br> to visualize

1. Select a trend element in the active visualization editor.
$\Rightarrow$ The properties of the trend element are displayed on the right side.
2. Click the value field of "Properties $\rightarrow$ Trend recording"
$\Rightarrow$ "Select trend recording" is displayed. The trend recordings available application-wide are listed under "Available trend recordings".
3. Select a trend record below "Available trend recordings".
4. Click $\rightarrow$.
$\Rightarrow$ The trend recording is located under "Selected trend recording".
5. Click "OK" to confirm the entry.
$\Rightarrow$ The selected trend recording is listed in "Values" in "Properties $\rightarrow$ Trend recording".

See also

- Trend Recording

A control that was added with the help of the "Trend Wizard" cannot be deleted via the Trend wizard dialog.

1. Select a control of a trend in the active visualization editor.
2. Press [Del] or "Delete" to delete the element.
3. Select the trend in the active visualization editor.
4. Delete the assigned value in "Properties $\rightarrow$ Assigned controls $\rightarrow$ <control>".

## NOTICE!

It is absolutely necessary to delete this reference manually. The property is not deleted automatically by deleting the control.

## Configuring the coordinate system of the trend diagram

1. Select a trend in the active visualization editor.
2. Use the "Visualization $\rightarrow$ Configure Trend Display Settings" command.
$\Rightarrow$ The "Display Settings" dialog opens.
3. Adapt the settings as needed.

See also

- \& Chapter 6.4.5.21.2.18 "Command 'Insert Elements for Controlling the Trend"" on page 3360

Reading a trend value at runtime

1. Open "View $\rightarrow$ Element Properties".
2. Select a trend element in your visualization.
$\Rightarrow$ The properties of the trend element are displayed on the right side.
3. Select the "Properties $\rightarrow$ Show cursor" option and "Show tooltip".
$\Rightarrow$ A cursor is drawn in the coordinate system.
4. Select the "Properties $\rightarrow$ Show tooltip" option.
5. Download the application to the controller and start the application.
6. 

If the diagram "runs", then the date range has been placed in such a way that its end time is the current time.

Select the date range so that the diagram does not run. If necessary, drag the scroll bar to an earlier date range.
$\Rightarrow$ A cursor is available. The tooltip of the cursor informs you of the trend values. For each trend variable, the legend displays the value at the point in time at which the cursor is positioned.

Deleting the You can insert an input element in the visualization which the operator can use to delete the trend recording history previous value recording in the trend visualization at runtime. The curve displayed until then is removed and the display starts over.

1. In the application (example: in the program PLC_PRG), implement the following code:
```
itfTrendRecording : ITrendRecording;
itfTrendStorageWriter : ITrendStorageWriter;
itfTrendStorageWriter3 : ITrendStorageWriter3;
sTrendRecordingName : STRING := 'TrendRecording';
itfTrendRecording :=
GlobalInstances.g_TrendRecordingManager.FindTrendRecording(ADR(sTr
endRecordingName));
xClearHistoryTrend: BOOL;
IF xClearHistoryTrend THEN
itfTrendRecording :=
GlobalInstances.g_TrendRecordingManager.FindTrendRecording(ADR(sTr
endRecordingName));
IF itfTrendRecording <> 0 THEN
    itfTrendStorageWriter :=
itfTrendRecording.GetTrendStorageWriter();
        IF QUERYINTERFACE(itfTrendStorageWriter,
itfTrendStorageWriter3) THEN
itfTrendStorageWriter3.ClearHistory();
            END IF
END IF
```

2. In the visualization of the trend recording, add a button for deleting the previous curve. Configure its "Toggle" property with the variable PLC_PRG.xClearHistoryTrend.
$\Rightarrow$ When xClearHistoryTrend is set to TRUE, the previously recorded curve is deleted. The recording immediately starts again.

### 6.4.5.14 Displaying and Editing Text Files

6.4.5.14.2 Configuring the Display of a Text File ..... 2934
6.4.5.14.3 Configuring the Editing of a Text File ..... 2937

### 6.4.5.14.1 General

With the help of the element "Text Editor" you can display a text file in the user interface and optionally also enable the user to edit the file.

### 6.4.5.14.2 Configuring the Display of a Text File

In order to display a text file that is located on the controller, you need not only the element "Text Editor", but also control elements for selecting, opening and closing the file. Optionally a text search function can be set up in the file with further control elements.
Example:

## Display File



Read-only mode

Configuring the element "Text Editor", example

1. Drag an element "Text Editor" into the visualization editor.
2. Declare the control variables for the element, for example as global variables in the GVL object.
$\Rightarrow$ Refer to the declaration of the control variables for this.
3. For the "Text Editor", configure the property "Editing mode" with "Read only".
4. Also configure the property "Control variables".

Assign the following variables there:

- "Control variables $\rightarrow$ File $\rightarrow$ Variable" with g_sFileName
- "Control variables $\boldsymbol{\rightarrow}$ File $\rightarrow$ Open" with g_bFileOpen
- "Control variables $\rightarrow$ File $\rightarrow$ Close" with g_bFileClose
- "Control variables $\rightarrow$ File $\rightarrow$ New $\rightarrow$ Variable" with g_bFileNew
- "Control variables $\rightarrow$ File $\rightarrow$ Save $\rightarrow$ Variable" with g_bFileSave
- "Control variables $\rightarrow$ Edit $\rightarrow$ Variable" with g_sEditSearchFor
- "Control variables $\rightarrow$ Edit $\rightarrow$ Find" with g_bEditFind
- "Control variables $\rightarrow$ Edit $\rightarrow$ Find next occurrence" with g_bEditFindNext

Declaring the control variables

Configuring control elements for the file selection

## Control elements for searching for a text.

VAR_GLOBAL

```
```

g_sFileName: STRING := 'Readme.txt';

```
```

g_sFileName: STRING := 'Readme.txt';
g_bFileOpen : BOOL;
g_bFileOpen : BOOL;
g_bFileClose: BOOL;
g_bFileClose: BOOL;
g_bFileNew: BOOL;
g_bFileNew: BOOL;
g_bFileSave: BOOL;
g_bFileSave: BOOL;
g_sEditSearchFor : STRING;
g_sEditSearchFor : STRING;
g_bEditFind : BOOL;
g_bEditFind : BOOL;
g_bEditFindNext : BOOL;
g_bEditFindNext : BOOL;
g_usiErrorHandlingVarForErrorCode: USINT;
g_usiErrorHandlingVarForErrorCode: USINT;
g_bVarForContentChanged : BOOL;
g_bVarForContentChanged : BOOL;
g-bVarForReadWriteMode: BOOL;
g-bVarForReadWriteMode: BOOL;
ENDD VAR

```
```

ENDD VAR

```
```

1. Add an element "Label".
2. Configure the property "Texts $\rightarrow$ Text" with File:.
3. Add an element "Rectangle" next to it, in which the user can then enter the file name:
4. Configure the property "Texts $\rightarrow$ Text" with $\% s$ :
5. Configure the property "Texts $\rightarrow$ Text variable" with g_sFileName.
6. Configure the property "Input configuration $\rightarrow$ OnMouseclick" with "Write a variable". In the dialog "Input Configuration", select "Text input" as the "Input type".

Activate the option "Use text output variable".
$\Rightarrow$ The rectangle for the input of the file name is configured.
7. Add an element "Button" for opening the file.
8. Configure the property "Texts $\rightarrow$ Text" with Open:
9. Configure the property "Input configuration $\rightarrow$ OnMouseclick" with "Toggle a variable".

Assign g_bFileOpen as a variable.
$\Rightarrow$ The button Open is configured.
10. Add a further element "Button" for closing the file.
11. Configure the property "Texts $\rightarrow$ Text" with Close:
12. Configure the property "Input configuration $\rightarrow$ OnMouseclick" with "Toggle a variable".

Assign g_bEditFile as a variable.
$\Rightarrow$ The button Close is configured.

1. Add an element "Label".
2. Configure the property "Texts $\rightarrow$ Text" with Text: .
3. Alongside it, add an element "Rectangle" for the input of the text to be found.
4. Configure the property "Texts $\rightarrow$ Text" with $\% \mathrm{~s}$ :
5. Configure the property "Texts $\rightarrow$ Text variable" with g_sEditSearchFor.
6. Configure the property "Input configuration $\rightarrow$ OnMouseclick" with "Write a variable". In the dialog "Input Configuration", select "Text input" as the "Input type".

Activate the option "Use text output variable".
$\Rightarrow$ The rectangle is configured.
7. Add an element "Button" for starting the search.
8. Configure its property "Texts $\rightarrow$ Text" with Find.
9. Configure the property "Input configuration $\rightarrow$ OnMouseclick" with "Toggle a variable". Assign g_bEditFind as a variable.
10. Also add the action "Execute ST-Code".

Program the action with: g_bEditFindNext := FALSE;
$\Rightarrow$ The button is configured.
11. Add a further element "Button".
12. Configure the property "Texts $\rightarrow$ Text" with Find next.
13. Configure the property "Input Configuration $\rightarrow$ OnMouseclick" with "Toggle a variable".

Assign g_bEditFind as a variable.
14. Also add the action "Execute ST code".

Program: g_bEditFindNext := TRUE;
$\Rightarrow$ The button is configured.

See also

- ${ }^{\text {® }}$ Chapter 6.4.5.20.1.41 "Visualization Element 'Text Editor'" on page 3274


### 6.4.5.14.3 Configuring the Editing of a Text File

In order to be able to create a new text file or edit an existing one on the controller with the "Text Editor" in the user interface, you need not only the element "Text Editor", but also control elements for selecting, opening, closing, saving and creating a file.

Example:


## Configuring the

 element "Text Editor", example:1. Drag an element "Text Editor" into the visualization editor.
2. Declare the control variables for the element, for example as global variables in the GVL object.
$\Rightarrow$ Refer below to the declaration of the control variables for this.
3. For the "Text Editor", configure the property "Editing mode" with "Read/Write".

Declaring the control variables
4. Also configure the property "Control variables".

Assign the following variables there:

- "Control variables $\rightarrow$ File $\rightarrow$ Variable" with g_sFileName
- "Control variables $\rightarrow$ File $\rightarrow$ Open" with g_bFileOpen
- "Control variables $\rightarrow$ File $\rightarrow$ Close" with g_bFileClose
- "Control variables $\rightarrow$ File $\rightarrow$ Save" with g_bFileSave
- "Control variables $\rightarrow$ File $\rightarrow$ New" with g_FileNew

VAR_GLOBAL

```
g_sFileName: STRING := 'Readme.txt';
g_bFileOpen : BOOL;
g_bFileClose: BOOL;
g_bFileSave: BOOL;
g_FileNew: BOOL;
g_usiErrorHandlingVarForErrorCode: USINT;
g_bVarForContentChanged : BOOL;
g_bVarForReadWriteMode: BOOL;
ENDD_VAR
```

Configuring control elements for file selection

1. Add an element "Label".
2. Configure it in the property "Texts $\rightarrow$ Text" with File:
3. Add an element "Rectangle" next to it.
4. Configure its property "Texts $\rightarrow$ Text" with $\% s$.
5. Configure its property "Texts $\rightarrow$ Text variable" with g_sFileName.
6. Configure the property "Input configuration $\rightarrow$ OnMouseclick" with "Write a variable". In the dialog "Input Configuration", select "Text input" as the "Input type".

Activate the option "Use text output variable".
$\Rightarrow$ The rectangle for the input of the file name is configured.
7. Add an element "Button".
8. Configure its property "Texts $\rightarrow$ Text" with New.
9. Configure the property "Input configuration $\rightarrow$ OnMouseclick" with "Toggle a variable".

Assign g_bFileNew as a variable.
$\Rightarrow$ The button New is configured.
10. Add a further element "Button".
11. Configure the property "Texts $\rightarrow$ Text" with Open:
12. Configure the property "Input configuration $\rightarrow$ OnMouseclick" with "Toggle a variable".

Assign g_bFileOpen as a variable.
$\Rightarrow$ The button Open is configured.
13. Add a further element "Button".
14. Configure its property "Texts $\rightarrow$ Text" with Save.
15. Configure the property "Input configuration $\rightarrow$ OnMouseclick" with "Toggle a variable".

Assign g_bFileSave as a variable.
$\Rightarrow$ The button Save is configured.
16. Add a further element "Button".
17. Configure its property "Texts $\rightarrow$ Text" with Close.
18. Configure the property "Input configuration $\rightarrow$ OnMouseclick" with "Toggle a variable". Assign g_bEditFile as a variable.
$\Rightarrow$ The button close is configured.

See also

- \& Chapter 6.4.5.20.1.41 "Visualization Element 'Text Editor"' on page 3274


### 6.4.5.15 Configuring a variable assignment with unit conversion

A variable that was assigned in a visualization can be linked with a unit conversion. This causes the variable value to be converted according to a predefined rule and the result is edited in the visualization.

You have already configured the conversion rules in the editor of an object of the type "Unit Conversion".

See also
-

## Linking a vari- <br> able with unit <br> conversion

$\checkmark$ Requirement: A project with a visualization is open. In addition, the application contains the object UnitConversion with the rule convert_A.

1. Select an element.
$\Rightarrow$ The view "Properties" opens.
2. When assigning a variable, link the variable iVar_A with a rule of the unit conversion:
convert_A.convert(iVar_A)
3. Compile, load and start the application.
$\Rightarrow$ The application runs. The visualization opens. The unit conversion is applied.
6.4.5.16 Using recipes in visualization elements

You can manage and use the recipes created in CODESYS by means of a visualization. For this purpose, the input configuration of a visualization element provides the following commands:

- "Read Recipe"
- "Write Recipe"
- "Load Recipe from File"
- "Save Recipe to File"
- "Create Recipe"
- "Delete Recipe"

See also

- ${ }^{4}$ Chapter 6.4.1.13.2.4 "Changing Values with Recipes" on page 2003
- ${ }^{\bullet}$, Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration'" on page 3370


## Example: <br> Loading recipes by means of visualization elements <br> Requirement: The "Visualization" object is added to the project. <br> 

1. Create a recipe according to the instructions in the section "Changing Values with Recipes - Creating Recipes".

Assign the following names:

- Recipe definition: "Recipes"
- Recipes: "Recipe1" and "Recipe2"
- Variables: iValue1 and iValue2

Type in different variable values of both recipes.
2. Open the "Visualization" object in the editor.
3. Drag a "Button" element to the visualization. Label it "Load Recipe 1". You can specify the text by double-clicking the element or in the "Texts $\rightarrow$ Text" property.
4. Click the value field of the "Input configuration": "OnMouseDown" property.
$\Rightarrow$ The "Input Configuration" dialog box opens.
5. Select "Execute command" in the left of the left side and click the button ${ }_{2}$
$\Rightarrow$ The configuration of the "Internal command" opens on the right side of the dialog.
6. Select the "Write Recipe" command from the drop-down list.
7. Click the $+{ }_{+}$button.
$\Rightarrow$ The "WriteRecipe" command is added to the list.
8. Specify the first parameter as Recipes and the second parameter as Recipe1.
9. Click "OK" to close the dialog box.
10. Drag a second button to the visualization, name it "Load Recipe 2", and repeat steps 4 to 8. For step 7, specify Recipe 2 as the second parameter.
11. Load the program to the controller and start it Click the "Load Recipe 1" and "Load Recipe 2", and monitor the variables iValue1 and iValue2.

The other recipe commands are assigned to visualization elements as described in this example. Refer to the help page of the input configuration for a description of the internal commands.

### 6.4.5.17 Creating a structured user interface

### 6.4.5.17.1 General

You can reference visualizations that are available or exist in the project in another visualization and thus reuse them. You obtain a structured user interface that consists of several visualizations. In principle you have the following possibilities to reference visualizations.
On the one hand you can display visualizations within a main visualization and toggle between them. The element "Frame" or "Tabs" serves here as a window area element that defines the display area for the referenced visualizations.
On the other hand you can configure a user input for a visualization that causes another visualization to open as a dialog. The requirements for this is that it has the visualization type "Dialog". A dialog is used to collect inputs from the user.
In addition, you can declare an interface for a visualization that is to be referenced in order to vary the display of the visualization at runtime. A visualization is thereby instanced with different data and executed.

See also

- ${ }^{*}>$ Chapter 6.4.5.20.1.6 "Visualization Element 'Frame'" on page 3053
- «y Chapter 6.4.5.20.1.10 "Visualization Element 'Tabs'" on page 3084


### 6.4.5.17.2 Displaying Multiple Visualizations in One Visualization

You can reference other visualizations within a main visualization either in a "Frame" or a "Tabs" element, and then display them in the window pane of the element.

In the case of the "Frame" element, you can freely program which of the visualizations is displayed at which time. One option is to use the switch frame variable of the "Frame" element, which automatically triggers a switch according to its value. You can also program additional controls which, after user input, trigger input actions that result in switching a visualization.

## NOTICE!

Visualizations can be nested at any depth by means of "Frame" elements. In order to use the "Switch to any visualization" frame selection type without any problems, a "Frame" must not contain more than 21 referenced visualizations.
For more information, see also the description for the "Input configuration" of an element: Action "Switch frame visualization".

Moreover, you can use the "Tabs" to reference visualizations. It is easy and advantageous that the "Tabs" element provides preconfigured control of the visualization switch.


In CODESYS Forge, you will find the sample project "Visualization Switching". There you will see a visualization that displays other visualizations in a frame area one after another at runtime. The visualization switch is controlled either by the user, programmatically, or via the FrameManager.

See also

- Sample project in CODESYS Forge

Switching frame visualizations by means of a variable

Connecting frame visualizations with a radio buttons element

In the main visualization, the "Frame" element displays one of the referenced frame visualizations at runtime. The user can select the "Radio Buttons" element which is displayed in the frame.

1. Create a new standard project in CODESYS.
2. Select the application in the device tree and click "Add Object $\rightarrow$ Visualization".
3. In the "Add Visualization" dialog, specify the name VisuMain and click "Add" to close the dialog.
4. Select the application in the device tree and click "Add Object $\rightarrow$ Visualization".
5. In the "Add Visualization" dialog, specify the name Visu1 and click "Add" to close the dialog.
6. Select the application in the device tree and click "Add Object $\rightarrow$ Visualization".
7. In the "Add Visualization" dialog, specify the name Visu2 and click "Add" to close the dialog.
8. Select the application in the device tree and click "Add Object $\rightarrow$ Visualization".
9. In the "Add Visualization" dialog, specify the name Visu3 and click "Add" to close the dialog.
$\Rightarrow$ In addition to the main visualization, there are three more visualization objects.
10. Open the Visul object.
11. In the "Visualization Toolbox", in the "Basic" category, select and drag the "Radio Buttons" element to the visualization editor.
$\Rightarrow$ The "Properties" view of the element opens.
12. Configure the properties of the rectangle as follows:

- Property "Texts", "Text" = Visu1
- Property "Text properties", "Font" = "Title"
- Property "Colors", "Normal state", "Fill color" = "Light gray"


13. Program the object Visu2 accordingly.

Properties of the rectangle:

- Property "Texts", "Text" = Visu2
- Property "Text properties", "Font" = "Title"
- Property "Colors", "Normal state", "Fill color" = "Gray"

14. Program the object Visu3 accordingly.

Properties of the rectangle:

- Property "Texts", "Text" = Visu3
- Property "Text properties", "Font" = "Title"
- Property "Colors", "Normal state", "Fill color" = "Dark gray"

15. Open the VisuMain object.
16. In the "Visualization Toolbox", in the "Basic" category, select and drag the "Frame" element to the visualization editor.
$\Rightarrow$ The "Frame Configuration" dialog opens.
17. In the "Available Visualizations" window area, on the "By Visualization Name" tab, select the object Visu1. In "Selected Visualizations", click "Add".
18. Then select the object Visu2 and click "Add" in "Selected Visualizations".
19. Then select the object Visu3 and click "Add" in "Selected Visualizations".
20. Click "OK" to exit the dialog.
$\Rightarrow$ Now the "Frame" element references the three selected visualizations. The references (1) are listed in the "References" property in the element properties of the "Frame" element. In addition to the visualization name, the corresponding index value (2) is also displayed.


Note: You can open the dialog when you click the "Configure" button in the value field of the "References" property. See (3). You can influence the index by means of the visualization order in the "Selected Visualizations" list.
21. In the "Visualization Toolbox", in the "Common Controls" category, drag the "Radio Buttons" element to the visualization editor.
$\Rightarrow$ The "Properties" view of the element opens.
22. In the "Radio button settings", "Radio button", click the "Create new" button.
$\Rightarrow$ This element has three switches to select from.
23. Configure the properties of the radio button as follows:

- Property "Radio button settings", "Areas", "[0]", "Text" = Visu1
- Property "Radio button settings", "Areas", "[1]", "Text" = Visu2
- Property "Radio button settings", "Areas", "[2]", "Text" = Visu3

| Typ | $X$ | $Y$ | Breite | Höhe |
| :---: | :--- | :--- | :--- | :--- |
| $\langle$ \#0 Frame | 10 | 10 | 160 | 110 |
| \#1 Radiobutton | 200 | 20 | 100 | 100 |


24. In the PLC_PRG program, declare a local variable for the number of the visualization that is active.

```
=>
    VAR
        iActiveVisu : INT; // Index of visu activated by the user
    END_VAR
```

25. Select the "Radio Buttons" element. In the value field of the "Variable" property, click $\qquad$
26. In the "Input Assistant" dialog, select the recently declared variable. Then exit the dialog. $\Rightarrow$ Property of the "Radio Buttons" element:

- Property "Variable" = PLC_PRG. iActiveVisu

27. Select the "Frame" element. Click in the value field of the "Switch frame variable", "Variable" property. Specify the recently declared variable here as well.
$\Rightarrow$ Property of the "Frame" element:

- Property "Switch frame variable", "Variable" = PLC_PRG.iActiveVisu

The control variable of the "Radio Buttons" element is also the switch frame variable of the "Frame" element. User input for the "Radio Buttons"element switches the frame visualization.
28. Click "Build $\rightarrow$ Generate Code".
29. Click "Online $\rightarrow$ Login" and start the application.
$\Rightarrow$ The visualization starts. One of the referenced visualizations is running in the frame. When you click an unselected option of the "Radio Buttons" element, the visualization switches the contents in the frame to the desired visualization.


In the example, the switch frame variable is connected to an input variable. Instead, you can also set the switch frame variable programmatically in the IEC code.

## Switching frame visualizations by means of a follow-up action <br> Programming a visualization

In the main visualization, the "Frame" element displays one of the frame visualizations at runtime. The user can use buttons to control the display in the frame. The user input triggers the "Switch frame visualization" input action.

1. Create a new standard project in CODESYS.
2. Select the application in the device tree and click "Add Object $\rightarrow$ Visualization".
3. In the "Add Visualization" dialog, specify the name VisuMain and click "Add" to close the dialog.
4. Select the application in the device tree and click "Add Object $\rightarrow$ Visualization".
5. In the "Add Visualization" dialog, specify the name Visu1 and click "Add" to close the dialog.
6. Select the application in the device tree and click "Add Object $\rightarrow$ Visualization".
7. In the "Add Visualization" dialog, specify the name Visu2 and click "Add" to close the dialog.
8. Select the application in the device tree and click "Add Object $\rightarrow$ Visualization".
9. In the "Add Visualization" dialog, specify the name Visu3 and click "Add" to close the dialog.
$\Rightarrow$ In addition to the main visualization, there are three more visualization objects.
10. Open the Visul object.
11. In the "Visualization Toolbox", in the "Basic" category, select and drag the "Rectangle" element to the visualization editor.
$\Rightarrow$ The "Properties" view of the element opens.
12. Configure the properties of the rectangle as follows:

- Property "Texts", "Text" = Visu1
- Property "Text properties", "Font" = "Title"
- Property "Colors", "Normal state", "Fill color" = "Light gray"


13. Program the object Visu2 accordingly.
$\Rightarrow$ Properties of the rectangle:

- Property "Texts", "Text" = Visu2
- Property "Text properties", "Font" = "Title"
- Property "Colors", "Normal state", "Fill color" = "Gray"

14. Program the object Visu3 accordingly.
$\Rightarrow$ Properties of the rectangle:

- Property "Texts", "Text" = Visu3
- Property "Text properties", "Font" = "Title"
- Property "Colors", "Normal state", "Fill color" = "Dark gray"

15. Open the VisuMain object.
16. In the "Visualization Toolbox", in the "Basic" category, select and drag the "Frame" element to the visualization editor.
$\Rightarrow$ The "Frame Configuration" dialog opens.
17. In the "Available Visualizations" window area, on the "By Visualization Name" tab, select the object Visu1. In "Selected Visualizations", click "Add".
18. Then select the object Visu2 and click "Add" in "Selected Visualizations".
19. Then select the object Visu3 and click "Add" in "Selected Visualizations".
20. Click "OK" to exit the dialog.
$\Rightarrow$ Now the "Frame" element references the three selected visualizations. The references (1) are listed in the "References" property in the element properties of the "Frame" element. In addition to the visualization name, the corresponding index value (2) is also displayed.


Note: You can open the dialog independently when you click the "Configure" button in the value field of the "References" property. See (3). You can influence the index by means of the visualization order in the "Selected Visualizations" list.
21. In the "Visualization Toolbox", in the "Common Controls" category, drag the "Button" element to the visualization editor.
$\Rightarrow$ The element is selected and its properties are visible in the "Properties" view.
22. Configure the "Texts", "Text" property with Visu1.
23. In the "Input configuration""OnMouseDown" property, click "Configure".
$\Rightarrow$ The "Input Configuration" dialog opens.
24. Select the "Switch frame visualization" action and click $\square$.
$\Rightarrow$ The action is displayed in the window on the right.
25. Configure the action:

- Select the "Switch local visualization" option.
- Set the "Visualization selection" to Visu1.
- Click "OK" to exit the dialog.
$\Rightarrow$ The follow-up action is configured in the "Input configuration" property.
Property "Input configuration", "OnMouseDown", "Switch frame visualization" $=0$

26. Drag another "Button" element to the visualization editor. Configure the button accordingly. $\Rightarrow$ Properties of the button:

- Property "Texts", "Text" = Visu2
- Property "Input configuration", "OnMouseDown", "Switch frame visualization" = 1

27. Drag another "Button" element to the visualization editor. Configure the button accordingly. $\Rightarrow$ Properties of the button:

- Property "Texts", "Text" = Visu3
- Property "Input configuration", "OnMouseDown", "Switch frame visualization" = 2

28. Click "Build $\rightarrow$ Generate Code".
29. Click "Online $\rightarrow$ Login" for the device and start the application.
$\Rightarrow$ The visualization starts. One of the referenced visualizations is running in the frame. When you click one of the buttons, the visualization switches the contents in the frame to the respective visualization.


Displaying visu- For the "Tabs", the navigation of the referenced visualizations is provided automatically. The first tabs element
Configuring a tabs element
alizations on a of the referenced visualizations is in the foreground, while the others are hidden behind it. The user can navigate between them by means of the tabs which are provided automatically.

1. Create a new standard project in CODESYS.
2. Select the application in the device tree and click "Add Object $\rightarrow$ Visualization".
3. In the "Add Visualization" dialog, specify the name VisuMain and click "Add" to close the dialog.
4. Select the application in the device tree and click "Add Object $\rightarrow$ Visualization".
5. In the "Add Visualization" dialog, specify the name Visul and click "Add" to close the dialog.
6. Select the application in the device tree and click "Add Object $\rightarrow$ Visualization".
7. In the "Add Visualization" dialog, specify the name Visu2 and click "Add" to close the dialog.
8. Select the application in the device tree and click "Add Object $\rightarrow$ Visualization".
9. In the "Add Visualization" dialog, specify the name Visu3 and click "Add" to close the dialog.
$\Rightarrow$ In addition to the main visualization, there are three more visualization objects.
10. Open the Visul object.
11. Drag a "Rectangle" element to the visualization editor.
$\Rightarrow$ The "Properties" view of the element opens.
12. Configure the properties of the rectangle as follows:

- Property "Texts", "Text" = Visu1
- Property "Text properties", "Font" = "Title"
- Property "Colors", "Normal state", "Fill color" = "Light gray"


13. Program the object Visu2 accordingly.
$\Rightarrow$ Properties of the rectangle:

- Property "Texts", "Text" = Visu2
- Property "Text properties", "Font" = "Title"
- Property "Colors", "Normal state", "Fill color" = "Gray"

14. Program the object Visu3 accordingly.
$\Rightarrow$ Properties of the rectangle:

- Property "Texts", "Text" = Visu3
- Property "Text properties", "Font" = "Title"
- Property "Colors", "Normal state", "Fill color" = "Dark gray"

15. Open the VisuMain object.
16. In the "Visualization Toolbox", in the "Basic" category, select and drag the "Frame" element to the visualization editor.
$\Rightarrow$ The "Frame Configuration" dialog opens.
17. In the "Available Visualizations" window area, on the "By Visualization Name" tab, select the object Visu1. In "Selected Visualizations", click "Add".
18. Then select the object Visu2 and click "Add" in "Selected Visualizations".
19. Then select the object Visu3 and click "Add" in "Selected Visualizations".
20. Click "OK" to exit the dialog.
$\Rightarrow$ Now the "Tabs" element references the three selected visualizations. The references (1) are listed in the "References" property in the element properties of the "Frame" element. In addition to the visualization name, the corresponding index value (2) is also displayed.


Note: You can open the dialog "Frame Configuration" dialog independently when you click the "Configure" button in the value field of the "References" property. See (3). You can influence the index by means of the visualization order in the "Selected Visualizations" list.
21. In the "Visualization Toolbox", in the "Common Controls" category, drag the "Tabs" element to the visualization editor.
$\Rightarrow$ The "Properties" view of the element opens.
22. Configure the properties of the tab as follows:

- Property "Tab width": 40
- Property "References", Visu1, "Header" = Visu1
- Property "References", Visu2, "Header" = Visu2
- Property "References", Visu3, "Header" = Visu3

| Type | $X$ | $Y$ | Width | Height |
| ---: | :--- | :--- | :--- | :--- |
| $\square$ \#0 Tab control | 10 | 10 | 250 | 150 |

## Visu1 $\mid$ Visu2 $\mid$ Visu3 $\mid$


23. Click "Build $\rightarrow$ Generate Code".
24. Click "Online $\rightarrow$ Login" for the device and start the application.
$\Rightarrow$ The visualization starts. One of the referenced visualizations is running in the "Tabs" element. Click the tab to switch to the respective visualization.


See also

- 鿊 "Dialog 'Frame Configuration'" on page 3348
- *y Chapter 6.4.5.20.1.6 "Visualization Element 'Frame'" on page 3053
- *y Chapter 6.4.5.20.1.10 "Visualization Element 'Tabs'" on page 3084


### 6.4.5.17.3 Calling a Visualization with an Interface

You can declare an interface for parameters for a visualization that is to be referenced. The actual parameters are passed to the interface (similar as in the case of a function block) when the visualization is called at runtime.

First of all, declare the interface variables in the visualization interface editor. Then configure the parameters that are transferred to the interface by assigning a data-type-compliant application variable to each interface variable. The assignment is configured in the "References" property in the case of a "Frame" or a "Tabs".

Depending on the display variant, the parameter transfer of local variables (with the VAR scope) is limited. If you execute the visualization as an integrated visualization, you can only transfer local variables having a basic data type as parameters. If the visualization is called as CODESYS TargetVisu or CODESYS WebVisu, then you can also transfer parameters with a user-defined data type.

| User-controlled | If you have configured visualization references and then save a change to the variable dec- <br> update of the <br> laration for one of these visualizations in an interface editor, then the "Updating the Frame |
| :--- | :--- |
| transfer parame- | Parameters" dialog appears automatically. The dialog prompts you to edit the references. A |
| ters | list of all the visualizations affected is displayed there, so that the parameter transfers can be <br> reassigned at the changed interface. |
|  | When the dialog is closed, the changes are accepted and the elements affected are displayed in |
| the "References" property. |  |

## Calling visualization with interface <br> (VAR_IN_OUT)

$\checkmark$ Requirement: The project contains a visualization and a main visualization. The main visualization contains an element that the visualization references.

1. Open the visualization.
2. Click "Visualization $\rightarrow$ Interface Editor".
3. Declare a variable in the interface editor.
$\Rightarrow$ The visualization has an interface and the "Updating the Frame Parameters" dialog appears.
4. Assign a type-compliant transfer parameter to the interface variables in all calls by entering an application variable in "Value". Close the dialog.
$\Rightarrow$ A transfer parameter is assigned at the points where the visualization is to be referenced. These now appear in the main visualization in the "References" property.

```
DATAPIE
(STRUCT) TYPE DATAPIE : // Parameter type used in visPie
STRUCT
    dwColor : DWORD; // Color data
    iStart : INT; // Angle data
    iEnd : INT;
    sLabel : STRING;
END_STRUCT
END_TYPE
GVL
```

```
{attribute 'qualified_only'}
```

{attribute 'qualified_only'}
VAR_GLOBAL CONSTANT
VAR_GLOBAL CONSTANT
c_dwBLUE : DWORD := 16\#FF0000FF; // Highly opaque
c_dwBLUE : DWORD := 16\#FF0000FF; // Highly opaque
c_dwGREEN : DWORD := 16\#FFOOFFO0; // Highly opaque
c_dwGREEN : DWORD := 16\#FFOOFFO0; // Highly opaque
c_dwYELLOW : DWORD := 16\#FFFFFFO0; // Highly opaque
c_dwYELLOW : DWORD := 16\#FFFFFFO0; // Highly opaque
c_dwGREY : DWORD :=16\#88888888; // Semitransparent
c_dwGREY : DWORD :=16\#88888888; // Semitransparent
c_dwBLACK : DWORD := 16\#88000000; // Semitransparent
c_dwBLACK : DWORD := 16\#88000000; // Semitransparent
c_dwRED: DWORD := 16\#FFFF0000; // Highly opaque
c_dwRED: DWORD := 16\#FFFF0000; // Highly opaque
END_VĀR
END_VĀR
PLC_PRG

```
```

PROGRAM PLC_PRG

```
PROGRAM PLC_PRG
VAR
VAR
    iInit: BOOL := TRUE;
    iInit: BOOL := TRUE;
    pieA : DATAPIE; // Used as argument when visPie is called
    pieA : DATAPIE; // Used as argument when visPie is called
    pieB : DATAPIE;
    pieB : DATAPIE;
    pieC : DATAPIE;
    pieC : DATAPIE;
    iDegree : INT; // Variable center angle for the pie element
    iDegree : INT; // Variable center angle for the pie element
used for animation
used for animation
END_VAR
END_VAR
IF iInit = TRUE THEN
IF iInit = TRUE THEN
    pieA.dwColor := GVL.c_dwBLUE;
    pieA.dwColor := GVL.c_dwBLUE;
    pieA.iStart := 0;
    pieA.iStart := 0;
    pieA.sLabel := 'Blue';
    pieA.sLabel := 'Blue';
    pieB.dwColor := GVL.c_dwGREEN;
    pieB.dwColor := GVL.c_dwGREEN;
    pieB.iStart := 22;
    pieB.iStart := 22;
    pieB.sLabel := 'Green';
    pieB.sLabel := 'Green';
    pieC.dwColor := GVL.c_dwYELLOW;
    pieC.dwColor := GVL.c_dwYELLOW;
    pieC.iStart := 45;
    pieC.iStart := 45;
    pieC.sLabel := 'Yellow';
    pieC.sLabel := 'Yellow';
    iInit := FALSE;
    iInit := FALSE;
END_IF
END_IF
iDegree := (iDegree + 1) MOD 360;
iDegree := (iDegree + 1) MOD 360;
pieA.iEnd := iDegree;
pieA.iEnd := iDegree;
pieB.iEnd := iDegree;
pieB.iEnd := iDegree;
pieC.iEnd := iDegree;
```

pieC.iEnd := iDegree;

```

Main visualization visMain at runtime:


Printing the instance name of a transfer parameter

In order to obtain and output the instance name of a transfer parameter, you can implement an interface variable (data type STRING) with the pragma \{attribute 'parameterstringof'\} in the VAR_INPUT scope.
\(\checkmark \quad\) The project contains a visualization and a main visualization. The main visualization contains elements that the visualization references.
1. Open the visualization.
2. Click "Visualization \(\rightarrow\) Interface Editor".
3. Declare an interface variable (VAR_IN_OUT).
\[
\Rightarrow \text { pieToDisplay : DATAPIE; }
\]
4. In the interface editor, declare a variable (VAR_INPUT) with attribute \{attribute
'parameterstringof'\}.
```

=> {attribute 'parameterstringof' := 'pieToDisplay'}
sNameToDisplay : STRING;

```
5. Save the changes.
\(\Rightarrow\) The "Updating the Frame Parameters" dialog does not open.
6. Insert a "Text Field" element.
7. In the "Texts", "Text" property, assign an output text to the text field.
```

=> Visualization of %s

```
8. In the "Text variables""Text variable" property, assign the interface variable to the text field.
```

g sNameToDisplay
visPie has a heading.

```

Example The visPie visualization consists of one pie until now. The visMain main visualization calls visPie in a "Tabs" control three times with different transfer parameters.

The visPie is extended with a text field that outputs the name of the parameters actually passed to the visualization. For this, the interface of visPie is extended with a string variable that contains the instance name of the specified transfer parameter. At runtime, each pie is overwritten.


Table 624: Properties of the "Text field" element:
\begin{tabular}{|l|l|}
\hline "Texts", "Text" & Visualization of \%s \\
\hline "Text variables", "Text variable" & sNameToDisplay \\
\hline
\end{tabular}

Interface of the 'visPie' visualization:
```

VAR_INPUT
{attribute 'parameterstringof' := 'pieToDisplay'}
sNameToDisplay : STRING;
END VAR
VAR IN OUT
piēToDisplay : DATAPIE;
END_VAR

```

\section*{Main visualization visMain at runtime:}
Visualization of PLC_PRG.pieA

\section*{See also}
- \({ }^{*}>\) Chapter 6.4.5.21.2.1 "Command 'Interface Editor'" on page 3340
- \({ }^{*}>\) Chapter 6.4.5.20.1.6 "Visualization Element 'Frame"' on page 3053
- \(\quad\) • Chapter 6.4.5.20.1.10 "Visualization Element 'Tabs'" on page 3084
- \(\quad \stackrel{y}{c}\) Chapter 6.4.5.21.3.3 "Dialog 'Update Frame Parameters'" on page 3367

\subsection*{6.4.5.17.4 Calling a dialog in a visualization}

You can configure a user input for a visualization that causes a referenced visualization to open as a dialog. For example, a user clicks on a button, whereupon a dialog opens requesting the input of values. A dialog is used to collect user inputs and, if it is modal, it can lead to inputs outside the dialog being blocked.

Only visualizations with the visualization type "Dialog" can be opened as dialog. The visualization type is configured in the dialog "Properties" of a visualization object.
Basic procedure:
\(\checkmark\) Requirement: The project contains a main visualization and a dialog.
1. Configure a user input for the main visualization with the action "OpenDialog" for the dialog.
\(\Rightarrow\) The opening of the dialog is configured.
2. Configure a user input for an element of the dialog with the action "CloseDialog".

Hint: in the case of non-modal dialogs you can also configure the user input for closing outside the dialog.
\(\Rightarrow\) The closing of the dialog is configured.

You can also use dialogs from the library instead of self-made dialogs. For example, if the library VisuDialogs is integrated in the project, you can use the dialogs VisuDialogs. Login or VisuDialogs. FileOpenSave contained in it.

See also
- \(\Rightarrow\) Chapter 6.4.5.6 "Configuring user inputs" on page 2885
- \({ }^{\bullet} \boldsymbol{y}\) Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration'" on page 3370
- ② Chapter 6.4.5.21.3.15 "Dialog 'Properties' of Visualization Objects" on page 3388

\section*{Configuring a}
visualization
object as a
dialog
1. Select the object in the view "Devices", open the context menu and select the command "Properties".
2. Select the tab "Visualization".
3. Activate the option "Dialog" and close the dialog with "OK".
\(\Rightarrow\) The visualization has the visualization type "Dialog" and can be called as such.

\section*{Configuring a dialog call}

When calling a dialog, a user normally clicks on a button, whereupon a dialog opens requesting an input.

In the following example, a dialog representing a calendar enables a date to be entered.
\(\checkmark\) Requirement: The project contains the visualizations visMain and dlgCalender.
1. Set the visualization type of dlgCalender to Dialog.
2. Drag a rectangle into the visualization visMain.
3. Configure the property "Texts \(\rightarrow\) Text" with the text Due Date: \%t [yyyy-MM-dd]. Configure the property "Text variables \(\rightarrow\) Text variable" with PLC_PRG. dateDue.
4. Drag a button into the visualization.
5. Configure the property "Texts \(\rightarrow\) Text" with the text Open dialog.

Configure the property "Input configuration \(\rightarrow\) OnMouseClick" for the action "Open Dialog" with dlgCalender.
\(\Rightarrow\) The user input for the opening of the dialog is configured.
6. Double-click on the dialog dlgCalender.
7. Drag the element "Date picker" into the visualization editor.
8. Configure the property "Texts \(\rightarrow\) Text" with Due Date: \%t [yyyy-MM-dd]. Configure the property "Variable" with PLC_PRG. dateCalender. \(\Rightarrow\) The element is configured.
9. Drag a button into the visualization editor.
10. Configure the property "Texts \(\rightarrow\) Text" with OK:
11. Configure the property "Input configuration \(\rightarrow\) OnMouseClick"for the action "Close Dialog "with dlgCalender, Result: OK.
12. Configure a further property "Input configuration \(\rightarrow\) OnMouseClick" for the action "Execute ST-Code" with PLC_PRG.dateDue := PLC_PRG.dateCalendar;
\(\Rightarrow\) The user input for the closing of the dialog is configured.
13. Drag a further button into the visualization editor.
14. Configure the property "Texts \(\rightarrow\) Text" with Cancel:
15. Configure the property "Input configuration \(\rightarrow\) OnMouseClick" for the action "Close Dialog "with dlgCalender, Result: Cancel.
\(\Rightarrow\) The user input for the cancellation of the dialog is configured.
17. Compile, load and start the application.

\section*{Due date: 2016-04-21}


Variable decla- PROGRAM PLC_PRG

\section*{ration:}
```

VAR
dateDue : DATE := DATE\#2000-01-01;
dateCalendar : DATE;
END_VAR

```

Opening a Normally a dialog appears only on the display variant on which the user has executed the dialog globally triggering event.
However, you can configure the opening of the dialog in such a way that the dialog appears simultaneously on all active display variants configured under the visualization manager. This way, for example, an input request can appear simultaneously on all display variants although a user only entered something on the CODESYS TargetVisu.

If a user closes the dialog on a CODESYS TargetVisu display variant, it will be closed on all display variants.

Implementing an application access to a dialog

Implementing an application access to the dialog Login from the library VisuDialogs:

In the application code you can implement the access to a dialog that is managed in the dialog manager. The dialog manager automatically instances and manages all visualizations of the type "Dialog". The access takes place via the internal visualization manager.
First of all, implement the access to the dialog manager by calling the GetDialogManager () method of the internal visualization manager. You can then use the methods of the dialog manager to program the program sequence of a dialog.
In the following example a button is configured so that it opens the preconfigured dialog Login when clicked on. The user can enter a name and a password in the dialog. The dialog Login is contained in the library VisuDialogs. You can also call a self-made dialog in the same way.
\(\checkmark\) Requirement: The library VisuDialogs is integrated in the project.
1. Insert a new visualization visMain under the application.
\(\Rightarrow\) The visualization editor opens.
2. Drag a button into the visualization editor.
3. Enter in its property "Text"Login.
\(\Rightarrow\) The button is labelled.
4. Click on "Configure "in the property "Input configuration \(\rightarrow\) OnMouseDown".
5. Select the input action "Execute ST-Code" and click on \(\triangle\).
6. Enter the following function call in the ST editor: OpenLoginDialog (pClientData) ;
\(\Rightarrow\) The main visualization contains a button. If a user clicks on the button, the dialog Login opens and the function OpenLoginDialog() is called.
7. Click on "Configure "in the property "Input configuration \(\rightarrow\) OnDialogClosed".
8. Select the input action "Execute ST-Code" and click on \(\rightarrow\).
9. Enter the following function call in the ST editor:

OnLoginDialogClosed (pClientData);
\(\Rightarrow\) If a user closes the dialog, the function OnLoginDialogClosed () is called.

Login


Implementation of the function OpenLoginDialog():

VAR INPUT
pClíentData : POINTER TO VisuStructClientData; END_VAR

VAR
dialogMan : IDialogManager;
loginDialog : IVisualisationDialog;
pLoginInfo : POINTER TO Login_VISU_STRUCT; //
Login_VISU_STRUCT contains the parameters defined in the interface of visualizatīon "Login".
result : Visu_DialogResult;
stTitle : STRĪNG := 'Login ...';
stPasswordLabelText: STRING;
stUserLabelText: STRING;
stUsername: STRING;
END_VAR
dialogMan := g_VisuManager.GetDialogManager(); // The
DialogManager is provided via the implicitly available VisuManager
IF dialogMan <> 0 AND pClientData <> 0 THEN loginDialog :=
dialogMan.GetDialog('VisuDialogs.Login'); // Dialog to be opened is specified

IF loginDialog <> 0 THEN
pLoginInfo :=
dialogMan.GetClientInterface(loginDialog, pClientData);
IF pLoginInfo <> 0 THEN // In the following the parameters of the login dialog in the Login_VISU_STRUCT will be read pLoginInfo^.stTitle := stTitle;
pLoginInfo^.stPasswordLabelTxt := stPasswordLabelText; pLoginInfo^.stUserLabelTxt := stUserLabelText; dialogMan. OpenDialog(loginDialog, pClientData, TRUE,
\(0)\);
END_IF
END_IF
END_IF

OnLoginDialogClosed() defines the reaction to the closing of a dialog.

\section*{Implementation of the function OnLoginDialog Closed () :}
```

FUNCTION OnLoginDialogClosed : BOOL
VAR INPUT
pClientData : POINTER TO VisuStructClientData;
END_VAR
VAR
dialogMan : IDialogManager;
loginDialog : IVisualisationDialog;
pLoginInfo : POINTER TO Login_VISU_STRUCT;
result : Visu_DialogResult;
stPassword: STTRING;
stUsername: STRING;
END_VAR
dialogMan := g_VisuManager.GetDialogManager(); // The
DialogManager is provided via the implicitly available VisuManager
IF dialogMan <> 0 AND pVisuClient <> 0 THEN
loginDialog :=
dialogMan.GetDialog('VisuDialogs.Login'); // Gets the login dialog
IF loginDialog <> 0 THEN
result := loginDialog.GetResult(); // Gets the result
(OK, Cancel) of the dialog
IF result = Visu_DialogResult.OK THEN
loginDialog.SetResult(Visu_DialogResult.None); //
Reset to default (none)
pLoginInfo :=
dialogMan.GetClientInterface(loginDialog, pVisuClient); // Structure
Login_VISU_STRUCT gets read;
// In the following the structure parameters can be
set
IF pLoginInfo <> 0 THEN
stPassword :=

```
```

pLoginInfo^.stPasswordpLoginInfo^.stPassword := ''; // Reset the
passwword
stUsername := pLoginInfo^.stUsername;
END IF
ELS\overline{IF result = Visu_DialogResult.Cancel THEN}
loginDialog.SetResult(Visu_DialogResult.None); / /
React on 'Cancel'
ELSE
// nothing to do
END IF
END_IF
END_IF

```

See also
- \(\xi^{\boldsymbol{y}}\) Chapter 6.4.5.20.3 "Methods of the Dialog Manager" on page 3335

\subsection*{6.4.5.17.5 Calling a Dialog with an Interface}

You can define an interface for a visualization that is called as a dialog.
Create a visualization for this with visualization type "Dialog" and declare an interface for the dialog. The reference the visualization in a primary visualization by means of a user input and transfer the parameters to the interface.
If you call the visualization as an integrated visualization, then the parameter that are transferred must be variables of a basic data type. If the visualization is called as CODESYS TargetVisu or CODESYS WebVisu, then the parameters can have user-defined data types as well.
See also
- * "Scopes" on page 3340
- © Chapter 6.4.5.21.3.15 "Dialog 'Properties' of Visualization Objects" on page 3388

\section*{Main procedure}
1. Set the visualization types of the visualization to dialog.
2. Declare variables in the interface editor of the dialog.
\(\Rightarrow\) The dialog has an interface. You can transfer parameters when calling the dialog.
3. Configure the elements of the dialog and use the interface variables.
4. Select an element in another visualization (usually the main visualization) for configuring how the dialog opens.
5. Click "Configure" in the property "Input configuration \(\rightarrow\) OnMouseDown". \(\Rightarrow\) The "Input Configuration" dialog box opens.
6. Select "Open dialog" in the list of selected input actions.
7. Select one from the "Dialog" drop-down list. \(\Rightarrow\) If the selected dialog has an interface, then the interface variables are listed below.
8. Assign a transfer parameter to the interface variables in the "Value" column.
9. Select the result for which the parameters were updated in the list "Update" "o "and" "on "Parameter in case of results".
10. Activate the option "Open dialog modal". Click "OK" to close the dialog box. \(\Rightarrow\) The dialog opening is configured.

Executing a dialog several times at the same time requires multiple instances of the dialog. These must have already been downloaded to the visualization device when downloading the application. For this purpose, set the number of instances to download in the visualization manager ("Visualizations" tab).

See also
- \({ }^{4}\) ) Chapter 6.4.5.21.3.15 "Dialog 'Properties' of Visualization Objects" on page 3388
- \({ }^{4}\) Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration'" on page 3370
- ※ Chapter 6.4.5.17.5 "Calling a Dialog with an Interface" on page 2962

Example the following application calls the "Change User Level" dialog and prompts the user to select a level and specify a password. If the password agrees, then the "OK" button is enabled. Then the user can close the dialog. The input of the level is also applied.

\begin{tabular}{|c|c|c|c|}
\hline Type & Name & Element properties & Description \\
\hline \multirow[t]{2}{*}{} & \multirow[t]{2}{*}{} & "Radio button order": "Left to right" & Display \\
\hline & & \begin{tabular}{l}
"Radio button settings \\
\(\rightarrow\) Radio button \(\rightarrow\) Areas": [0] bis [7]"" \\
" \(<n>\) ] \(\rightarrow\) Text": <n>
\end{tabular} & Label of eight radio buttons with numbers from 0 to 7 \\
\hline \multirow[t]{3}{*}{\[
\begin{array}{|l|}
\hline \text { \#3 } \\
\text { Text } \\
\text { Field }
\end{array}
\]} & \multirow[t]{3}{*}{Input passw ord} & "Texts \(\rightarrow\) Text": \%s & Output with placeholder for text variable \\
\hline & & "Text variables \(\rightarrow\) Text variable": sItfPwd & Assignment of interface variable sItfPwd for which a parameter is transferred at call time. Includes the user input at runtime. \\
\hline & & \begin{tabular}{l}
"Input configuration \\
\(\rightarrow\) OnMouseDown \\
\(\rightarrow\) Write variable": \\
Variable:, InputType:Edit, \\
Use text output \\
variable : TRUE
\end{tabular} & In the "Input configuration" dialog, "Text input" is selected for the "Input type" drop-down list and the option "Use text output variable" is activated. \\
\hline \# 4 Text Field & Label for level & "Texts \(\rightarrow\) Text": Level: & Label \\
\hline \begin{tabular}{l}
\#5 \\
Text \\
Field
\end{tabular} & \begin{tabular}{l}
Label \\
for \\
passw \\
ord
\end{tabular} & "Texts \(\rightarrow\) Text": Password & Label \\
\hline \#6 & OK & "Texts \(\rightarrow\) Text": OK & Label \\
\hline & & \begin{tabular}{l}
"Colors \(\rightarrow\) Color": Element base color \\
"Colors \(\rightarrow\) Alarm color": Al arm fill color
\end{tabular} & Configuration of the display in state-dependent colors. You can switch between colors. \\
\hline & & "Color variables \(\rightarrow\) Toggle color":
sItfPwd <> MUX (iItfLevel,
sItfLevel0, sItfLevel1,
sItfLevel2, sItfLevel3,
sItfLevel4, sItfLevel5,
sItfLevel6, sItfLevel7); & If the password and the user input do not agree, then the expression is TRUE. Then the button is displayed in the alarm color. \\
\hline & & \begin{tabular}{l}
"State variables \\
\(\rightarrow\) Deactivate inputs": \\
sItfPwd <> MUX(iItfLevel, sItfLevel0, sItfLevell, sItfLevel2, sItfLevel3, sItfLevel4, sItfLevel5, sItfLevel6, sItfLevel7);
\end{tabular} & \begin{tabular}{l}
If the password and the user input do not agree, then the expression is TRUE. The button is deactivated. \\
If the password agrees, then the button is enabled.
\end{tabular} \\
\hline & & \begin{tabular}{l}
"Input configuration \\
\(\rightarrow\) OnMouseDown \\
\(\rightarrow\) Close dialog": \\
Close Dialog: \\
visChangeUserLevel, \\
Result : OK
\end{tabular} & If a user clicks the "OK" button, then the visChangeUserLevel dialog is closed and the parameters are updated. \\
\hline \begin{tabular}{l}
\#7 \\
Butto \\
n
\end{tabular} & Cance l & "Texts \(\rightarrow\) Text": Cancel & Label \\
\hline
\end{tabular}
\begin{tabular}{|l|l|l|l|}
\hline Type & Name & Element properties & Description \\
\hline \multirow{4}{|l|}{\begin{tabular}{ll} 
& \begin{tabular}{l} 
"Colors \(\rightarrow\) Color": Element base \\
color
\end{tabular} \\
\cline { 3 - 4 } & \begin{tabular}{l} 
"Input configuration \\
\(\rightarrow\) OnMouseDown \\
\(\rightarrow\) Close dialog": \\
Close Dialog: \\
visChangeUserLevel, \\
Result : Cancel
\end{tabular}
\end{tabular}} & \begin{tabular}{l} 
If a user clicks the "Cancel" button, \\
then the visChangeUserLevel \\
dialog is closed.
\end{tabular} \\
\hline
\end{tabular}

Main visualization visMain:


Table 626: Element list of the visMain visualization:
\begin{tabular}{|c|c|c|c|}
\hline Type & Name & Element properties & Description \\
\hline \multirow[t]{2}{*}{\#5 Text Field} & \multirow[t]{2}{*}{Button for change user level} & "Texts \(\rightarrow\) Text": \%s & Output with placeholder \\
\hline & & "Text variables \(\rightarrow\) Text variable": PLC_PRG.iLevel & Assignment of the PLC_PRG.iLevel variables to the placeholder. Includes the level number. \\
\hline \multirow[t]{2}{*}{\#6 Button} & \multirow[t]{2}{*}{Title} & "Texts \(\rightarrow\) Text": Change user level & \\
\hline & & \begin{tabular}{l}
"Input configuration \\
\(\rightarrow\) OnMouseDown \\
\(\rightarrow\) Open dialog": \\
Open Dialog: \\
visChangeUserLe vel
\end{tabular} & \begin{tabular}{l}
If a user clicks the Change user level button, then the visChangeUserLe vel dialog opens with the parameter list stored here. \\
Tip: Click "Configure" to view the stored configuration in the "Input Configuration" dialog (input action "Open dialog").
\end{tabular} \\
\hline
\end{tabular}

Table 627: Configuration of the call of dialog visChangeUserLevel:
\begin{tabular}{|l|l|l|l|}
\hline Parameter & Type & Value & Description \\
\hline \begin{tabular}{l} 
The parameter list is stored in the "Input Configuration" dialog (input action "Open \\
dialog").
\end{tabular} & STRING & \begin{tabular}{l} 
'ChangeUse user \\
level'
\end{tabular} & \begin{tabular}{l} 
Transfer of a string \\
for the title.
\end{tabular} \\
\hline sItfTitle & STRING & 'pwd0' & \begin{tabular}{l} 
Transfer of a string \\
as password for \\
Level0.
\end{tabular} \\
\hline sItfLevel0 & STRING & 'pwd1' & \begin{tabular}{l} 
Transfer of a string \\
as password for \\
Level1.
\end{tabular} \\
\hline sItfLevel1 & STRING & 'pwd2' & \begin{tabular}{l} 
Transfer of a string \\
as password for \\
Level2.
\end{tabular} \\
\hline sItfLevel2 &
\end{tabular}
\begin{tabular}{|l|l|l|l|}
\hline Parameter & Type & Value & Description \\
\hline sItfLevel3 & STRING & 'pwd3' & \begin{tabular}{l} 
Transfer of a string \\
as password for \\
Level3.
\end{tabular} \\
\hline sItfLevel4 & STRING & 'pwd4' & \begin{tabular}{l} 
Transfer of a string \\
as password for \\
Level4.
\end{tabular} \\
\hline sItfLevel5 & STRING & 'pwd5' & \begin{tabular}{l} 
Transfer of a string \\
as password for \\
Level5.
\end{tabular} \\
\hline SItfLevel6 & STRING & 'pwd6' & \begin{tabular}{l} 
Transfer of a string \\
as password for \\
Level6.
\end{tabular} \\
\hline sItfLevel7 & STRING & 'pwd7' & \begin{tabular}{l} 
Transfer of a string \\
as password for \\
Level7.
\end{tabular} \\
\hline iItfLevel & INT & PLC_PRG.iLevel & \begin{tabular}{l} 
Transfer of a vari- \\
able for the level \\
specified by the user.
\end{tabular} \\
\hline sItfPwd & STRING & PLC_PRG.sPwd & \begin{tabular}{l} 
Transfer of a vari- \\
able for the pass- \\
word specified by the \\
user.
\end{tabular} \\
\hline
\end{tabular}

Table 628: List "Update" and "Parameter in case of result"
\begin{tabular}{|l|l|}
\hline "Value" & Description \\
\hline "OK" & activated \\
\hline
\end{tabular}
```

"Open in dialog mode" activated Input outside of the dialog is not possible.

```

Application
code PLC_PRG:
```

PROGRAM PLC_PRG
VAR
iLevel: INT;
sPwd : STRING;
END_VAR

```


After clicking " \(O K\) ", the selection is applied.

\section*{visMain \(\times\)}


The example shows the procedure for multiple return values. However, the password can be returned more easily with a local variable in the dialog.

Accessing The variables declared in the interface of a visualization are available automatically as structure parameters programmatically variables. They are identified by <Name of visualization>_VISU_STRUCT. Therefore, you can access the interface variables of visualizations that appear as a dialog. Normally you use the structure in the application code of a function that is called by a user input.
\begin{tabular}{ll} 
Passing & To pass a complex data structure, you can flag an interface variable of type VAR_IN_OUT with \\
pointers as & the pragma attribute VAR_IN_OUT_AS_POINTER and pass a pointer or reference to it as a \\
parameters & parameter.
\end{tabular}

Procedure for using references
1. Declare the user data object (DUT).
2. In the interface editor of a dialog, declare an interface variable (VAR_IN_OUT) as a reference to the data object by assigning the attribute 'VAR_IN_OUT A \(\bar{S}\) _POINTER' to the variable.
3. Program the user interface: use the dialog in a visualization or assign the dialog in the input configuration of a visualization element. Then access to the referenced data is possible.

\section*{Example: \\ Using an inter- \\ face with the \\ pragma \\ 'VAR_IN_OUT \\ AS_POINTER' \({ }^{-}\)}
```

    FUNCTION_BLOCK ControlFB
    VAR
bOk : BOOL := TRUE;
nCounter : INT;
nValue : INT;
END_VAR
nCounter := nCounter + 1;

```

Declaration of an interface variable with VAR IN OUT AS POINTER


User interface: dialog opens:


See also
- \(\Leftrightarrow\) Chapter 6.4.5.20.4 "Attribute 'VAR_IN_OUT_AS_POINTER'" on page 3337

\subsection*{6.4.5.18 Configuring and executing display variants}

\subsection*{6.4.5.18.1 General}

You can select from different variants for displaying your visualization created in CODESYS. An advantage is that you can have not only one, but multiple display variants running at the same time. During this time, the contents of the visualization are the same for all variants. This also applies to the integrated visualization: when the visualization editor is open in CODESYS, the visualization is also displayed there with the same active contents.
The following object types are available:
- "TargetVisu"

The display as a variant of CODESYS TargetVisu is possible one time. You can insert an object of this type below the Visualization Manager.
- "WebVisu"

The display as a variant of CODESYS WebVisu is possible any number of times. You can insert any number of objects.
- "Remote TargetVisu"

The display as a variant of CODESYS TargetVisu is possible any number of times. You can insert any number of objects.
When you insert a variant below the Visualization Manager, the task configuration is extended by the visualization task VISU_TASK (the flow unit of the visualizations). The task is automatically deleted when no more objects exist below the Visualization Manager or the objects below are excluded from compiling. You can set this in the "Properties" dialog of an object, on the "Compile" tab.


> If no object is inserted below the Visualization Manager, then the visualization created there is displayed automatically as an integrated visualization when the application starts.

\section*{Exception handling at runtime}

If an error or an exception occurs in a visualization at runtime, the execution of the visualization is stopped without stopping the execution of the application. An error screen appears informing you of this. In addition, the error screen (usually) enables you to restart the visualization. This exception handling takes place from visualization profile CODESYS V3.5 SP7, compiler version 3.5.7.0 and a runtime system from version 3.5.7.0.

Select the command "Stop Execution at Handled Exceptions" in order to investigate the cause of the occurrence of exceptions and the error position.
See also
- \(\sum^{4}\) Chapter 6.4.1.21.3.7.19 "Command 'Stop Execution on Handled Exceptions'" on page 2639

\section*{Identifying display variants}

In order to programmatically identify a display variant, the VisuFbClientTagDataHelper library module from the VisuElemBase library is available to you. The library itself is referenced in VisuElems. The library module is typically called with VisuElems.VisuFbClientTagDataHelper.
Further information on this library module can be found in its documentation in the library manager.

See also
- ② Chapter 6.4.1.9.9 "Using Library POUs" on page 1890

\subsection*{6.4.5.18.2 Executing as CODESYS WebVisu}

NOTICE!
Recommendations for data security
In order to minimize the risk of breaches of data security, we recommend the following organizational and technical measures for the system on which your applications run:
As far as possible, avoid exposing the PLC and control networks to open networks and the Internet. For protection, use additional data-link layers such as a VPN for remote access and install firewall mechanisms. Limit access to authorized persons, change any existing standard passwords during the initial commissioning and continue to change them regularly.
If you nevertheless wish to publish your web visualization, it is urgently recommended that you provide it at least with simple password protection in order to prevent anyone accessing your control functionality over the Internet. (See an example in the project SimpleWebvisuLogin. project, which is provided with the standard installation of the development system).
Use the latest versions of the gateway server and the web server.

You can execute a visualization as CODESYS WebVisu.

The requirement for this is that the runtime system contains a web server with WebVisu support. This enables communication between target system and web browser. The web server on the target system is started as soon as an application with WebVisu configuration is started and runs until all applications with WebVisu are ended. The device can then display visualizations in connected HTML5-capable web browsers.
The web-based display variant of the CODESYS Visualization enables remote access to a plant as well as its remote monitoring, service and diagnosis over the Internet. A web browser communicates by Java Script (optionally with SSL encryption) with the web server in the controller and displays the visualization by means of HTML5. This technology is supported by virtually all browsers and is thus also available on terminal devices with iOS or Android.

\section*{Configuring and starting display variants}
\(\checkmark \quad\) An executable visualization visMain exists in the project.
1. Select the object "Visualization manager" and select the command "Add object".
2. Select the object "WebVisu" and enter the name WebVisu_A.
\(\Rightarrow\) There is a new object in the device tree underneath the object "Visualization Manager". The associated editor opens.

The visualization task VISU_TASK is automatically added under the task configuration.
3. Select the visualization visMain in the "Start Visualization".
4. In "Name of .htm file", enter the name webvisuA.
5. Click on "Show used visualizations" and check whether the selected visualization is activated for a download to the associated device.
\(\Rightarrow\) The visualization is configured. The settings under "Scaling options" determine the window size and the scaling.
6. Start a suitable runtime system with web server and WebVisu support.

Configure the communication settings for your system.
\(\Rightarrow\) The runtime system runs.
7. Compile, load and start the application.
\(\Rightarrow\) The application and the web server run.
8. Start a web browser with the following address: http://localhost:8080/ webvisuA.htm
\(\Rightarrow\) The page is displayed and you can see the data of the application and operate the application.

See also
- 出 Chapter 6.4.5.21.4.7 "Object 'TargetVisu'" on page 3408

\section*{Calling a page in the web server}
\(\square\) Requirement: A visualization with WebVisu is started.
1. Start a current browser with JavaScript and support of HTML5-Canvas, e.g. Firefox, Chrome, IE>=9.
2. Enter the following address in the web browser:
http://localhost:8080/webvisu.htm
Formal: http://<IP address of webserver>:<port of webserver>/<name of HTM-file>
<name of HTM-file> is the HTML start page of the visualization defined in the object "WebVisu".
\(\Rightarrow\) The page is displayed and you can see the data of the application and operate the application.

Identifying Web- In order to be able to identify a WebVisu with the help of the library block
Visu
VisuFbClientTagDataHelper, the WebVisu needs a name. In order to be able to specifically address it in the application, expand the URL call by the parameter ClientName=<Name>.

Example: http://localhost:8080/webvisu.htm?ClientName=VisClientxy.

See also
- ② Chapter 6.4.5.21.4.8 "Object 'WebVisu'" on page 3409

\subsection*{6.4.5.18.3 Executing as an Integrated Visualization}

You can execute the visualization as an integrated visualization. In this case a display variant of the visualization runs on the development system without the visualization code being loaded to the controller.
Use the integrated visualization for the testing and diagnosis of your application, or for the service and commissioning of a plant.

The requirement for this is that there are no objects under the visualization manager. Alternatively, any objects located there can be excluded from compilation. You can configure an individual object accordingly in its dialog "Properties" on the tab "Compile".
See also
- \({ }^{\sharp}\) Chapter 6.4.5.21.3.15 "Dialog 'Properties' of Visualization Objects" on page 3388

\section*{Configuring and starting display variants}

\section*{\(\checkmark\) A visualization project is open.}
1. Remove all objects from underneath the visualization manager or exclude the objects from compilation.
\(\Rightarrow\) The VISU_TASK has been removed from under the task configuration.
2. Load the application to the controller.
\(\Rightarrow\) Now no visualization code will be transferred on loading the application.
3. Start the application.
\(\Rightarrow\) The visualization in the visualization editor is being executed. You can operate your application.

\section*{\(\square\) \\ Use the command "Activate keyboard usage" in order to toggle between the keyboard usage of the integrated visualization and the keyboard usage of CODESYS.}

See also
- \({ }^{\text {² }}\) Chapter 6.4.5.21.2.4 "Command 'Activate Keyboard Usage"' on page 3343
- \({ }^{\text {® }}\), Chapter 6.4.5.21.1 "Keyboard Shortcuts for Default Keyboard Action" on page 3338

\section*{Restrictions in the variable output}

\section*{Data server restrictions}

\section*{Restrictions in variable types}

Numerical variable values, which are output within a text in an integrated visualization, are displayed according to the current display format. You can select the display format with the command "Debug \(\rightarrow\) Display".

See also
- 㘯 Chapter 6.4.1.21.3.8.24 "Command 'Display Mode' - 'Binary', 'Decimal', 'Hexadecimal'" on page 2654

A variable value that is transferred via the data server is not output. The integrated visualization only outputs the initialization or the last transferred value.

The integrated visualization thus only enables a passive observation of the application.

VAR_INPUT variables behave like integrated visualizations such as VAR_IN_OUT variables during execution.

Restrictions in Only the following expressions, which are also used in the monitoring mechanism of the develexpressions and opment system, are supported in an integrated visualization.
monitoring
Variable access:
- Example: PLC_PRG.myPou.nCounter

Array access:
- Access to an array of scalar data types, where a variable is used as an index

Example: a [i]
- Access to an array of complex data types (structure, function block, array), where a variable is used as an index
Example: a [i].x
- Access to a multidimensional array of all kinds of data types with one or more variable indices
Example: a[i, 1, j].x
- Access to an array with constant index

Example: a [3]
- Accesses like those described above in which simple operators are used for the calculations inside the index brackets.
Example: a [i+3]
- Nested combinations of the complex expressions listed above

Example: a[i + 4 * j].aInner[j * 3].x
Operators in index calculations:
- +, -, *, /, MOD

Restrictions in the input action "Execute STCode"

Pointer monitoring:
- Example: \(\mathrm{p}^{\wedge}\). x

Methods and function calls are not supported with the exception of the following:
- Standard string functions
- Type conversion functions Example: INT_TO_DWORD
- Operators such as SEL, MIN, etc.

\section*{Example}
```

PLC_PRG.n := 20 * PLC_PRG.m; // Don't use this!
IF P\overline{LC_PRG.n < MAX_COUNT THEN}
PLC_PR\overline{G}.n := PLC_PRGG.n + 1;
END-IF
//Usese the following!
PLC_PRG.n := MIN(MAX_COUNT, PLC_PRG.n + 1);

```

Restrictions in the interface of a visualization

No interface (INTERFACE) may be declared in the interface editor of a visualization.

\subsection*{6.4.5.18.4 Configure File Transfer Mode}

When downloading, usually files required by the visualization for displaying, are transmitted to the respective display unit. These are especially image files or text list files.

Alternatively, you can configure,that the visualization accesses local files. So no files are transferred with a download

The following configuration is required to allow the visualization access to local files:
- The file paths for image files or text files lists are relative.
- The link type for image files is "Link to file".

Using local visualization files
\(\checkmark\) Requirement: You have opened a visualization project with a image pool.
1. Open the image pool.
2. Select for each image under "Link Type" the setting "Link to file". \(\Rightarrow\) The image is linked.
3. Select the command "Project \(\rightarrow\) Project Setting" and select the category "Visualization".
4. Insert in tab "General" in "Image files" the local paths of the image files with relative path names.
\(\Rightarrow\) Example: . \; . \images \(\backslash\)
Note: When no path is specifiet, the setting in dialog box "Options", category "Visualization", tab "File Options" setting "Image files" is usesd.
5. Open the visualization manager.
6. Activate under "Extended settings" the option "Visible".
7. Activate under "File Transfer Mode" the option "Use local visualization files".
\(\Rightarrow\) When downloading, no files are transferred. When displaying the visualization, the local files are used.

See also
- \({ }^{4}\) Chapter 6.4.5.21.4.2 "Object 'Visualization manager'" on page 3398
- \({ }^{4}\) Chapter 6.4.1.21.2.15 "Object 'Image Pool'" on page 2468
- \({ }^{\aleph}\) Chapter 6.4.5.21.3.13 "Dialog 'Project Settings' - 'Visualization"'" on page 3387
- 学 Chapter 6.4.5.21.3.9 "Dialog Box 'Options' - 'Visualization'" on page 3384

\subsection*{6.4.5.19 Applying Visualization Styles}

\subsection*{6.4.5.19.1 General}

A visualization style is a collection of colors, fonts, images, and any values that are defined as style properties. When designing a visualization element, you can use these style properties only. The you have a uniform, style-dependent appearance.
An element that applies style colors and style fonts behaves according to the selected style design in each selected style. In this way, a style property, such as Element basic color, can be blue In one style and gray in another style. In contrast, if the color of an element has a fixed value, this color is fixed even when the style is switched.

All applicable styles are consistent because they define a fixed set of style properties. Therefore, you can switch smoothly between styles in order to customize your visualization. You can preview a style to get an impression of how it behaves.

CODESYS provides different styles, for example the styles Flat style and White Style. These provided styles are installed in the visualization style repository.
The selected style that applies to all visualizations in the application is set in the "Visualization Manager" object ("Settings" tab, "Style Settings" group, "Selected style"). In addition, the "Properties" view provides its style properties when designing an element. For each element, you can assign these styles instead of fixed values.

The style is applied to all visualizations that are below an application. The settings of the "Options - Visualization Styles" dialog are also considered for a library visualization or a visualization in the POUs view.

See also
- I \({ }^{\xi}\) Chapter 6.4.5.21.3.11 "Dialog 'Project Environment' - 'Visualization Styles'" on page 3386
- \(\Leftrightarrow\) Chapter 6.4.5.21.3.7 "Dialog 'Options' - 'Visualization Styles"' on page 3382
- \({ }^{\text {• }}\) Chapter 6.4.5.21.4.2 "Object 'Visualization manager'" on page 3398

Designing visu- The set style includes style properties. These are provided in the "Properties" view of an alization elements with style properties element in the drop-down list of the "Value" column. It is checked which style properties are appropriate for which property. For example, only style properties with color definitions are available for a color assignment.

A style can have directly defined visualization element properties. If this style is used in the project, then these properties are not configurable anymore.
\(\checkmark\) Requirement: A project is open with a visualization.
1. Double-click the visualization.
2. Select an element.
3. Choose "View \(\rightarrow\) Element Properties".
4. Click in the input field of a color in the window "Properties" (category "Colors").
\(\Rightarrow\) The list box opens with style properties. The style colors are based on the currently selected style.
5. Select a style property.
\(\Rightarrow\) The visualization shows the element according to the style.
```

Example
A visualization uses the style CompanyStyle 8 , which defines the colors CompanyRed, CompanyBlue, and CompanyGreen. An element is selected in the visualization. You can configure the element in the "Properties" view. By clicking into the value field of the "Color" property, you receive a drop-down list with the entries CompanyRed, CompanyBlue, and CompanyGreen.

```

\section*{Switching visualization styles}

When setting a style in the visualization manager, all complete styles in the repository are available for selection. It does not matter and it is not evident if a style have been derived from another style.
You can preview a style to get an impression of how it behaves.
How a visualization implements a style at runtime also depends on the display variant. For example, if a font that is defined in the style is not available, the display variant shows the visualization with a preset font.
\(\square\) Requirement: A project is open with a visualization.
1. Double-click the "Visualization Manager" object in the device tree.
\(\Rightarrow\) The editor opens.
2. Click in the input field of "Selected style" "Settings" tab, "Style Settings" group).
\(\Rightarrow\) All styles that are installed in the repository are listed.
3. Mouse over a style.
\(\Rightarrow\) A preview of how the style is displayed appears in a new window.
4. Select a style.
\(\Rightarrow\) The style is applied. The preview in "Style Settings" shows the new setting.
5. Double-click a visualization.
\(\Rightarrow\) The visualization appears in the new style.

\section*{Updating versions}
\(\square\) Requirement: A project is open with a visualization.
1. Click "Project \(\rightarrow\) Project Environment", "Visualization Styles" tab.
\(\Rightarrow\) CODESYS lists all new versions of the currently used styles.
2. Click "Set All to Newest".
\(\Rightarrow\) The style is updated. Visualizations and their elements apply the new style.

\subsection*{6.4.5.19.2 Editing visualization styles in the visualization style editor}

A style is an XML file with the file extension *.visustyle.xml. It contains a specific set of style properties. CODESYS checks the style properties in the consistency check.

You can create a new style or customize an existing style. The visualization style editor is available for this.

If you customize a style, then a new style is created as a hierarchy of styles. A hierarchy consists of at least two styles based on each other. The nesting depth is unrestricted. A hierarchy is identified simply with its top derived style. You can derive multiple different styles from one base style by extending the styles by differing style properties. This save memory and therefore should be your preferred method.

A base style does not have to be consistent for itself. Instead, you must identify it as an incomplete style. Only the top derived style must be consistent.

\section*{Example of a style hierarchy}

\section*{Style Petrostyle}

In a partial style, you can combine any style properties to form efficient hierarchies without having to worry about consistency. For example, you can collect all image references into one partial style. Then you derive the style and define more style properties for colors. This style is also incomplete. You derive the style again and define more style properties for its fonts. The top style is now completely.

- (1): CompanyImg is a partial style defining image references.
- (2): CompanyColor is a partial, derived style based on CompanyImg and also defines colors.
- (3): PetrolStyle is a complete, derived style based on CompanyColor and also defines a special color.
- (4): The hierarchy of styles comprises PetrolStyle, CompanyColor, and CompanyImg.

In the visualization style editor, you can open a style, define its style properties, and localize its name. If the style is consistent, then you can install it in the visualization style repository. The editor is not integrated in CODESYS. However, you can start the editor in CODESYS.

Names for style properties

Starting the editor in CODESYS

A style property is an entry for a specific color, a specific font, or a specific image reference.
If this name contains a dash, then the Visualization Style Editor can sort the style properties by the prefixed terms before the dash and display them in a hierarchy. Otherwise the names can be sorted in alphabetical order or sequential order or in sequential order (as saved in the XML file). CODESYS displays the style properties in the order of names actually saved in the XML file for the style.

Example: Element-Alarm-Fill-Color

See also
- \(\quad\) y Chapter 6.4.5.22.3 "Editor 'Visualization Style Editor'" on page 3750
1. Double-click the "Visualization Manager" object.
\(\Rightarrow\) The editor opens.
2. Click the symbol 3 ("Settings" tab, "Style Settings" group).
3. Click "Open Style Editor" from the drop-down list.
\(\Rightarrow\) The "Visualization Style Editor" opens.

Starting the editor independent of CODESYS
\(\triangleright \quad\) Choose "Visualization Styles Editor" from the CODESYS install folder in the Start menu. If you have a standard installation, then this link is located in CODESYS (the program folder for CODESYS).
\(\Rightarrow\) The visualization style editor opens.

\section*{Deriving visualization styles}

This is the recommended way to create a style that combines existing style properties with new ones.

Starting the editor in CODESYS and deriving stylesRequirement: CODESYS is open with a project containing a visualization.
1. Double-click the "Visualization Manager" object in the device tree.
\(\Rightarrow\) The editor opens.
2. Click the symbol 3 ("Settings" tab, "Style Settings" group).
\(\Rightarrow\) A list of commands opens.
3. Choose "Create and Edit Derived Style".
\(\Rightarrow\) The visualization style editor starts and the "Create a New Visualization Style" dialog box opens.
4. Type a name.
5. Select a directory.
6. Select a base style. The default style is set in CODESYS. You can also select a style from the repository.
\(\Rightarrow\) The new style appears in the visualization style editor. The style properties from the base style are displayed yellow.
7. Add a new style property or modify an existing value.
8. Provide a version for the style and click "File \(\rightarrow\) Save and Install".
\(\Rightarrow\) The style is installed in the repository. The memory requirement is low because only the style property added in step 7 is saved.

\section*{Copying visualization styles}
\(\checkmark\) Requirement: CODESYS is open with a project containing a visualization.
1. Double-click the "Visualization Manager" object in the device tree.
\(\Rightarrow\) The editor opens.
2. Click the symbol 3 ("Settings" tab, "Style Settings" group).
\(\Rightarrow\) A list of commands opens.
3. Choose "Copy and Edit Style".
\(\Rightarrow\) The visualization style editor starts and the "Open Existing Style as a Copy" dialog box opens.
4. Select which style should be copied ("Style").
5. Type a directory in "Destination" and click "OK".
\(\Rightarrow\) The new style appears in the visualization style editor. All style properties are identical to those in the copied style.
6. Type a name.
7. Add a new style property or modify the value of an existing style property.
8. Provide a version for the style and click "File \(\rightarrow\) Save and Install".
\(\Rightarrow\) The style is installed in the repository and the style properties are identical to the added style property, except for the style property added in step 8. The memory requirement is high because the common style properties are defined in both styles.

\section*{Creating new visualization styles}
\(\checkmark\) Requirement: CODESYS is open with a project containing a visualization.
1. Double-click the "Visualization Manager" object in the device tree.
\(\Rightarrow\) The editor opens.
2. Click the symbol 3 ("Settings" tab, "Style Settings" group).
\(\Rightarrow\) A list of commands opens.
3. Click "Open Style Editor".
\(\Rightarrow\) The visualization style editor opens.
4. Click "File \(\rightarrow\) New Style".
\(\Rightarrow\) The "Create a New Visualization Style" dialog box opens.
5. Type a name. Specify a base style.
6. Specify a directory and click "OK" to close the dialog box.
\(\Rightarrow\) The new style appears in the visualization style editor.
7. Add a new style property.
8. Provide a version for the style and click "File \(\rightarrow\) Save and Install".
\(\Rightarrow\) The style is installed in the Visualization Styles Repository with the version number.

Adding a style property

Using the visualization style editor, you can edit a style to save and install as a new version.
\(\square\) Requirement: The visualization style editor is open with a style.
1. Select a style property and click "Styles \(\rightarrow\) New Style (Afterwards)" in the "Style Properties" tab.
\(\Rightarrow\) A new style property is added.
2. In the "General" tab, type a new version number in the "Version" setting.
3. Choose "File \(\rightarrow\) Save and Install".
\(\Rightarrow\) The changes are saved and the style is installed in the repository as the new version.

Localizing style
properties

You can assign a language-dependent name to a style property. CODESYS displays a style property by its localized name, depending on the language settings in category "International Settings" (menu "Tools \(\rightarrow\) Options").
\(\checkmark\) Requirement: The visualization style editor is open with a style.
1. Translate the name of the style property into the localized language in the "Localization" tab.
2. Provide a version for the file in the "General" tab.
3. Choose "Save and Install". \(\Rightarrow\) The edited style is installed in the repository currently selected in CODESYS.
4. Update the style.
5. Set the language settings in CODESYS to the localized language.
6. Open a visualization and select an element. The style settings in its properties are displayed in the localized language.

\subsection*{6.4.5.19.3 Managing visualization styles in repositories}

The styles that are listed in CODESYS in the drop-down lists of different dialogs and editors are all checked for consistency and installed in the visualization style repository. For derived styles, the hierarchy is checked completely and all styles of the hierarchy are installed. The repository is a version control system within the development system.

You can open a style as write-protected from the visualization style repository in the visualization style editor. The "Save" and "Save and Install" commands are not available there for read-only files. However, you can derive it as the basis for a new style or as a copy.

See also
- \({ }^{4}\) Chapter 6.4.5.21.2.20 "Command 'Visualization Style Repository'" on page 3363
- \({ }^{\text {® }}\) Chapter 6.4.5.22.3 "Editor 'Visualization Style Editor'" on page 3750

\section*{Installing styles to repositories}
1. Click "Tools \(\rightarrow\) Visualization Style Repository". \(\Rightarrow\) The "Visualization Styles" dialog box opens.
2. Select the "System" repository in the drop-down list of "Storage location".
\(\Rightarrow\) All versions of the installed styles are listed in "Installed Visualization Styles".
3. Click on the "Install" button.
\(\Rightarrow\) The "Select Visualization style(s)" dialog box opens.
4. Select a style file and click "Open" to close the dialog box.
\(\Rightarrow\) The style is installed in the "System" repository. It appears now in the tree view below "Installed Visualization Styles".

\section*{Uninstalling}
styles
1. Click "Tools \(\rightarrow\) Visualization Style Repository". \(\Rightarrow\) The "Visualization Styles" dialog box opens.
2. Select a repository in the drop-down list of "Storage location".
\(\Rightarrow\) All versions of the installed styles are listed in "Installed Visualization Styles".
3. Select a style there.
4. Click the "Uninstall" button.
\(\Rightarrow\) The "Select Visualization Style(s)" dialog box opens.

Managing repositories
1. Click "Tools \(\rightarrow\) Visualization Style Repository".
\(\Rightarrow\) The "Visualization Styles" dialog box opens.
2. Click on the "Edit Locations" button.
\(\Rightarrow\) The dialog makes it possible to manage other repositories.

\subsection*{6.4.5.20 Reference, Programming}
6.4.5.20.1 Visualization Elements. ..... 2987
6.4.5.20.2 Placeholders with Format Definition in the Output Text. ..... 3329
6.4.5.20.3 Methods of the Dialog Manager. ..... 3335
6.4.5.20.4 Attribute 'VAR_IN_OUT_AS_POINTER' ..... 3337
6.4.5.20.5 Attribute 'parameterstringof' ..... 3338
6.4.5.20.1.1 Visualization Element 'Rectangle', 'Rounded Rectangle', 'El- lipse' ..... 2988
6.4.5.20.1.2 Visualization Element 'Line' ..... 3000
6.4.5.20.1.3 Visualization Element 'Polygon', 'Polyline', 'Bézier Curve'. ..... 3012
6.4.5.20.1.4 Visualization Element 'Pie' ..... 3025
6.4.5.20.1.5 Visualization Element 'Image' ..... 3038
6.4.5.20.1.6 Visualization Element 'Frame'. ..... 3053
6.4.5.20.1.7 Visualization Element 'Label' ..... 3068
6.4.5.20.1.8 Visualization Element 'Combo Box, Integer'. ..... 3072
6.4.5.20.1.9 Visualization Element 'Combo Box, Array' ..... 3079
6.4.5.20.1.10 Visualization Element 'Tabs'.... ..... 3084
6.4.5.20.1.11 Visualization Element 'Button' ..... 3089
6.4.5.20.1.12 Visualization Element 'Group Box'... ..... 3101
6.4.5.20.1.13 Visualization Element 'Table'. ..... 3106
6.4.5.20.1.14 Visualization Element 'Text Field'. ..... 3113
6.4.5.20.1.15 Visualization Element 'Scroll Bar'. ..... 3125
6.4.5.20.1.16 Visualization Element 'Slider' ..... 3134
6.4.5.20.1.17 Visualization Element 'Spin Box' ..... 3140
6.4.5.20.1.18 Visualization Element 'Invisible Input' ..... 3147
6.4.5.20.1.19 Visualization Element 'Progress Bar' ..... 3152
6.4.5.20.1.20 Visualization Element 'Check Box' ..... 3156
6.4.5.20.1.21 Visualization Element 'Radio Buttons' ..... 3161
6.4.5.20.1.22 Visualization Element 'Alarm Table' ..... 3166
6.4.5.20.1.23 Visualization Element 'Alarm Banner' ..... 3175
6.4.5.20.1.24 Visualization Element 'Bar Display' ..... 3181
6.4.5.20.1.25 Visualization Element 'Meter \(90^{\circ}\). ..... 3187
6.4.5.20.1.26 Visualization Element 'Meter 180 \({ }^{\circ}\) ..... 3194
6.4.5.20.1.27 Visualization Element 'Meter' ..... 3201
6.4.5.20.1.28 Visualization Element 'Potentiometer'. ..... 3208
6.4.5.20.1.29 Visualization Element 'Histogram' ..... 3216
6.4.5.20.1.30 Visualization Element 'Image Switcher'. ..... 3221
6.4.5.20.1.31 Visualization Element 'Lamp' ..... 3226
6.4.5.20.1.32 Visualization Element 'Dip Switch', 'Power Switch', 'Push Switch', 'Push Switch LED', 'Rocker Switch'. ..... 3231
6.4.5.20.1.33 Visualization Element 'Rotary Switch' ..... 3235
6.4.5.20.1.34 Visualization Element 'Trace' ..... 3240
6.4.5.20.1.35 Visualization Element 'Trend' ..... 3246
6.4.5.20.1.36 Visualization Element 'Legend' ..... 3254
6.4.5.20.1.37 Visualization Element 'ActiveX' ..... 3258
6.4.5.20.1.38 Visualization Element 'Web Browser' ..... 3262
6.4.5.20.1.39 Visualization Element 'Busy Symbol, Cube' ..... 3266
6.4.5.20.1.40 Visualization Element 'Busy Symbol, Flower' ..... 3270
6.4.5.20.1.41 Visualization Element 'Text Editor' ..... 3274
6.4.5.20.1.42 Visualization Element 'Path3D' ..... 3279
6.4.5.20.1.43 Visualization Element 'Control Panel' ..... 3282
6.4.5.20.1.44 Visualization Element 'Cartesian XY Chart' ..... 3296
6.4.5.20.1.45 Visualization Element 'Date Range Picker' ..... 3301
6.4.5.20.1.46 Visualization Element 'Time Range Picker' ..... 3306
6.4.5.20.1.47 Visualization Element 'Date Picker' ..... 3311
6.4.5.20.1.48 Visualization Element 'Analog Clock' ..... 3317

\section*{Visualization Element 'Rectangle', 'Rounded Rectangle', 'Ellipse'}

Symbol:

Category: "Basic"
The "Rectangle", "Rounded Rectangle", and "Ellipse" are the same type of element. They can be converted into another element type by changing the "Element type" property.

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name"" & \begin{tabular}{l} 
Optional \\
Example: Werkstueck_3 \\
\\
Hint: Assign individual names for elements so that they are found faster in the \\
element list.
\end{tabular} \\
\hline "Type of element" & "Rectangle", "Rounded Rectangle"," "Ellipse" \\
\hline
\end{tabular}

Element property 'Position'

The position defines the location and size of the element in the visualization window. This is based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X\) " & \begin{tabular}{l} 
The x-coordinate of the upper left corner of the element \\
Specified in pixels \\
Example: 10
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
The y-coordinate of the upper left corner of the element \\
Specified in pixels \\
Example: 10
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels \\
Example: 30
\end{tabular} \\
\hline & \begin{tabular}{l} 
Tip: You can change the values in " \(X\) "," "Y"," "Width", and "Height" by dragging the \\
corresponding symbols 草- to another position in the editor.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Angle" & \begin{tabular}{l}
Static angle of rotation (in degrees) \\
Example: 35 \\
The element is displayed rotated in the editor. The point of rotation is the center of the element. A positive value rotates clockwise. \\
Tip: You can change the value in the editor by focusing the element to the handle. When the cursor is displayed as a rotating arrow \(\mathcal{A}\), you can rotate the element about its center as a handle. \\
(1): Handle \\
Note: If a dynamic angle of rotation is also configured in the property "Absolute movement \(\rightarrow\) Internal rotation", then the static and dynamic angles of rotation are added in runtime mode. The static angle of rotation acts as an offset.
\end{tabular} \\
\hline
\end{tabular}

See also
- *) Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Radius setting Visible only when "Rounded Rectangle" is selected in the "Type of element" property.
\begin{tabular}{|l|l|}
\hline "Radius" & \begin{tabular}{l} 
Rounding of the corners. \\
"From style" \\
"Relative to the element size" \\
"Explicit": Allows for specifying a custom value in the "Value" setting.
\end{tabular} \\
\hline "Value" & \begin{tabular}{l} 
Radius of the rounded corners (in pixels) \\
Example: 5 \\
Requirement: "Explicit" is selected in the "Radius" setting.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center' rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline\(" X "\) & X-coordinate of the point of rotation \\
\hline\(" Y "\) & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus})\) to other positions in the editor.

\section*{Element property 'Colors'}
\begin{tabular}{|l|l|}
\hline "Normal state" & \begin{tabular}{l} 
The normal state is in effect if the variable in "Color variables \(\rightarrow\) Toggle color" is \\
not defined or it has the value FALSE.
\end{tabular} \\
\hline "Frame color" & Frame and fill color for the corresponding state of the variable. \\
\hline "Fill color" & \\
\hline "Transparency" & \begin{tabular}{l} 
Transparency value (0 to 255) for defining the transparency of the selected color. \\
Example: \(255:\) The color is opaque. 0: The color is completely transparent.
\end{tabular} \\
\hline "Alarm state" & \begin{tabular}{l} 
The alarm state is in effect if the variable in "Color variables \(\rightarrow\) Toggle color" has \\
the value TRUE.
\end{tabular} \\
\hline "Use gradient color" & \(\boxed{\checkmark}:\) The element is displayed with a gradient of two colors. \\
\hline "Gradient setting" & The "Gradient editor" dialog box opens. \\
\hline
\end{tabular}

See also
- \({ }^{4}\) ) Chapter 6.4.5.21.3.5 "Dialog 'Gradient Editor'" on page 3369

Element prop- The properties contain fixed values for setting the look of the element. erty 'Appear-
ance'
\begin{tabular}{|c|c|}
\hline "Line width" & \begin{tabular}{l}
Value in pixels \\
Example: 2 \\
Note: The values 0 and 1 both result in a line weight of 1 pixel. If no line should be displayed, then the "Line style" property must be set to the option "Invisible".
\end{tabular} \\
\hline "Fill attributes" & \begin{tabular}{l}
The way in which the element is filled. \\
- "Filled":The element is filled with the color from property "Colors \(\rightarrow\) Fill color". \\
- "Invisible": The fill color is invisible.
\end{tabular} \\
\hline "Line style" & \begin{tabular}{l}
Type of line representation \\
- "Solid" \\
- "Dashes" \\
- "Dots" \\
- "Dash Dot" \\
- "Dash Dot Dot" \\
- "not visible"
\end{tabular} \\
\hline
\end{tabular}

You can assign variables in the "Appearance variables" property for controlling the appearance dynamically. The fixed values here are overwritten.

\section*{See also}
- *y "Element property 'Appearance variables'" on page 3051

Element prop- The properties contains character strings for labeling the element. The character string can also erty 'Texts' contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.
CODESYS accepts the specified texts automatically into the "GlobalTextList" text list. Therefore, these texts can be localized.
\begin{tabular}{|l|l|}
\hline "Text" & \begin{tabular}{l} 
Character string (without single straight quotation marks) for the labeling the \\
element. Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter]. \\
Example: Accesses : \%i \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Text".
\end{tabular} \\
\hline "Tooltip" & \begin{tabular}{l} 
Character string (without single straight quotation marks) that is displayed as the \\
tooltip of an element. \\
Example: Number of valid accesses. \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Tooltip".
\end{tabular} \\
\hline
\end{tabular}

See also
- *y "Element property 'Text variables'" on page 2993
- \(\xi^{4}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872
- \({ }^{4}\) Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329

Element prop- The properties contain fixed values for the text properties.
erty 'Text prop-
erties'
\begin{tabular}{|c|c|}
\hline "Horizontal alignment" & Horizontal alignment of the text within the element. \\
\hline "Vertical alignment" & Vertical alignment of the text within the element. \\
\hline "Text format" & \begin{tabular}{l}
Definition for displaying texts that are too long \\
- "Default": The long text is truncated. \\
- "Line break": The text is split into parts. \\
- "Ellipsis": The visible text ends with "..." indicating that it is not complete.
\end{tabular} \\
\hline "Font" & Example: "Default"
: The "Font" dialog box opens.
Drop-down list with style fonts. \\
\hline "Font color" & Example: "Black"
: The "Color" dialog box opens.
: Drop-down list with style colors. \\
\hline "Transparency" & \begin{tabular}{l}
Whole number (value range from 0 to 255). This determines the transparency of the respective color. \\
Example: 255: The color is opaque. \\
0 : The color is completely transparent. \\
Please note: If the color is a style color and already has a transparency value, then this property is write-protected.
\end{tabular} \\
\hline
\end{tabular}

\section*{Element property 'Absolute movement'}

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" \\
\hline " \(X "\) & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG. iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Y" & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG. iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\oplus\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
\hline "Scaling" & \begin{tabular}{l} 
Variable (integer data type). Causes centric stretching. \\
Example: PLC_PRG. iScaling. \\
The reference point is the "Center" property. \\
The value 1 shrinks the element by a factor of \(0.001 . ~ T h e ~ v a l u e ~\)
\end{tabular} 1000 returns the element \\
to its original size.
\end{tabular}

You can link the variables to a unit conversion.

See also

Element property 'Relative movement'

The properties contains variables for moving the element. The reference point is the position of the element ("Position" property). The shape of the element can change.
\begin{tabular}{|l|l|}
\hline "Movement top-left" & \multicolumn{1}{|c|}{\begin{tabular}{|l|l|}
\hline " & \\
\hline "Y" & \begin{tabular}{l} 
Variable (integer data type). It contains the number (in pixels) that the left edge \\
is moved horizontally. Incrementing the value moves the element to the right. \\
Example: PLC__PRG. iDeltax
\end{tabular} \\
\hline "Movement bottom-right" & \begin{tabular}{l} 
Variable (integer data type). It contains the number (in pixels) that the top edge \\
is moved vertically. Incrementing the value moves the element to the down. \\
Example: PLC_PRRG. iDeltaY
\end{tabular} \\
\hline "X" & \begin{tabular}{l} 
Variable (integer data type). It contains the number (in pixels) that the right edge \\
is moved horizontally. Incrementing the value moves the element to the right. \\
Example: PLC_PRG. iDeltaWidth
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Variable (integer data type). It contains the number (in pixels) that the bottom \\
edge is moved vertically. Incrementing the value moves the element to the down. \\
Example: PLC_PRG. iDeltaHeight
\end{tabular} \\
\hline
\end{tabular}} \\
\hline
\end{tabular}

See also
- «2 "Element property 'Absolute movement"' on page 2991

Element prop- These properties are variables with contents that replace a format definition. erty 'Text variables'
\begin{tabular}{|l|l|}
\hline "Text variable" & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses
\end{tabular} \\
& \begin{tabular}{l} 
Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Text". \\
Note: If you specify a variable of type enumeration with text list support, then \\
the name of the enumeration data type is added automatically in angle brackets \\
after the variable name. Example: PLC_PRG.enVar <enume ration name>. \\
Then the symbolic value of the enumeration component is printed instead of the \\
numeric value when text is printed. Refer to the help page for the enumerations.
\end{tabular} \\
\hline & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses InTool tip \\
Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Tooltip".
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{\text {• }}\) Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329
- ⓨ "Element property 'Texts'" on page 2990
- \(\quad\) • 2 Chapter 6.4.1.20.5.18 "Enumerations" on page 2263

Element prop- Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is erty 'Dynamic texts'
\begin{tabular}{|c|c|}
\hline "Text list" & \begin{tabular}{l}
Variable (string) or name of the text list as a fixed string in single straight quotation marks. \\
Example: 'Errorlist' \\
Drop-down list with the dialogs available in the text lists.
\end{tabular} \\
\hline "Text index" & \begin{tabular}{l}
Text list ID. This refers to the desired output text. \\
- As fixed string with the ID in single straight quotation marks. \\
Example: '1' \\
- As a variable (STRING) for dynamically controlling the text output. \\
Example: strTextID \\
Sample assignment: PLC_PRG.strTextID := '1';
\end{tabular} \\
\hline "Tooltip index" & \begin{tabular}{l}
Text list ID. This refers to the desired output text. \\
- As fixed string with the ID in single straight quotation marks. \\
Example: '2' \\
- As a variable (STRING) for dynamically controlling the text output. \\
Example: strToolTipID \\
Sample assignment: PLC_PRG.strToolTipID := '2';
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{*}\) Chapter 6.4.1.21.2.28 "Object 'Text List'" on page 2532

Element prop- The variables allow for dynamic control of the text display.
erty 'Font varia-
bles'
\begin{tabular}{|c|c|}
\hline "Font name" & \begin{tabular}{l}
Variable (STRING). Includes the font of the text. \\
Example: PLC_PRG.stFontVar := 'Arial'; \\
The selection of fonts corresponds to the default "Font" dialog.
\end{tabular} \\
\hline "Size" & \begin{tabular}{l}
Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name. \\
- <pt>: Points (default) \\
Example: PLC_PRG.iFontHeight <pt> \\
Code: iFontHeight : INT := 12; \\
- <px>: Pixels \\
Example: PLC_PRG.iFontHeight <px> \\
Code: iFontHeight : INT := 19; \\
If you click in the value field, a drop-down list opens on the right for setting the unit. \\
Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property "Text property \(\rightarrow\) Font".
\end{tabular} \\
\hline "Flags" & \begin{tabular}{l}
Variable (DWORD). Contains the flags for displaying fonts. \\
Flags: \\
- 1: Italics \\
- 2: Bold \\
- 4: Underline \\
- 8: Strikethrough \\
Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: PLC_PRG.dwFontType :=6;
\end{tabular} \\
\hline "Character set" & \begin{tabular}{l}
Variable (DWORD). Contains a character set number for the font. \\
The selection of character set numbers corresponds to the "Script" setting of the standard "Font" dialog.
\end{tabular} \\
\hline "Color" & \begin{tabular}{l}
Variable (DWORD). Includes the color of the text. \\
Example: PLC_PRG.dwColorFont:= 16\#FF000000;
\end{tabular} \\
\hline "Flags for text alignment" & \begin{tabular}{l}
Variable (integer data type). Contains the coding for text alignment. \\
Example: PLC_PRG.dwTextAlignment. \\
Coding: \\
- 0 : Top left \\
- 1: Horizontal center \\
- 2: Right \\
- 4: Vertical center \\
- 8: Bottom \\
Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: PLC_PRG.dwFontType := 5;
\end{tabular} \\
\hline
\end{tabular}

Fixed values for displaying texts are set in "Text properties".

See also
- © "Element property 'Text properties"' on page 2991

Element prop- The Element property is used as an interface for project variables to dynamically control colors erty 'Color varia- at runtime. bles'
\begin{tabular}{|c|c|}
\hline "Toggle color" & \begin{tabular}{l}
The property controls the toggled color at runtime. \\
Value assignment: \\
- FALSE: The element is displayed with the color specified in the "Color" property. \\
- TRUE: The element is displayed with the color specified in the "Alarm color" property. \\
Assignment options: \\
- Placeholder for the user input variable \\
- "<toggle/tap variable>" \\
- "<NOT toggle/tap variable>" \\
The color change is not controlled by its own variable, but by a user input variable. \\
Note: Specify a variable for the mouse events "Tap" or "Toggle" in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both "Toggle" and "Tap", then the variable specified in "Tap" is used. \\
Hint: Click the symbol to insert the placeholder "<toggle/tap variable>". When you activate the "Inputconfiguration", "Tap FALSE" property, then the "<NOT toggle/tap variable>" placeholder is displayed. \\
- Instance path of a project variable (BOOL) \\
Example: PLC_PRG.xColorIsToggeled \\
Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.
\end{tabular} \\
\hline \begin{tabular}{l}
"Normal state" \\
"Alarm state"
\end{tabular} & The properties listed below control the color depending on the state. The normal state is in effect if the variable in "Color variables", "Toggle color" is not defined or it has the value FALSE. The alarm state is in effect if the variable in "Colorvariables", "Toggle color" has the value TRUE. \\
\hline "Frame color" & \begin{tabular}{l}
Assignment options: \\
- Variable (DWORD) for the frame color Example: PLC_PRG.dwBorderColor \\
- Color literal Example of green and opaque: 16\#FF00FF00
\end{tabular} \\
\hline "Filling color" & \begin{tabular}{l}
Assignment options: \\
- Variable (DWORD) for the fill color \\
Example: PLC_PRG.dwFillColor \\
- Color literal \\
Example of gray and opaque: \(16 \#\) FF888888
\end{tabular} \\
\hline
\end{tabular}

The transparency part of the color value is evaluated only if the "Activate semitransparent drawing" option of the visualization manager is selected.

Select the "Advanced" option in the toolbar of the properties view. Then all element properties are visible.

See also
- 乡 Chapter 6.4.5.10.4 "Animating a color display" on page 2914

Element prop- The properties contain IEC variables for controlling the appearance of the element dynamically. erty 'Appearance variables'
\begin{tabular}{|l|l|}
\hline "Line width" & Variable (integer data type). Contains the line weight (in pixels). \\
\hline "Fill attributes" & Variable (DWORD). Controls whether the fill color of the element is visible. \\
& - Variable value = 0: Filled \\
& - Variable value > 0: Invisible; no fill color \\
\hline "Line style" & Variable (DWORD). Controls the line style. \\
& Coding: \\
& - 0 : Solid line \\
& - 1: Dashed line \\
& - \(2:\) Dotted line \\
& - 3: Line type "Dash Dot" \\
& - 3: Line type "Dash Dot Dot" \\
& - 8: Invisible; no line \\
\hline
\end{tabular}

Fixed values can be set in the "Appearance" property. These values can be overwritten by dynamic variables at runtime.

See also
- * "Element property 'Appearance"' on page 3002

Element prop- The variables control the element behavior dynamically.
erty 'State varia-
bles'
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \(:=\) FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

\section*{The "Invisible" property is supported by the "Client Animation" functionality.}

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", "X", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain the configurations for the user input when using the mouse or keyboard. erty 'Input con- A user input defines an event and one or more actions that are executed when an event occurs. figuration'

The "Configure" button opens the "Input Configuration" dialog. There you can create or edit user inputs.
Configured user inputs are listed below the events. They each include the action that is triggered and the setting in short form.

Example: "Execute ST Code": З PLC_PRG.i_x := 0;
\begin{tabular}{|l|l|}
\hline "OnDialogClosed" & Input event: The user closes the dialog. \\
\hline "OnMouseClick" & \begin{tabular}{l} 
Input event: The user clicks the mouse button completely in the element area. \\
The mouse button is clicked and released.
\end{tabular} \\
\hline "OnMouseDown" & Input event: The user clicks down on the mouse button. \\
\hline "OnMouseEnter" & Input event: The user drags the mouse pointer to the element. \\
\hline "OnMouseLeave" & Input event: The user drags the mouse pointer away from the element. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "OnMouseMove" & Input event: The user moves the mouse pointer over the element area. \\
\hline "OnMouseUp" & \begin{tabular}{l} 
Input events: \\
- \begin{tabular}{l} 
The user releases the mouse button within the element area. It is irrelevant \\
whether the user has previously pressed the mouse button inside or outside \\
the element area.
\end{tabular} \\
- The user presses the mouse button within the element area, leaves the \\
element area, and then releases the mouse button. \\
Note: This CODESYS-specific triggering behavior guarantees that actions for \\
key elements are completed. A key element starts an action for "OnMouseDown"" \\
and ends the action for "OnMouseUp". \\
Example: A visualization user presses the mouse button within the element area \\
of the key element and then moves the cursor position so that it lies outside the \\
element area. The action is ended anyway because "OnMouseUp" is triggered.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Tap" & \begin{tabular}{l} 
When a mouse click event occurs, the variable defined in "Variable" is described \\
in the application. The coding depends on the "Tap FALSE" and "Tap on enter if \\
captured" options.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL) that is set on mouse click event. \\
Example: PLC_PRG. bIsTapped \\
TRUE: A mouse click event exists. It lasts as long as the user presses the mouse \\
button over the element. It ends when the button is released. \\
FALSE: A mouse click event does not exist. \\
Requirement: The "Tap FALSE" option is not activated.
\end{tabular} \\
\hline "Tap FALSE" & \begin{tabular}{l} 
V: The mouse click event leads to a complementary value in "Variable". \\
TRUE: A mouse click event does not exist.
\end{tabular} \\
\hline "Tap on enter if captured" & \begin{tabular}{l} 
FALSE: While the mouse click event exists.
\end{tabular} \\
\hline \begin{tabular}{l} 
pointer is dragged within the element area or not while the mouse button is \\
pressed. \\
TRUE: While the mouse click event exists and the mouse pointer is moved over \\
the element area. \\
FALSE: A mouse click event does not exist. Or the user moves the mouse \\
pointer outside of the element area while the mouse button is pressed. \\
The value is TRUE again as soon as the user moves the pointer back to the \\
element area. The mouse is then captured.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Toggle" & \begin{tabular}{l} 
With the onset of a mouse click event, the variable is set; when the mouse click \\
event is completed, the variable is reset.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL). Its value toggled when the mouse click event is ended. This is \\
when the user releases the mouse button while the mouse pointer is over the \\
element area. \\
If the user releases the mouse button while the mouse pointer is outside of the \\
element area, then the mouse click event is not ended and the value is not \\
toggled. \\
Hint: The user can cancel a started toggle input by dragging the mouse pointer \\
out of the element area.
\end{tabular} \\
\hline "Toggle on up if captured" & \begin{tabular}{l} 
V: The value toggles regardless of where the mouse pointer is when the mouse \\
button is released. The mouse is then captured.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Hotkey" & \begin{tabular}{l}
Keyboard shortcut on the element for triggering specific input actions. \\
When the keyboard shortcut event occurs, the input actions in the "Events" property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.
\end{tabular} \\
\hline "Key" & \begin{tabular}{l}
Key pressed for input action. \\
Example: [T] \\
Note: The following properties appear when a key is selected.
\end{tabular} \\
\hline "Events" & \begin{tabular}{l}
- "None" \\
- "Mouse down": Pressing the key triggers the input actions that are configured in the "OnMouseDown" property. \\
- "Mouse up": Releasing the key triggers the input actions that are configured in the "OnMouseUp" property. \\
- "Mouse down/up": Pressing and releasing the key triggers the input actions that are configured in the "OnMouseDown" property and the "OnMouseUp" property.
\end{tabular} \\
\hline "Shift" & Combination with the Shift key Example: [Shift]+[T]. \\
\hline "Control" & Combination with the Ctrl key Example: [Ctrl]+[T]. \\
\hline "Alt" & Combination with the Alt key Example: \([A I t]+[T]\). \\
\hline
\end{tabular}

All keyboard shortcuts and their actions that are configured in the visualization are listed on the "Keyboard Configuration" tab.

\section*{See also}
- ⓨ Chapter 6.4.5.21.2.2 "Command 'Keyboard Configuration'" on page 3341
- \(\xi^{2}\) Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration'" on page 3370

Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'
\begin{tabular}{|c|c|}
\hline "Access rights" & \begin{tabular}{l}
Opens the "Access rights" dialog. There you can edit the access privileges for the element. \\
Status messages: \\
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
- "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline \multicolumn{2}{|l|}{See also} \\
\hline & 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366 \\
\hline
\end{tabular}

See also
- \({ }^{4}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Line'}

Symbol:

\section*{Category: "Basic"}

The element draws a simple line.

\section*{Element proper- \\ ties}
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Optional. \\
Example: Separator_Header_ \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list.
\end{tabular} \\
\hline "Type of element" & "ine" \\
\hline
\end{tabular}

Element prop- The following properties define the position and length of the element in the visualization erty 'Position' window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical y -axis runs downwards.
\begin{tabular}{|c|c|}
\hline "Dots" & \begin{tabular}{l}
" \([0]\) ": Coordinates of the starting point \\
" \([1]\) ": Coordinate of the end point \\
You can also change the values by dragging the box symbols (ए) to other positions in the editor.
\end{tabular} \\
\hline "Angle" & \begin{tabular}{l}
Static angle of rotation (in degrees). \\
Example: 35 \\
The element is displayed rotated in the editor. The point of rotation is the center of the element. A positive value rotates clockwise. \\
Tip: You can change the value in the editor by focusing the element to the handle. When the cursor is displayed as a rotating arrow \(\boldsymbol{A}^{\text {, }}\), you can rotate the element about its center as a handle. \\
Example: \\
(1): Handle \\
Note: If a dynamic angle of rotation is also configured in the property "Absolute movement \(\rightarrow\) Internal rotation", then the static and dynamic angles of rotation are added in runtime mode. The static angle of rotation acts as an offset.
\end{tabular} \\
\hline
\end{tabular}

See also
- ㅋy Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center'
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline " \(Y\) " & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols (ఉ) to other positions in the editor.

Element prop- The properties contain fixed values for setting colors. erty 'Colors'
\begin{tabular}{|l|l|}
\hline "Color" & \begin{tabular}{l} 
Color of the line in normal state. \\
Please note that the normal state is in effect if the expression in the \\
"Color variables \(\rightarrow\) Toggle color" property is not defined or it has the value \\
FALSE.
\end{tabular} \\
\hline "Alarm color" & \begin{tabular}{l} 
Color of the line in alarm state. \\
Please note that the alarm state is in effect if the expression in the \\
"Color variables \(\rightarrow\) Toggle color" property has the value TRUE.
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l} 
Value (0 to 255\()\) for defining the transparency of the selected color. \\
Example \(255:\) The color is opaque. \(0:\) The color is completely transparent.
\end{tabular} \\
\hline
\end{tabular}

See also
- 乡y Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

Element prop- The properties contain fixed values for setting the look of the element.
erty 'Appearance'

Value in pixels
Example: 2
Note: The values 0 and 1 both result in a line weight of one pixel. If no line should be displayed, then the "Line style" property must be set to the option "Invisible".
Type of line representation
- "Solid"
- "Dashes"
- "Dots"
- "Dash Dot"
- "Dash Dot Dot"
- "not visible"
\begin{tabular}{|c|c|}
\hline "Line width" & \begin{tabular}{l}
Value in pixels \\
Example: 2 \\
Note: The values 0 and 1 both result in a line weight of one pixel. If no line should be displayed, then the "Line style" property must be set to the option "Invisible".
\end{tabular} \\
\hline "Line style" & \begin{tabular}{l}
Type of line representation \\
- "Solid" \\
- "Dashes" \\
- "Dots" \\
- "Dash Dot" \\
- "Dash Dot Dot" \\
- "not visible"
\end{tabular} \\
\hline
\end{tabular}

You can assign variables in the "Appearance variables" property for controlling the appearance dynamically. The fixed values are defined here.

\section*{See also}
- ③ "Element property 'Appearance variables'" on page 3051

Element prop- The properties contains character strings for labeling the element. The character string can also erty 'Texts' contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.
CODESYS accepts the specified texts automatically into the "GlobalTextList" text list. Therefore, these texts can be localized.
\begin{tabular}{|l|l|}
\hline "Text" & \begin{tabular}{l} 
Character string (without single straight quotation marks) for the labeling the \\
element. Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter]. \\
Example: Accesses: \%i \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Text".
\end{tabular} \\
\hline "Tooltip" & \begin{tabular}{l} 
Character string (without single straight quotation marks) that is displayed as the \\
tooltip of an element. \\
Example: Number of valid accesses. \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Tooltip".
\end{tabular} \\
\hline
\end{tabular}

See also
- «̌ "Element property 'Text variables'" on page 3005
- \(\sum^{*}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872
- \({ }^{\text {• }}\) Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329

Element prop- The properties contain fixed values for the text properties. erty 'Text properties'
\begin{tabular}{|c|c|}
\hline "Horizontal alignment" & Horizontal alignment of the text within the element. \\
\hline "Vertical alignment" & Vertical alignment of the text within the element. \\
\hline "Text format" & \begin{tabular}{l}
Definition for displaying texts that are too long \\
- "Default": The long text is truncated. \\
- "Line break": The text is split into parts. \\
- "Ellipsis": The visible text ends with "..." indicating that it is not complete.
\end{tabular} \\
\hline "Font" & \begin{tabular}{l}
Example: "Default"
: The "Font" dialog box opens. \\
*: Drop-down list with style fonts.
\end{tabular} \\
\hline "Font color" & \begin{tabular}{l}
Example: "Black"
: The "Color" dialog box opens. \\
*: Drop-down list with style colors.
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l}
Whole number (value range from 0 to 255). This determines the transparency of the respective color. \\
Example: 255: The color is opaque. \\
0 : The color is completely transparent. \\
Please note: If the color is a style color and already has a transparency value, then this property is write-protected.
\end{tabular} \\
\hline
\end{tabular}

\section*{Element property 'Absolute movement'}

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|c|c|c|}
\hline \multicolumn{3}{|l|}{"Movement"} \\
\hline "X" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular}} \\
\hline "Y" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the \(Y\) position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular}} \\
\hline "Rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle1. \\
The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol. \\
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
\end{tabular} &  \\
\hline "Scaling" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (integer data type). Causes centric stretching. \\
Example: PLC_PRG.iScaling. \\
The reference point is the "Center" property. \\
The value 1 shrinks the element by a factor of 0.001 . The value 1000 returns the element to its original size.
\end{tabular}} \\
\hline "Interior rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle2. \\
In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol. \\
Note: If a static angle of rotation is specified in the property "Position \(\rightarrow\) Angle", then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.
\end{tabular} &  \\
\hline "Use REAL values" & \begin{tabular}{l}
Note: Only available if the device supports the use of REAL coordinates. \\
\(\square\) : The properties of the absolute movement are interpreted as REAL values. The values are not rounded. \\
The option allows for the individual fine-tuning of drawing the element, for example for the visualization of a smoother rotation. \\
Hint: If a horizontal or vertical line is drawn blurry on a specific visualization platform, then this can be corrected by an offset of 0.5 px in the direction of the line thickness.
\end{tabular} & \\
\hline
\end{tabular}
 erty 'Relative

Variable (numeric data type). It contains the number (in pixels) that the starting point of the line is moved.
Incrementing the \(X\) value moves the element to the right. Incrementing the \(Y\) value moves the element to the down.
Variable (numeric data type). It contains the number (in pixels) that the end point of the line is moved.

Incrementing the \(X\) value moves the element to the right. Incrementing the Y value moves the element to the down.

See also
- (3) "Element property 'Absolute movement'" on page 3003

Element prop- These properties are variables with contents that replace a format definition. erty 'Text variables'
\begin{tabular}{|l|l|}
\hline "Text variable" & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses \\
Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Text". \\
Note: If you specify a variable of type enumeration with text list support, then \\
the name of the enumeration data type is added automatically in angle brackets \\
after the variable name. Example: PLC_PRG.enVar <enumeration name>. \\
Then the symbolic value of the enumeration component is printed instead of the \\
numeric value when text is printed. Refer to the help page for the enumerations.
\end{tabular} \\
\hline "Tooltip variable" & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses InTooltip
\end{tabular} \\
& Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Tooltip". \\
\hline
\end{tabular}

See also
- \({ }^{4}\) Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329
- « "Element property 'Texts'" on page 3003
- \(\overbrace{}^{*}\) Chapter 6.4.1.20.5.18 "Enumerations" on page 2263

Element prop- Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is erty 'Dynamic texts' currently indexed in the variable.
\begin{tabular}{|l|l|}
\hline "Text list" & \begin{tabular}{l} 
Variable (string) or name of the text list as a fixed string in single straight quota- \\
tion marks. \\
Example: ' Errorlist'
\end{tabular} \\
& : Drop-down list with the dialogs available in the text lists.
\end{tabular}

See also
- \({ }^{\text {• }}\) Chapter 6.4.1.21.2.28 "Object 'Text List"' on page 2532

Element prop- The variables allow for dynamic control of the text display. erty 'Font variables'
\begin{tabular}{|c|c|}
\hline "Font name" & \begin{tabular}{l}
Variable (STRING). Includes the font of the text. \\
Example: PLC_PRG.stFontVar := 'Arial'; \\
The selection of fonts corresponds to the default "Font" dialog.
\end{tabular} \\
\hline "Size" & \begin{tabular}{l}
Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name. \\
- <pt>: Points (default) \\
Example: PLC_PRG.iFontHeight <pt> \\
Code: iFontHeight : INT := 12; \\
- <px> : Pixels \\
Example: PLC_PRG.iFontHeight <px> \\
Code: iFontHeight : INT := 19; \\
If you click in the value field, a drop-down list opens on the right for setting the unit. \\
Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property "Text property \(\rightarrow\) Font".
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Flags" & \begin{tabular}{l}
Variable (DWORD). Contains the flags for displaying fonts. \\
Flags: \\
- 1: Italics \\
- 2: Bold \\
- 4: Underline \\
- 8: Strikethrough \\
Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: PLC_PRG.dwFontType :=6;
\end{tabular} \\
\hline "Character set" & \begin{tabular}{l}
Variable (DWORD). Contains a character set number for the font. \\
The selection of character set numbers corresponds to the "Script" setting of the standard "Font" dialog.
\end{tabular} \\
\hline "Color" & \begin{tabular}{l}
Variable (DWORD). Includes the color of the text. \\
Example: PLC_PRG.dwColorFont:= 16\#FF000000;
\end{tabular} \\
\hline "Flags for text alignment" & \begin{tabular}{l}
Variable (integer data type). Contains the coding for text alignment. \\
Example: PLC_PRG.dwTextAlignment. \\
Coding: \\
- 0 : Top left \\
- 1: Horizontal center \\
- 2: Right \\
- 4: Vertical center \\
- 8: Bottom \\
Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: PLC_PRG.dwFontType := 5;
\end{tabular} \\
\hline
\end{tabular}

Fixed values for displaying texts are set in "Text properties".

See also
- «ヶ "Element property 'Text properties'" on page 3003

Element prop- The Element property is used as an interface for project variables to dynamically control colors erty 'Color varia- at runtime. bles'
\begin{tabular}{|c|c|}
\hline "Toggle color" & \begin{tabular}{l}
The property controls the toggled color at runtime. \\
Value assignment: \\
- FALSE: The element is displayed with the color specified in the "Color" property. \\
- TRUE: The element is displayed with the color specified in the "Alarm color" property. \\
Assigning the property: \\
- Placeholder for the user input variable \\
- "<toggle/tap variable>" \\
- "<NOT toggle/tap variable>" \\
The color change is not controlled by its own variable, but by a user input variable. \\
Note: Specify a variable for the mouse events "Tap" or "Toggle" in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both "Toggle" and "Tap", then the variable specified in "Tap" is used. \\
Hint: Click the symbol to insert the placeholder "<toggle/tap variable>". When you activate the "Inputconfiguration", "Tap FALSE" property, then the "<NOT toggle/tap variable>" placeholder is displayed. \\
- Instance path of a project variable (BOOL) \\
Example: PLC_PRG.xColorIsToggeled \\
Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.
\end{tabular} \\
\hline "Color" & \begin{tabular}{l}
- Variable (DWORD) for the color \\
Example: PLC_PRG.dwColor \\
- Color literal \\
Example of gray and opaque: 16\#FF888888 \\
Please note that the normal state is in effect if the expression in the "Colorvariables \(\rightarrow\) Toggle color" property is not defined or it has the value FALSE.
\end{tabular} \\
\hline "Alarm color" & \begin{tabular}{l}
Color variable in the alarm state \\
- Variable (DWORD) for the alarm color \\
Example: PLC_PRG.dwAlarmColor \\
- Color literal \\
Example of red and opaque: 16\#FFFF0000 \\
Please note that the alarm state is in effect if the expression in the "Colorvariables \(\rightarrow\) Toggle color" property has the value TRUE.
\end{tabular} \\
\hline
\end{tabular}

The transparency part of the color value is evaluated only if the "Activate semitransparent drawing" option of the visualization manager is selected.

Select the "Advanced" option in the toolbar of the properties view. Then all element properties are visible.

\section*{See also}
- \& Chapter 6.4.5.10.4 "Animating a color display" on page 2914
- * Chapter 6.4.5.21.4.2 "Object 'Visualization manager'" on page 3398

Element prop- The variables control the element behavior dynamically. erty 'State variables'
\begin{tabular}{|l|l|}
\hline "Invisible" & Variable (BOOL). Toggles the visibility of the element. \\
& \begin{tabular}{l} 
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \(=\) FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

Element prop- Dynamic definition of the weight of a line element using a variable. erty 'Line width variable'
\begin{tabular}{|l|l|}
\hline "Integer value" & \begin{tabular}{l} 
Variable (integer data type). Defines the line weight of the element (in pixels)., \\
This overwrites the fixed value that is defined in "Appearance \(\rightarrow\) Line weight". \\
Note: The value 0 codes the same as 1 and sets the line weight to one pixel.
\end{tabular} \\
\hline
\end{tabular}

\section*{Element property 'Line style variable'}
\begin{tabular}{|l|l|}
\hline "Integer value"" & Variable (integer data type). Defines the appearance of the line at runtime. \\
& \(\bullet\)\begin{tabular}{ll} 
1: Solid \\
\(\bullet\) & 2: Dashes \\
\(\bullet\) & 3: Dots \\
& \(\bullet\) \\
& 4: Dash Dot \\
& \(\bullet\) \\
& 5: Dash Dot Dot \\
& 6: Invisible: The line is not drawn. \\
\hline
\end{tabular} \\
\hline
\end{tabular}

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain the configurations for the user input when using the mouse or keyboard. erty 'Input con- A user input defines an event and one or more actions that are executed when an event occurs. figuration'

The "Configure" button opens the "Input Configuration" dialog. There you can create or edit user inputs.
Configured user inputs are listed below the events. They each include the action that is triggered and the setting in short form.

Example: "Execute ST Code": ßPLC_PRG.i_x := 0;
\begin{tabular}{|l|l|}
\hline "OnDialogClosed" & Input event: The user closes the dialog. \\
\hline "OnMouseClick" & \begin{tabular}{l} 
Input event: The user clicks the mouse button completely in the element area. \\
The mouse button is clicked and released.
\end{tabular} \\
\hline "OnMouseDown" & Input event: The user clicks down on the mouse button. \\
\hline "OnMouseEnter" & Input event: The user drags the mouse pointer to the element. \\
\hline "OnMouseLeave" & Input event: The user drags the mouse pointer away from the element. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "OnMouseMove" & Input event: The user moves the mouse pointer over the element area. \\
\hline "OnMouseUp" & \begin{tabular}{l} 
Input events: \\
- \begin{tabular}{l} 
The user releases the mouse button within the element area. It is irrelevant \\
whether the user has previously pressed the mouse button inside or outside \\
the element area.
\end{tabular} \\
- The user presses the mouse button within the element area, leaves the \\
element area, and then releases the mouse button. \\
Note: This CODESYS-specific triggering behavior guarantees that actions for \\
key elements are completed. A key element starts an action for "OnMouseDown"" \\
and ends the action for "OnMouseUp". \\
Example: A visualization user presses the mouse button within the element area \\
of the key element and then moves the cursor position so that it lies outside the \\
element area. The action is ended anyway because "OnMouseUp" is triggered.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Tap" & \begin{tabular}{l} 
When a mouse click event occurs, the variable defined in "Variable" is described \\
in the application. The coding depends on the "Tap FALSE" and "Tap on enter if \\
captured" options.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL) that is set on mouse click event. \\
Example: PLC_PRG. bIsTapped \\
TRUE: A mouse click event exists. It lasts as long as the user presses the mouse \\
button over the element. It ends when the button is released. \\
FALSE: A mouse click event does not exist. \\
Requirement: The "Tap FALSE" option is not activated.
\end{tabular} \\
\hline "Tap FALSE" & \begin{tabular}{l} 
V: The mouse click event leads to a complementary value in "Variable". \\
TRUE: A mouse click event does not exist.
\end{tabular} \\
\hline "Tap on enter if captured" & \begin{tabular}{l} 
FALSE: While the mouse click event exists.
\end{tabular} \\
\hline \begin{tabular}{l} 
pointer is dragged within the element area or not while the mouse button is \\
pressed. \\
TRUE: While the mouse click event exists and the mouse pointer is moved over \\
the element area. \\
FALSE: A mouse click event does not exist. Or the user moves the mouse \\
pointer outside of the element area while the mouse button is pressed. \\
The value is TRUE again as soon as the user moves the pointer back to the \\
element area. The mouse is then captured.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Toggle" & \begin{tabular}{l} 
With the onset of a mouse click event, the variable is set; when the mouse click \\
event is completed, the variable is reset.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL). Its value toggled when the mouse click event is ended. This is \\
when the user releases the mouse button while the mouse pointer is over the \\
element area. \\
If the user releases the mouse button while the mouse pointer is outside of the \\
element area, then the mouse click event is not ended and the value is not \\
toggled. \\
Hint: The user can cancel a started toggle input by dragging the mouse pointer \\
out of the element area.
\end{tabular} \\
\hline "Toggle on up if captured" & \begin{tabular}{l} 
V: The value toggles regardless of where the mouse pointer is when the mouse \\
button is released. The mouse is then captured.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Hotkey" & \begin{tabular}{l}
Keyboard shortcut on the element for triggering specific input actions. \\
When the keyboard shortcut event occurs, the input actions in the "Events" property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.
\end{tabular} \\
\hline "Key" & \begin{tabular}{l}
Key pressed for input action. \\
Example: [T] \\
Note: The following properties appear when a key is selected.
\end{tabular} \\
\hline "Events" & \begin{tabular}{l}
- "None" \\
- "Mouse down": Pressing the key triggers the input actions that are configured in the "OnMouseDown" property. \\
- "Mouse up": Releasing the key triggers the input actions that are configured in the "OnMouseUp" property. \\
- "Mouse down/up": Pressing and releasing the key triggers the input actions that are configured in the "OnMouseDown" property and the "OnMouseUp" property.
\end{tabular} \\
\hline "Shift" & Combination with the Shift key Example: [Shift]+[T]. \\
\hline "Control" & Combination with the Ctrl key Example: [Ctrl]+[T]. \\
\hline "Alt" & Combination with the Alt key Example: \([A I t]+[T]\). \\
\hline
\end{tabular}

All keyboard shortcuts and their actions that are configured in the visualization are listed on the "Keyboard Configuration" tab.

\section*{See also}
- ⓨ Chapter 6.4.5.21.2.2 "Command 'Keyboard Configuration'" on page 3341
- \(\xi^{2}\) Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration'" on page 3370

Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages: \\
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
- "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\(\qquad\)\begin{tabular}{l} 
See also \\
• \& Chapter 6.4.5.21.3.1 "Dialog 'Access Rights"" on page 3366
\end{tabular}
\end{tabular}

See also
- \({ }^{\text {- }}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Polygon', 'Polyline', 'Bézier Curve'}

Symbol:

Category: "Basic"
The "Polygon", "Polyline", and "Bézier Curve" are the same element type. They can be converted into another type by changing the "Element type" property.

Elements can be dragged to the editor. The element is then drawn with five points: [0] to [4]
Other positions are added as follows: Move the mouse pointer over a corner point; the mouse pointer changes shape. Now if you press and hold [Ctrl] and click the left mouse button, another point is created. You can delete a point by pressing and holding [Shift]+[Ctrl] and click the selected point.

As an alternative, you can select the element in the toolbox area and in the editor click multiple times. At the same time, a connecting line is drawn from one point to the other. End by double-clicking the element or right-clicking it one time.

\section*{Element proper- \\ ties}
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Optional. \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list. \\
Example: Werkstueck_1
\end{tabular} \\
\hline "Type of element" & \begin{tabular}{l} 
- "Polygon" \\
- "Polyline" \\
- "Bézier Curve"
\end{tabular} \\
\hline
\end{tabular}

Element prop- The following properties define the position of the corner points in the visualization window. erty 'Position' These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|c|c|}
\hline "Dots" & \begin{tabular}{l}
[0]..[n]: Coordinates of the corner points \\
Specified in pixels \\
You can also change the values by dragging the box symbols (만) to other positions in the editor.
\end{tabular} \\
\hline "Angle" & \begin{tabular}{l}
Static angle of rotation (in degrees). \\
Example: 35 \\
The element is displayed rotated in the editor. The point of rotation is the center of the element. A positive value rotates clockwise. \\
Tip: You can change the value in the editor by focusing the element to the handle. When the cursor is displayed as a rotating arrow \(\mathcal{A}\), you can rotate the element about its center as a handle. \\
(1): Handle \\
Note: If a dynamic angle of rotation is also configured in the property "Absolute movement \(\rightarrow\) Internal rotation", then the static and dynamic angles of rotation are added in runtime mode. The static angle of rotation acts as an offset.
\end{tabular} \\
\hline
\end{tabular}

See also
- 乡y Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center'
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline "Y" & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\Psi})\) to other positions in the editor.

\section*{Element prop- \\ erty 'Colors'}
\begin{tabular}{|l|l|}
\hline "Normal state" & \begin{tabular}{l} 
The normal state is in effect if the variable in "Color variables \(\rightarrow\) Toggle color" is \\
not defined or it has the value FALSE.
\end{tabular} \\
\hline "Frame color" & Frame and fill color for the corresponding state of the variable. \\
\hline "Fill color" & "Transparency" \\
\hline Transparency value (0 to 255) for defining the transparency of the selected color. \\
Example: \(255:\) The color is opaque. \(0:\) The color is completely transparent.
\end{tabular}\(|\)\begin{tabular}{ll} 
"Alarm state" & \begin{tabular}{l} 
The alarm state is in effect if the variable in "Color variables \(\rightarrow\) Toggle color" has \\
the value TRUE.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Use gradient color" & \(\boxed{\square}:\) The element is displayed with a gradient of two colors. \\
\hline "Gradient setting" & The "Gradient editor" dialog box opens. \\
\hline
\end{tabular}

See also
- \({ }^{\text {• }}\) Chapter 6.4.5.21.3.5 "Dialog 'Gradient Editor'" on page 3369

Element prop- The properties contain fixed values for setting the look of the element. erty 'Appearance'
\begin{tabular}{|c|c|}
\hline "Line width" & \begin{tabular}{l}
Value in pixels \\
Example: 2 \\
Note: The values 0 and 1 both result in a line weight of 1 pixel. If no line should be displayed, then the "Line style" property must be set to the option "Invisible".
\end{tabular} \\
\hline "Fill attributes" & \begin{tabular}{l}
The way in which the element is filled. \\
- "Filled":The element is filled with the color from property "Colors \(\boldsymbol{\rightarrow}\) Fill color". \\
- "Invisible": The fill color is invisible.
\end{tabular} \\
\hline "Line style" & \begin{tabular}{l}
Type of line representation \\
- "Solid" \\
- "Dashes" \\
- "Dots" \\
- "Dash Dot" \\
- "Dash Dot Dot" \\
- "not visible"
\end{tabular} \\
\hline
\end{tabular}

You can assign variables in the "Appearance variables" property for controlling the appearance dynamically. The fixed values here are overwritten.

\section*{See also}
- « "Element property 'Appearance variables'" on page 3051

Element prop- The properties contains character strings for labeling the element. The character string can also erty 'Texts' contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.
CODESYS accepts the specified texts automatically into the "GlobalTextList" text list. Therefore, these texts can be localized.
\begin{tabular}{|l|l|}
\hline "Text" & \begin{tabular}{l} 
Character string (without single straight quotation marks) for the labeling the \\
element. Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter]. \\
Example: Accesses: \%i \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Text".
\end{tabular} \\
\hline "Tooltip" & \begin{tabular}{l} 
Character string (without single straight quotation marks) that is displayed as the \\
tooltip of an element. \\
Example: Number of valid accesses. \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Tooltip".
\end{tabular} \\
\hline
\end{tabular}

See also
- « "Element property 'Text variables'" on page 3018
- « Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872
- \({ }^{\bullet}\) Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329

Element prop- The properties contain fixed values for the text properties. erty 'Text properties'
\begin{tabular}{|c|c|}
\hline "Horizontal alignment" & Horizontal alignment of the text within the element. \\
\hline "Vertical alignment" & Vertical alignment of the text within the element. \\
\hline "Text format" & \begin{tabular}{l}
Definition for displaying texts that are too long \\
- "Default": The long text is truncated. \\
- "Line break": The text is split into parts. \\
- "Ellipsis": The visible text ends with "..." indicating that it is not complete.
\end{tabular} \\
\hline "Font" & Example: "Default"
: The "Font" dialog box opens.
Drop-down list with style fonts. \\
\hline "Font color" & Example: "Black"
The "Color" dialog box opens.
Drop-down list with style colors. \\
\hline "Transparency" & \begin{tabular}{l}
Whole number (value range from 0 to 255). This determines the transparency of the respective color. \\
Example: 255: The color is opaque. \\
0 : The color is completely transparent. \\
Please note: If the color is a style color and already has a transparency value, then this property is write-protected.
\end{tabular} \\
\hline
\end{tabular} erty 'Absolute movement'

Element prop- The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline " \({ }^{\text {In }}\) & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG. iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|}
\hline "Rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle1. \\
The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol. \\
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
\end{tabular} &  \\
\hline "Scaling" & \begin{tabular}{l}
Variable (integer data type). Causes centric stretchi Example: PLC_PRG.iScaling. \\
The reference point is the "Center" property. \\
The value 1 shrinks the element by a factor of 0.00 to its original size.
\end{tabular} & The value 1000 returns the element \\
\hline "Interior rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle2. \\
In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. \\
The rotation point is shown as the \(\boldsymbol{\oplus}\) symbol. \\
Note: If a static angle of rotation is specified in the property "Position \(\rightarrow\) Angle", then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.
\end{tabular} &  \\
\hline "Use REAL values" & \begin{tabular}{l}
Note: Only available if the device supports the use of REAL coordinates. \\
\(\checkmark\) : The properties of the absolute movement are interpreted as REAL values. The values are not rounded. \\
The option allows for the individual fine-tuning of drawing the element, for example for the visualization of a smoother rotation. \\
Hint: If a horizontal or vertical line is drawn blurry on a specific visualization platform, then this can be corrected by an offset of 0.5 px in the direction of the line thickness.
\end{tabular} & \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

See also

\section*{Element property 'Dynamic points'}
\begin{tabular}{|l|l|}
\hline "Array of points" & \begin{tabular}{l} 
Variable (POINTER TO). Points to an array of the structure \\
VisuElems.VisuStructPoint. The elements ix and iY of \\
VisuSt ructPoint contain the xy-coordinates of a point The current number of \\
array elements implicitly contains the variable in the property "Number of points". \\
The variable that is assigned to the property "Number of points" contains the \\
number of array elements and therefore the number of corner points. \\
Example: pPoints : PoINTER TO ARRAY [0. .100] OF \\
VisuElems.VisuStructPoint;
\end{tabular} \\
\hline "Number of points" & \begin{tabular}{l} 
Variable (integer data type): Contains the number of array elements and there- \\
fore the number of corner points for displaying the element. \\
Example: PLC_PRG. iNumberOfPoints \(:=24 ;\) \\
In the example, the element has 24 points. This definition is necessary because \\
the individual points are defined by a pointer and this does not allow control over \\
the number of points. \\
Note: In this way, it is possible to adapt the display of the element dynamically by \\
updating the number of corner points.
\end{tabular} \\
\hline
\end{tabular}

Element prop- These properties are variables with contents that replace a format definition.
erty 'Text varia-
bles'
\begin{tabular}{|l|l|}
\hline "Text variable" & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses
\end{tabular} \\
& \begin{tabular}{l} 
Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Text". \\
Note: If you specify a variable of type enumeration with text list support, then \\
the name of the enumeration data type is added automatically in angle brackets \\
after the variable name. Example: PLC_PRG. enVar <enumeration name>. \\
Then the symbolic value of the enumeration component is printed instead of the \\
numeric value when text is printed. Refer to the help page for the enumerations.
\end{tabular} \\
\hline & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses InTooltip
\end{tabular} \\
Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Tooltiable". \\
\hline
\end{tabular}

See also
- \({ }^{2}\) Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329
- *' "Element property 'Texts"' on page 3015
- \#y Chapter 6.4.1.20.5.18 "Enumerations" on page 2263

Element property 'Dynamic texts'

Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is currently indexed in the variable.
\begin{tabular}{|c|c|}
\hline "Text list" & \begin{tabular}{l}
Variable (string) or name of the text list as a fixed string in single straight quotation marks. \\
Example: 'Errorlist' \\
: Drop-down list with the dialogs available in the text lists.
\end{tabular} \\
\hline "Text index" & \begin{tabular}{l}
Text list ID. This refers to the desired output text. \\
- As fixed string with the ID in single straight quotation marks. \\
Example: '1' \\
- As a variable (STRING) for dynamically controlling the text output. \\
Example: strTextID \\
Sample assignment: PLC_PRG.strTextID := '1';
\end{tabular} \\
\hline "Tooltip index" & \begin{tabular}{l}
Text list ID. This refers to the desired output text. \\
- As fixed string with the ID in single straight quotation marks. \\
Example: '2' \\
- As a variable (STRING) for dynamically controlling the text output. \\
Example: strToolTipID \\
Sample assignment: PLC_PRG.strToolTipID := '2';
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{\text {• }}\), Chapter 6.4.1.21.2.28 "Object 'Text List"' on page 2532

Element prop- The variables allow for dynamic control of the text display. erty 'Font variables'
\begin{tabular}{|c|c|}
\hline "Font name" & \begin{tabular}{l}
Variable (STRING). Includes the font of the text. \\
Example: PLC_PRG.stFontVar := 'Arial'; \\
The selection of fonts corresponds to the default "Font" dialog.
\end{tabular} \\
\hline "Size" & \begin{tabular}{l}
Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name. \\
- <pt>: Points (default) \\
Example: PLC_PRG.iFontHeight <pt> \\
Code: iFontHeight : INT := 12; \\
- <px> : Pixels \\
Example: PLC_PRG.iFontHeight <px> \\
Code: iFontHeight : INT := 19; \\
pt \\
pt \\
px \\
If you click in the value field, a drop-down list opens on the right for setting the unit. \\
Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property "Text property \(\rightarrow\) Font".
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Flags" & \begin{tabular}{l}
Variable (DWORD). Contains the flags for displaying fonts. \\
Flags: \\
- 1: Italics \\
- 2: Bold \\
- 4: Underline \\
- 8: Strikethrough \\
Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: PLC_PRG.dwFontType := 6;
\end{tabular} \\
\hline "Character set" & \begin{tabular}{l}
Variable (DWORD). Contains a character set number for the font. \\
The selection of character set numbers corresponds to the "Script" setting of the standard "Font" dialog.
\end{tabular} \\
\hline "Color" & \begin{tabular}{l}
Variable (DWORD). Includes the color of the text. \\
Example: PLC_PRG.dwColorFont:= 16\#FF000000;
\end{tabular} \\
\hline "Flags for text alignment" & \begin{tabular}{l}
Variable (integer data type). Contains the coding for text alignment. \\
Example: PLC_PRG.dwTextAlignment. \\
Coding: \\
- 0 : Top left \\
- 1: Horizontal center \\
- 2: Right \\
- 4: Vertical center \\
- 8: Bottom \\
Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: PLC_PRG.dwFontType := 5;
\end{tabular} \\
\hline
\end{tabular}

Fixed values for displaying texts are set in "Text properties".

See also
- «ヶ "Element property 'Text properties'" on page 3016

Element prop- The Element property is used as an interface for project variables to dynamically control colors erty 'Color varia- at runtime. bles'
\begin{tabular}{|c|c|}
\hline "Toggle color" & \begin{tabular}{l}
The property controls the toggled color at runtime. \\
Value assignment: \\
- FALSE: The element is displayed with the color specified in the "Color" property. \\
- true: The element is displayed with the color specified in the "Alarm color" property. \\
Assignment options: \\
- Placeholder for the user input variable \\
- "<toggle/tap variable>" \\
- "<NOT toggle/tap variable>" \\
The color change is not controlled by its own variable, but by a user input variable. \\
Note: Specify a variable for the mouse events "Tap" or "Toggle" in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both "Toggle" and "Tap", then the variable specified in "Tap" is used. \\
Hint: Click the symbol to insert the placeholder "<toggle/tap variable>". When you activate the "Inputconfiguration", "Tap FALSE" property, then the "<NOT toggle/tap variable>" placeholder is displayed. \\
- Instance path of a project variable (BOOL) \\
Example: PLC_PRG.xColorIsToggeled \\
Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.
\end{tabular} \\
\hline \begin{tabular}{l}
"Normal state" \\
"Alarm state"
\end{tabular} & The properties listed below control the color depending on the state. The normal state is in effect if the variable in "Color variables", "Toggle color" is not defined or it has the value FALSE. The alarm state is in effect if the variable in "Colorvariables", "Toggle color" has the value tRUE. \\
\hline "Frame color" & \begin{tabular}{l}
Assignment options: \\
- Variable (DWORD) for the frame color \\
Example: PLC_PRG.dwBorderColor \\
- Color literal Example of green and opaque: 16\#FF00FF00
\end{tabular} \\
\hline "Filling color" & \begin{tabular}{l}
Assignment options: \\
- Variable (DWORD) for the fill color \\
Example: PLC_PRG.dwFillColor \\
- Color literal \\
Example of gray and opaque: 16\#FF888888
\end{tabular} \\
\hline
\end{tabular}

The transparency part of the color value is evaluated only if the "Activate semitransparent drawing" option of the visualization manager is selected.

Select the "Advanced" option in the toolbar of the properties view. Then all element properties are visible.

\section*{See also}
- \& Chapter 6.4.5.10.4 "Animating a color display" on page 2914

Element prop- The properties contain IEC variables for controlling the appearance of the element dynamically. erty 'Appearance variables'
\begin{tabular}{|l|l|}
\hline "Line width" & Variable (integer data type). Contains the line weight (in pixels). \\
\hline "Fill attributes" & Variable (DWORD). Controls whether the fill color of the element is visible. \\
& - Variable value = 0: Filled \\
& - Variable value > 0: Invisible; no fill color \\
\hline "Line style" & Variable (DWORD). Controls the line style. \\
& Coding: \\
& - \(0:\) Solid line \\
& - 1: Dashed line \\
& - 2: Dotted line \\
& - 3: Line type "Dash Dot" \\
& - 3: Line type "Dash Dot Dot" \\
& - 8: Invisible; no line \\
\hline
\end{tabular}

Fixed values can be set in the "Appearance" property. These values can be overwritten by dynamic variables at runtime.

\section*{See also}
- \#y "Element property 'Appearance"' on page 3043

Element prop- The variables control the element behavior dynamically. erty 'State varia-

\section*{bles'}
\begin{tabular}{|l|l|}
\hline "Invisible"" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \(=\) FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", "X", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain the configurations for the user input when using the mouse or keyboard. erty 'Input con- A user input defines an event and one or more actions that are executed when an event occurs. figuration'
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
The "Configure" button opens the "Input Configuration" dialog. There you can create or edit user inputs. \\
Configured user inputs are listed below the events. They each include the action that is triggered and the setting \\
in short form. \\
Example: "Execute ST Code":
\end{tabular} \\
\hline "Onc__PRG.i_x \(:=0 ;\) \\
\hline "OnDialogClosed" & Input event: The user closes the dialog. \\
\hline "OnMouseDown" & \begin{tabular}{l} 
Input event: The user clicks the mouse button completely in the element area. \\
The mouse button is clicked and released.
\end{tabular} \\
\hline "OnMouseEnter" & Input event: The user clicks down on the mouse button. \\
\hline "OnMouseLeave" & Input event: The user drags the mouse pointer to the element. \\
\hline
\end{tabular}
\(\left.\begin{array}{|l|l|}\hline \text { "OnMouseMove" } & \text { Input event: The user moves the mouse pointer over the element area. } \\ \hline \text { "OnMouseUp" } & \begin{array}{l}\text { Input events: } \\ \text { - } \\ \text { The user releases the mouse button within the element area. It is irrelevant } \\ \text { whether the user has previously pressed the mouse button inside or outside } \\ \text { the element area. }\end{array} \\ \text { - The user presses the mouse button within the element area, leaves the } \\ \text { element area, and then releases the mouse button. } \\ \text { Note: This CODESYS-specific triggering behavior guarantees that actions for } \\ \text { key elements are completed. A key element starts an action for "OnMouseDown" } \\ \text { and ends the action for "OnMouseUp". } \\ \text { Example: A visualization user presses the mouse button within the element area } \\ \text { of the key element and then moves the cursor position so that it lies outside the } \\ \text { element area. The action is ended anyway because "OnMouseUp" is triggered. }\end{array}\right\}\)
\begin{tabular}{|l|l|}
\hline "Tap" & \begin{tabular}{l} 
When a mouse click event occurs, the variable defined in "Variable" is described \\
in the application. The coding depends on the "Tap FALSE" and "Tap on enter if \\
captured" options.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL) that is set on mouse click event. \\
Example: PLC_PRG . bI sTapped \\
TRUE: A mouse click event exists. It lasts as long as the user presses the mouse \\
button over the element. It ends when the button is released.
\end{tabular} \\
& \begin{tabular}{l} 
FALSE: A mouse click event does not exist. \\
Requirement: The "Tap FALSE" option is not activated.
\end{tabular} \\
\hline "Tap FALSE" & \begin{tabular}{l} 
V: The mouse click event leads to a complementary value in "Variable". \\
TRUE: A mouse click event does not exist.
\end{tabular} \\
\hline "Tap on enter if captured" & \begin{tabular}{l} 
FALSE: While the mouse click event exists.
\end{tabular} \\
\hline \begin{tabular}{l} 
pointer is dragged within the element area or not while the mouse button is \\
pressed. \\
TRUE: While the mouse click event exists and the mouse pointer is moved over \\
the element area. \\
FALSE: A mouse click event does not exist. Or the user moves the mouse \\
pointer outside of the element area while the mouse button is pressed. \\
The value is TRUE again as soon as the user moves the pointer back to the \\
element area. The mouse is then captured.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Toggle" & \begin{tabular}{l} 
With the onset of a mouse click event, the variable is set; when the mouse click \\
event is completed, the variable is reset.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL). Its value toggled when the mouse click event is ended. This is \\
when the user releases the mouse button while the mouse pointer is over the \\
element area. \\
If the user releases the mouse button while the mouse pointer is outside of the \\
element area, then the mouse click event is not ended and the value is not \\
toggled. \\
Hint: The user can cancel a started toggle input by dragging the mouse pointer \\
out of the element area.
\end{tabular} \\
\hline "Toggle on up if captured" & \begin{tabular}{l} 
V: The value toggles regardless of where the mouse pointer is when the mouse \\
button is released. The mouse is then captured.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Hotkey" & \begin{tabular}{l}
Keyboard shortcut on the element for triggering specific input actions. \\
When the keyboard shortcut event occurs, the input actions in the "Events" property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.
\end{tabular} \\
\hline "Key" & \begin{tabular}{l}
Key pressed for input action. \\
Example: [T] \\
Note: The following properties appear when a key is selected.
\end{tabular} \\
\hline "Events" & \begin{tabular}{l}
- "None" \\
- "Mouse down": Pressing the key triggers the input actions that are configured in the "OnMouseDown" property. \\
- "Mouse up": Releasing the key triggers the input actions that are configured in the "OnMouseUp" property. \\
- "Mouse down/up": Pressing and releasing the key triggers the input actions that are configured in the "OnMouseDown" property and the "OnMouseUp" property.
\end{tabular} \\
\hline "Shift" & Combination with the Shift key Example: [Shift]+[T]. \\
\hline "Control" & Combination with the Ctrl key Example: [Ctrl]+[T]. \\
\hline "Alt" & Combination with the Alt key Example: \([A I t]+[T]\). \\
\hline
\end{tabular}

All keyboard shortcuts and their actions that are configured in the visualization are listed on the "Keyboard Configuration" tab.

\section*{See also}
- ⓨ Chapter 6.4.5.21.2.2 "Command 'Keyboard Configuration'" on page 3341
- \(\xi^{2}\) Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration'" on page 3370

Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages:
\end{tabular} \\
& \begin{tabular}{ll}
\(\bullet\) "Not set. Full rights.": Access rights for all user groups : "operable" \\
\(\bullet \quad\) "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\xi^{2}\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

See also
- \(\xi^{4}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Pie'}

Symbol:

Category: "Basic"
The element draws a pie of any angle.

\section*{Element proper- \\ ties}
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Example: Error_rate_part_1 \\
Optional \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list.
\end{tabular} \\
\hline & "Pie" \\
\hline
\end{tabular}

Element prop- The position defines the location and size of the element in the visualization window. This is erty 'Position' based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X\) " & \begin{tabular}{l} 
The \(x\)-coordinate of the upper left corner of the element \\
Specified in pixels \\
Example: 10
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
The \(y\)-coordinate of the upper left corner of the element \\
Specified in pixels \\
Example: 10
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels \\
Example: 30
\end{tabular} \\
\hline & \begin{tabular}{l} 
Tip: You can change the values in " " ", "Y", "Width", and "Height" by dragging the \\
corresponding symbols థథ- to another position in the editor.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Angle" & \begin{tabular}{l} 
Static angle of rotation (in degrees). \\
Example: 35 \\
The element is displayed rotated in the editor. The point of rotation is the center \\
of the element. A positive value rotates clockwise. \\
Tip: You can change the value in the editor by focusing the element to the \\
handle. When the cursor is displayed as a rotating arrow \(\Omega\), you can rotate the \\
element about its center as a handle.
\end{tabular} \\
\hline
\end{tabular}

See also
- © Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874
\begin{tabular}{|l|l|l|}
\hline "Begin" & \begin{tabular}{l} 
Start angle of the pie. If you also define a variable \\
for the start, then the start angle is calculated from \\
the sum of the values for "Begin" and "Variable for \\
begin".
\end{tabular} & \begin{tabular}{l} 
Example: \\
- "Begin": 330 \\
- \\
"End": 90
\end{tabular} \\
\hline & \begin{tabular}{l} 
"End" \\
for the ende of the pie. If you also define a variable the end angle is calculated from \\
the sum of the values for "End" and "Variable for \\
end". \\
The pie is drawn clockwise from the start angle to \\
the end angle.
\end{tabular} \\
\hline "Variable for begin" & The start of the sector is defined dynamically by a variable. \\
\hline "Variable for end" & The end of the sector is defined dynamically by a variable. \\
\hline \begin{tabular}{l} 
"Only show circle \\
line"
\end{tabular} & \begin{tabular}{ll} 
V: The pie is drawn without the radius line or filling color. \\
\hline
\end{tabular} \\
\hline
\end{tabular}

\section*{Element prop- \\ erty 'Center'}
\begin{tabular}{|l|l|}
\hline " \(X\) " & Display of the center coordinates. You cannot modify these values here in the \\
properties. \\
If the Pie is selected in the editor, then the center of the Pie (as well as the \\
center of the enveloping box) is visualized with the symbol \(\boldsymbol{\oplus}\). Moreover, the \\
element is decorated with a position, begin, and end boxes that you can move. \\
\hline
\end{tabular}

Element prop-
erty 'Colors'
\(\left.\begin{array}{|l|l|}\hline \text { "Normal state" } & \begin{array}{l}\text { The normal state is in effect if the variable in "Color variables } \rightarrow \text { Toggle color" is } \\
\text { not defined or it has the value FALSE. }\end{array} \\
\hline \text { "Frame color" } & \text { Frame and fill color for the corresponding state of the variable. } \\
\hline \text { "Fill color" } & \text { "Transparency" } \\
\hline \text { Transparency value (0 to 255) for defining the transparency of the selected color. } \\
\text { Example: } 255: \text { The color is opaque. } 0: \text { The color is completely transparent. }\end{array}\right] .\)\begin{tabular}{ll} 
The alarm state is in effect if the variable in "Color variables \(\rightarrow\) Toggle color" has \\
the value TRUE.
\end{tabular}.

See also
- \({ }^{\circ}\), Chapter 6.4.5.21.3.5 "Dialog 'Gradient Editor'" on page 3369

Element prop- The properties contain fixed values for setting the look of the element. erty 'Appearance'
\begin{tabular}{|c|c|}
\hline \multirow[t]{3}{*}{"Line width"} & Value in pixels \\
\hline & Example: 2 \\
\hline & Note: The values 0 and 1 both result in a line weight of 1 pixel. If no line should be displayed, then the "Line style" property must be set to the option "Invisible". \\
\hline "Fill attributes" & \begin{tabular}{l}
The way in which the element is filled. \\
- "Filled":The element is filled with the color from property "Colors \(\boldsymbol{\rightarrow}\) Fill color". \\
- "Invisible": The fill color is invisible.
\end{tabular} \\
\hline "Line style" & \begin{tabular}{l}
Type of line representation \\
- "Solid" \\
- "Dashes" \\
- "Dots" \\
- "Dash Dot" \\
- "Dash Dot Dot" \\
- "not visible"
\end{tabular} \\
\hline
\end{tabular}

You can assign variables in the "Appearance variables" property for controlling the appearance dynamically. The fixed values here are overwritten.

See also
- «" Element property 'Appearance variables'" on page 3051

Element prop- The properties contains character strings for labeling the element. The character string can also erty 'Texts' contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.
CODESYS accepts the specified texts automatically into the "GlobalTextList" text list. Therefore, these texts can be localized.
\begin{tabular}{|l|l|}
\hline "Text" & \begin{tabular}{l} 
Character string (without single straight quotation marks) for the labeling the \\
element. Add a line break by pressing the keyboard shortcut [CtrI] + [Enter]. \\
Example: Accesses: \%i \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Text".
\end{tabular} \\
\hline "Tooltip" & \begin{tabular}{l} 
Character string (without single straight quotation marks) that is displayed as the \\
tooltip of an element. \\
Example: Number of valid accesses. \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Tooltip".
\end{tabular} \\
\hline
\end{tabular}

See also
- \#y "Element property 'Text variables"' on page 3031
- \(\Leftrightarrow\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872
- Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329

Element prop- The properties contain fixed values for the text properties.
\begin{tabular}{|c|c|}
\hline "Horizontal alignment" & Horizontal alignment of the text within the element. \\
\hline "Vertical alignment" & Vertical alignment of the text within the element. \\
\hline "Text format" & \begin{tabular}{l}
Definition for displaying texts that are too long \\
- "Default": The long text is truncated. \\
- "Line break": The text is split into parts. \\
- "Ellipsis": The visible text ends with "..." indicating that it is not complete.
\end{tabular} \\
\hline "Font" & \begin{tabular}{l}
Example: "Default"
The "Font" dialog box opens. \\
マ: Drop-down list with style fonts.
\end{tabular} \\
\hline "Font color" & \begin{tabular}{l}
Example: "Black"
The "Color" dialog box opens. \\
*: Drop-down list with style colors.
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l}
Whole number (value range from 0 to 255). This determines the transparency of the respective color. \\
Example: 255: The color is opaque. \\
0 : The color is completely transparent. \\
Please note: If the color is a style color and already has a transparency value, then this property is write-protected.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" & \multicolumn{1}{|c|}{\begin{tabular}{l} 
"X" \\
\hline Variable (integer data type). Defines the X position (in pixels). \\
Example: PLC_PRG. iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular}} \\
\hline V" & \begin{tabular}{l} 
Variable (integer data type). Defines the Y position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline "Scaling" & \begin{tabular}{l} 
Variable (integer data type). Causes centric stretching. \\
Example: PLC_PRG. iScaling. \\
The reference point is the "Center" property. \\
The value 1 shrinks the element by a factor of 0.001. The value 1000 returns the element \\
to its original size.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Interior rotation" & \begin{tabular}{l} 
Variable (integer data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol. \\
If a static angle of rotation is specified in "Position \\
G Angle", then the static angle of rotation and the \\
angle of rotation are added.
\end{tabular} \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

The " \(X\) ", " \(Y\) ", and "Interior rotation" properties are supported by the "Client Animation" functionality.

Element prop- These properties are variables with contents that replace a format definition. erty 'Text varia-
bles'
\begin{tabular}{|l|l|}
\hline "Text variable" & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses \\
Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Text".
\end{tabular} \\
& \begin{tabular}{l} 
Note: If you specify a variable of type enumeration with text list support, then \\
the name of the enumeration data type is added automatically in angle brackets \\
after the variable name. Example: PLC_PRG.enVar <enumeration name>. \\
Then the symbolic value of the enumeration component is printed instead of the \\
numeric value when text is printed. Refer to the help page for the enumerations.
\end{tabular} \\
\hline & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses InTooltip
\end{tabular} \\
Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Tooltip". \\
\hline
\end{tabular}

See also
- \({ }^{*}\) Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329
- " "Element property 'Texts"' on page 3029
- © Chapter 6.4.1.20.5.18 "Enumerations" on page 2263

Element prop- Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is erty 'Dynamic texts' currently indexed in the variable.
\begin{tabular}{|l|l|}
\hline "Text list" & \begin{tabular}{l} 
Variable (string) or name of the text list as a fixed string in single straight quota- \\
tion marks. \\
Example: ' Errorlist'
\end{tabular} \\
& : Drop-down list with the dialogs available in the text lists.
\end{tabular}

See also
- \({ }^{\text {• }}\) Chapter 6.4.1.21.2.28 "Object 'Text List"' on page 2532

Element prop- The variables allow for dynamic control of the text display. erty 'Font variables'
\begin{tabular}{|c|c|}
\hline "Font name" & \begin{tabular}{l}
Variable (STRING). Includes the font of the text. \\
Example: PLC_PRG.stFontVar := 'Arial'; \\
The selection of fonts corresponds to the default "Font" dialog.
\end{tabular} \\
\hline "Size" & \begin{tabular}{l}
Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name. \\
- <pt>: Points (default) \\
Example: PLC_PRG.iFontHeight <pt> \\
Code: iFontHeight : INT := 12; \\
- <px> : Pixels \\
Example: PLC_PRG.iFontHeight <px> \\
Code: iFontHeight : INT := 19; \\
If you click in the value field, a drop-down list opens on the right for setting the unit. \\
Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property "Text property \(\rightarrow\) Font".
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Flags" & \begin{tabular}{l}
Variable (DWORD). Contains the flags for displaying fonts. \\
Flags: \\
- 1: Italics \\
- 2: Bold \\
- 4: Underline \\
- 8: Strikethrough \\
Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: PLC_PRG.dwFontType :=6;
\end{tabular} \\
\hline "Character set" & \begin{tabular}{l}
Variable (DWORD). Contains a character set number for the font. \\
The selection of character set numbers corresponds to the "Script" setting of the standard "Font" dialog.
\end{tabular} \\
\hline "Color" & \begin{tabular}{l}
Variable (DWORD). Includes the color of the text. \\
Example: PLC_PRG.dwColorFont:= 16\#FF000000;
\end{tabular} \\
\hline "Flags for text alignment" & \begin{tabular}{l}
Variable (integer data type). Contains the coding for text alignment. \\
Example: PLC_PRG.dwTextAlignment. \\
Coding: \\
- 0 : Top left \\
- 1: Horizontal center \\
- 2: Right \\
- 4: Vertical center \\
- 8: Bottom \\
Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: PLC_PRG.dwFontType := 5;
\end{tabular} \\
\hline
\end{tabular}

Fixed values for displaying texts are set in "Text properties".

See also
- «ヶ "Element property 'Text properties'" on page 3029

Element prop- The Element property is used as an interface for project variables to dynamically control colors erty 'Color varia- at runtime. bles'
\begin{tabular}{|c|c|}
\hline "Toggle color" & \begin{tabular}{l}
The property controls the toggled color at runtime. \\
Value assignment: \\
- FALSE: The element is displayed with the color specified in the "Color" property. \\
- TRUE: The element is displayed with the color specified in the "Alarm color" property. \\
Assignment options: \\
- Placeholder for the user input variable \\
- "<toggle/tap variable>" \\
- "<NOT toggle/tap variable>" \\
The color change is not controlled by its own variable, but by a user input variable. \\
Note: Specify a variable for the mouse events "Tap" or "Toggle" in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both "Toggle" and "Tap", then the variable specified in "Tap" is used. \\
Hint: Click the symbol to insert the placeholder "<toggle/tap variable>". When you activate the "Inputconfiguration", "Tap FALSE" property, then the "<NOT toggle/tap variable>" placeholder is displayed. \\
- Instance path of a project variable (BOOL) \\
Example: PLC_PRG.xColorIsToggeled \\
Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.
\end{tabular} \\
\hline \begin{tabular}{l}
"Normal state" \\
"Alarm state"
\end{tabular} & The properties listed below control the color depending on the state. The normal state is in effect if the variable in "Color variables", "Toggle color" is not defined or it has the value FALSE. The alarm state is in effect if the variable in "Colorvariables", "Toggle color" has the value TRUE. \\
\hline "Frame color" & \begin{tabular}{l}
Assignment options: \\
- Variable (DWORD) for the frame color Example: PLC_PRG.dwBorderColor \\
- Color literal Example of green and opaque: 16\#FF00FF00
\end{tabular} \\
\hline "Filling color" & \begin{tabular}{l}
Assignment options: \\
- Variable (DWORD) for the fill color Example: PLC_PRG.dwFillColor \\
- Color literal Example of gray and opaque: \(16 \#\) FF888888
\end{tabular} \\
\hline
\end{tabular}

The transparency part of the color value is evaluated only if the "Activate semitransparent drawing" option of the visualization manager is selected.

Select the "Advanced" option in the toolbar of the properties view. Then all element properties are visible.

\section*{See also}
- \& Chapter 6.4.5.10.4 "Animating a color display" on page 2914

Element prop- The properties contain IEC variables for controlling the appearance of the element dynamically. erty 'Appearance variables'
\begin{tabular}{|l|l|}
\hline "Line width" & Variable (integer data type). Contains the line weight (in pixels). \\
\hline "Fill attributes" & Variable (DWORD). Controls whether the fill color of the element is visible. \\
& - Variable value = 0: Filled \\
& - Variable value > 0: Invisible; no fill color \\
\hline "Line style" & Variable (DWORD). Controls the line style. \\
& Coding: \\
& - \(0:\) Solid line \\
& - 1: Dashed line \\
& - 2: Dotted line \\
& - 3: Line type "Dash Dot" \\
& - 3: Line type "Dash Dot Dot" \\
& - 8: Invisible; no line \\
\hline
\end{tabular}

Fixed values can be set in the "Appearance" property. These values can be overwritten by dynamic variables at runtime.

See also
- ©s "Element property 'Appearance"' on page 3043

Element prop- The variables control the element behavior dynamically. erty 'State varia-
bles'
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
\\
\\
\\
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain the configurations for the user input when using the mouse or keyboard. erty 'Input con- A user input defines an event and one or more actions that are executed when an event occurs. figuration'

The "Configure" button opens the "Input Configuration" dialog. There you can create or edit user inputs.
Configured user inputs are listed below the events. They each include the action that is triggered and the setting in short form.

Example: "Execute ST Code": ßPLC_PRG.i_x := 0;
\begin{tabular}{|l|l|}
\hline "OnDialogClosed" & Input event: The user closes the dialog. \\
\hline "OnMouseClick" & \begin{tabular}{l} 
Input event: The user clicks the mouse button completely in the element area. \\
The mouse button is clicked and released.
\end{tabular} \\
\hline "OnMouseDown" & Input event: The user clicks down on the mouse button. \\
\hline "OnMouseEnter" & Input event: The user drags the mouse pointer to the element. \\
\hline "OnMouseLeave" & Input event: The user drags the mouse pointer away from the element. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "OnMouseMove" & Input event: The user moves the mouse pointer over the element area. \\
\hline "OnMouseUp" & \begin{tabular}{l} 
Input events: \\
- \begin{tabular}{l} 
The user releases the mouse button within the element area. It is irrelevant \\
whether the user has previously pressed the mouse button inside or outside \\
the element area.
\end{tabular} \\
- The user presses the mouse button within the element area, leaves the \\
element area, and then releases the mouse button. \\
Note: This CODESYS-specific triggering behavior guarantees that actions for \\
key elements are completed. A key element starts an action for "OnMouseDown"" \\
and ends the action for "OnMouseUp". \\
Example: A visualization user presses the mouse button within the element area \\
of the key element and then moves the cursor position so that it lies outside the \\
element area. The action is ended anyway because "OnMouseUp" is triggered.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Tap" & \begin{tabular}{l} 
When a mouse click event occurs, the variable defined in "Variable" is described \\
in the application. The coding depends on the "Tap FALSE" and "Tap on enter if \\
captured" options.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL) that is set on mouse click event. \\
Example: PLC_PRG . bI sTapped \\
TRUE: A mouse click event exists. It lasts as long as the user presses the mouse \\
button over the element. It ends when the button is released.
\end{tabular} \\
& \begin{tabular}{l} 
FALSE: A mouse click event does not exist. \\
Requirement: The "Tap FALSE" option is not activated.
\end{tabular} \\
\hline "Tap FALSE" & \begin{tabular}{l} 
V: The mouse click event leads to a complementary value in "Variable". \\
TRUE: A mouse click event does not exist.
\end{tabular} \\
\hline FALSE: While the mouse click event exists.
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Toggle" & \begin{tabular}{l} 
With the onset of a mouse click event, the variable is set; when the mouse click \\
event is completed, the variable is reset.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL). Its value toggled when the mouse click event is ended. This is \\
when the user releases the mouse button while the mouse pointer is over the \\
element area. \\
If the user releases the mouse button while the mouse pointer is outside of the \\
element area, then the mouse click event is not ended and the value is not \\
toggled. \\
Hint: The user can cancel a started toggle input by dragging the mouse pointer \\
out of the element area.
\end{tabular} \\
\hline "Toggle on up if captured" & \begin{tabular}{l} 
V: The value toggles regardless of where the mouse pointer is when the mouse \\
button is released. The mouse is then captured.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Hotkey" & \begin{tabular}{l} 
Keyboard shortcut on the element for triggering specific input actions. \\
When the keyboard shortcut event occurs, the input actions in the "Events" \\
property are triggered. In this way, it is not the input action itself that leads to this \\
input action, but the mouse input action.
\end{tabular} \\
\hline "Key" & \begin{tabular}{l} 
Key pressed for input action. \\
Example: [T] \\
Note: The following properties appear when a key is selected.
\end{tabular} \\
\hline "Events" & \begin{tabular}{l} 
"None" \\
- "Mouse down": Pressing the key triggers the input actions that are configured \\
in the "OnMouseDown" property. \\
"Mouse up": Releasing the key triggers the input actions that are configured \\
in the "OnMouseUp" property.
\end{tabular} \\
\hline "Mouse down/up": Pressing and releasing the key triggers the input actions \\
that are configured in the "OnMouseDown" property and the "OnMouseUp" \\
property.
\end{tabular}

All keyboard shortcuts and their actions that are configured in the visualization are listed on the "Keyboard Configuration" tab.

\section*{See also}
- ⓨ Chapter 6.4.5.21.2.2 "Command 'Keyboard Configuration'" on page 3341
- \(\xi^{2}\) Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration'" on page 3370

Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'
\begin{tabular}{|c|c|}
\hline "Access rights" & \begin{tabular}{l}
Opens the "Access rights" dialog. There you can edit the access privileges for the element. \\
Status messages: \\
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
- "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline \multicolumn{2}{|l|}{See also} \\
\hline & 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366 \\
\hline
\end{tabular}

\section*{Visualization Element 'Image'}

Symbol:

The element adds an image to the visualization. The displayed image is managed in the image pool and referenced in the visualization element by means of a static ID. You can also change the displayed image dynamically by using a variable instead of the static ID.

With the "Background" command, you can define a background for the entire visualization.

Directories that contain the images for use in visualizations can be defined in the project settings (category "Visualization").

\section*{Element proper-}
ties
\begin{tabular}{|c|c|}
\hline "Element name" & \begin{tabular}{l}
Example: Status bar \\
Optional \\
Hint: Assign individual names for elements so that they are found faster in the element list.
\end{tabular} \\
\hline "Type of element" & "Image" \\
\hline "Static ID" & \begin{tabular}{l}
Identifier of the image file for a static assignment \\
ID of the image file on, as it is defined in the corresponding image pool. If the image is not included in the global image pool in the POU view, then the instance path must be specified. Then the name of the image pool is preceded to make the entry unique. Example: imagepool2.button_image. \\
When a new ID is specified, a file selection dialog opens. The selected file is saved to the "GloballmagePool". \\
See also: Help for the "Image Pool" object.
\end{tabular} \\
\hline "Show frame" & \(\square\) : The image file is displayed with a frame. \\
\hline "Clipping" & Requirement: The "Scaling type" property is "Fixed".
Only part of the visualization is displayed that fits in the element frame. \\
\hline "Transparent" & The image pixels that have the "Transparent color" are displayed as transparent. \\
\hline "Transparent color" & \begin{tabular}{l}
Effective only if the "Transparent" option is activated. \\
The... button opens the color selection dialog. This is where you select the transparent color.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Scaling type" & \begin{tabular}{l}
Definition of how an image fits in the element frame. \\
- "Isotropic": The entire image is displayed in the element frame, either larger or smaller. As a result, the proportion of height and width are retained. If the alignment of the elements to each other should also be retained within a scaled frame element, then note the following. Unwanted horizontal or vertical offsets can be prevented by setting the properties "Horizontal alignment" and "Vertical alignment" to "Centered". The alignment of the elements is retained and there are no resulting horizontal or vertical offsets. Example: A lamp is centered above a switch. The lamp should remain in the horizontally centered position, even if the frame is resized. \\
- "Anisotropic": The image resizes automatically to the dimensions of the element frame, filling the entire element frame. As a result, the proportions are not retained. \\
- "Fixed": The image retains its original size, even if the element frame is resized. Note also that the "Clipping" option is selected. \\
For each reassignment of an image ID, the element size is adapted automatically to the image size.
\end{tabular} \\
\hline "Horizontal alignment" & \begin{tabular}{l}
Horizontal alignment of the element within the element frame: \\
- "Left" \\
- "Centered" \\
- "Right" \\
Requirement: The scaling type of the image is "Isotropic" or "Fixed". \\
Note: If the visualization is referenced, then the horizontal alignment takes effect within the frame position. \\
औ: The "Variable" property is shown below this.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Enumeration variable (ENUM \\
VisuElemBase.VisuEnumVerticalAlignment). Contains the horizontal alignment. \\
Example: PLC_PRG.eHorizontalAlignment
\end{tabular} \\
\hline "Vertical alignment" & \begin{tabular}{l}
Vertical alignment of the element within the element frame: \\
- "Top" \\
- "Centered" \\
- "Bottom" \\
Requirement: The scaling type of the image is "Isotropic" or "Fixed". \\
Note: If the visualization is referenced, then the horizontal alignment takes effect within the frame position. \\
औ: The "Variable" property is shown below this.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Enumeration variable (ENUM \\
VisuElemBase.VisuEnumVerticalAlignment). Contains the vertical alignment. \\
Example: PLC_PRG.eVerticalAlignment
\end{tabular} \\
\hline
\end{tabular}


See also
- Object 'Image Pool'

Element property 'Position'

The position defines the location and size of the element in the visualization window. This is based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|c|c|}
\hline " \(X\) " & \begin{tabular}{l}
The x-coordinate of the upper left corner of the element Specified in pixels \\
Example: 10
\end{tabular} \\
\hline "Y" & \begin{tabular}{l}
The y-coordinate of the upper left corner of the element Specified in pixels \\
Example: 10
\end{tabular} \\
\hline "Width" & \begin{tabular}{l}
Specified in pixels \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l}
Specified in pixels \\
Example: 30
\end{tabular} \\
\hline & Tip: You can change the values in " \(X\) ", " \(Y\) ", "Width", and "Height" by dragging the corresponding symbols \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Angle" & \begin{tabular}{l} 
Static angle of rotation (in degrees). \\
Example: 35 \\
The element is displayed rotated in the editor. The point of rotation is the center \\
of the element. A positive value rotates clockwise. \\
Tip: You can change the value in the editor by focusing the element to the \\
handle. When the cursor is displayed as a rotating arrow \\
element about its center as a handle.
\end{tabular} \\
& \begin{tabular}{l} 
(1): Handle can rotate the \\
Note: If a dynamic angle of rotation is also configured in the property \\
"Absolute movement \(\rightarrow\) Internal rotation", then the static and dynamic angles \\
of rotation are added in runtime mode. The static angle of rotation acts as an \\
offset.
\end{tabular} \\
\hline
\end{tabular}

See also
- ② Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center'
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline "Y" & \(Y\)-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus}\) ) to other positions in the editor.

Element prop- The properties contain fixed values for setting colors. erty 'Colors'
\begin{tabular}{|l|l|}
\hline "Color" & \begin{tabular}{l} 
Color for the frame \\
Requirement: "Show frame" property is activated. \\
Rlease note that the normal state is in effect if the expression in the \\
"Color variables \(\rightarrow\) Toggle color" property is not defined or it has the value \\
FALSE.
\end{tabular} \\
\hline "Alarm color" & \begin{tabular}{l} 
Color for the frame in alarm state \\
Requirement: "Show frame" property is activated. \\
Please note that the alarm state is in effect if the expression in the \\
"Color variables \(\rightarrow\) Toggle color" property has the value TRUE.
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l} 
Value (0 to 255) for defining the transparency of the selected color. \\
Example 255: The color is opaque. 0: The color is completely transparent.
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\xi^{\&}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

Element prop- The properties contain fixed values for setting the look of the element. erty 'Appearance'
\begin{tabular}{|l|l|}
\hline "Line width" & Value in pixels \\
& Example: 2 \\
& \begin{tabular}{l} 
Note: The values 0 and 1 both result in a line weight of one pixel. If no line \\
should be displayed, then the "Line style" property must be set to the option \\
"Invisible".
\end{tabular} \\
\hline "Line style" & \begin{tabular}{l} 
Type of line representation \\
\\
\\
\\
\\
\\
\\
\\
\\
\\
• "Solid" "Dashes" \\
• "Dots" \\
• "Dash Dot" \\
• "not visible"
\end{tabular} \\
\hline
\end{tabular}

You can assign variables in the "Appearance variables" property for controlling the appearance dynamically. The fixed values are defined here.

See also
- *y "Element property 'Appearance variables'" on page 3051

Element prop- The properties contains character strings for labeling the element. The character string can also erty 'Texts' contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.
CODESYS accepts the specified texts automatically into the "GlobalTextList" text list. Therefore, these texts can be localized.
\begin{tabular}{|l|l|}
\hline "Text" & \begin{tabular}{l} 
Character string (without single straight quotation marks) for the labeling the \\
element. Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter]. \\
Example: Accesses: \%i \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Text".
\end{tabular} \\
\hline "Tooltip" & \begin{tabular}{l} 
Character string (without single straight quotation marks) that is displayed as the \\
tooltip of an element. \\
Example: Number of valid accesses. \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Tooltip".
\end{tabular} \\
\hline
\end{tabular}

See also
- *4 "Element property 'Text variables"' on page 3046
- « Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872
- \(\Longleftrightarrow\) Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329

Element prop- The properties contain fixed values for the text properties.
erty 'Text prop-
erties'
\begin{tabular}{|l|l|}
\hline "Horizontal alignment" & Horizontal alignment of the text within the element. \\
\hline "Vertical alignment" & Vertical alignment of the text within the element. \\
\hline "Text format" & \begin{tabular}{l} 
Definition for displaying texts that are too long \\
- "Default": The long text is truncated. \\
- "Line break": The text is split into parts. \\
- "Ellipsis": The visible text ends with "..." indicating that it is not complete.
\end{tabular} \\
\hline "Font" & \begin{tabular}{l} 
Example: "Default" \\
n...: The "Font" dialog box opens. \\
- : Drop-down list with style fonts.
\end{tabular} \\
\hline "Font color" & \begin{tabular}{l} 
Example: "Black" \\
"..: The "Color" dialog box opens. \\
v: Drop-down list with style colors.
\end{tabular} \\
\hline & \begin{tabular}{l} 
Whole number (value range from 0 to 255 ). This determines the transparency of \\
the respective color. \\
Example: \(255:\) The color is opaque.
\end{tabular} \\
0: The color is completely transparent. \\
Please note: If the color is a style color and already has a transparency value, \\
then this property is write-protected.
\end{tabular}

\section*{Element prop- \\ erty 'Image ID \\ variable'}
\begin{tabular}{|l|l|}
\hline "Image ID" & \begin{tabular}{l} 
Variable (STRING). Contains the image ID. The contents of the string corre- \\
sponds to the description of the "Static ID" property. \\
Example: PLC_PRG.stImageID \(:=\) 'ImagePool_A.Image3'; \\
\hline
\end{tabular} \\
\hline
\end{tabular}

See also
- \#y Chapter 6.4.5.21.5.5 "Visualization Element 'Image'" on page 3463
- \({ }^{\bullet}\) Chapter 6.4.1.21.2.15 "Object 'Image Pool"' on page 2468

Element prop- You can use this element property for animating a series of image files. erty 'Dynamic image'
\begin{tabular}{|l|l|}
\hline "Bitmap version" & \begin{tabular}{l} 
Variable (integer data type). Contains the version of the image. \\
If the variable changes, then the visualization re-reads the image referenced in \\
the "Image ID" property and displays it. \\
The visualization displays animations when the image file on the controller is \\
updated continuously, thus incrementing the version variable. The application \\
must be programmed for this. \\
Possible applications \\
- Displaying graphics that are generated by the application \\
- Displaying images that are refreshed by a camera
\end{tabular} \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|c|c|c|}
\hline \multicolumn{3}{|l|}{"Movement"} \\
\hline " \(X\) " & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular}} \\
\hline "Y" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular}} \\
\hline "Rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle1. \\
The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol. \\
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
\end{tabular} &  \\
\hline "Scaling" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (integer data type). Causes centric stretching. \\
Example: PLC_PRG.iScaling. \\
The reference point is the "Center" property. \\
The value 1 shrinks the element by a factor of 0.001 . The value 1000 returns the element to its original size.
\end{tabular}} \\
\hline "Interior rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle2. \\
In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. \\
The rotation point is shown as the \(\boldsymbol{\oplus}\) symbol. \\
Note: If a static angle of rotation is specified in the property "Position \(\rightarrow\) Angle", then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.
\end{tabular} &  \\
\hline
\end{tabular}


You can link the variables to a unit conversion.

See also

Element property 'Relative movement'

The properties contains variables for moving the element. The reference point is the position of the element ("Position" property). The shape of the element can change.
\begin{tabular}{|c|c|}
\hline "Movement top-left" & \\
\hline "X" & \begin{tabular}{l}
Variable (integer data type). It contains the number (in pixels) that the left edge is moved horizontally. Incrementing the value moves the element to the right. \\
Example: PLC_PRG.iDeltaX
\end{tabular} \\
\hline " \(Y\) " & \begin{tabular}{l}
Variable (integer data type). It contains the number (in pixels) that the top edge is moved vertically. Incrementing the value moves the element to the down. \\
Example: PLC_PRG.iDeltaY
\end{tabular} \\
\hline "Movement bottom-right" & \\
\hline "X" & \begin{tabular}{l}
Variable (integer data type). It contains the number (in pixels) that the right edge is moved horizontally. Incrementing the value moves the element to the right. \\
Example: PLC_PRG.iDeltaWidth
\end{tabular} \\
\hline " \(Y\) " & \begin{tabular}{l}
Variable (integer data type). It contains the number (in pixels) that the bottom edge is moved vertically. Incrementing the value moves the element to the down. \\
Example: PLC_PRG.iDeltaHeight
\end{tabular} \\
\hline
\end{tabular}

See also
- «2 "Element property 'Absolute movement"' on page 3016

Element prop- These properties are variables with contents that replace a format definition. erty 'Text variables'
\begin{tabular}{|l|l|}
\hline "Text variable" & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses
\end{tabular} \\
& \begin{tabular}{l} 
Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Text". \\
Note: If you specify a variable of type enumeration with text list support, then \\
the name of the enumeration data type is added automatically in angle brackets \\
after the variable name. Example: PLC_PRG.enVar <enumeration name>. \\
Then the symbolic value of the enumeration component is printed instead of the \\
numeric value when text is printed. Refer to the help page for the enumerations.
\end{tabular} \\
\hline "Tooltip variable" & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses InTooltip \\
Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Tooltip".
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\Leftrightarrow\) Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329
- ⓨ "Element property 'Texts'" on page 3043
- \(\quad \Rightarrow\) Chapter 6.4.1.20.5.18 "Enumerations" on page 2263

Element property 'Dynamic texts'

Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is currently indexed in the variable.
\begin{tabular}{|c|c|}
\hline "Text list" & \begin{tabular}{l}
Variable (string) or name of the text list as a fixed string in single straight quotation marks. \\
Example: 'Errorlist' \\
*: Drop-down list with the dialogs available in the text lists.
\end{tabular} \\
\hline "Text index" & \begin{tabular}{l}
Text list ID. This refers to the desired output text. \\
- As fixed string with the ID in single straight quotation marks. \\
Example: '1' \\
- As a variable (STRING) for dynamically controlling the text output. \\
Example: strTextID \\
Sample assignment: PLC_PRG.strTextID := '1';
\end{tabular} \\
\hline "Tooltip index" & \begin{tabular}{l}
Text list ID. This refers to the desired output text. \\
- As fixed string with the ID in single straight quotation marks. \\
Example: '2' \\
- As a variable (STRING) for dynamically controlling the text output. \\
Example: strToolTipID \\
Sample assignment: PLC_PRG.strToolTipID := '2';
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\quad \stackrel{y}{c}\) Chapter 6.4.1.21.2.28 "Object 'Text List"" on page 2532

\footnotetext{
Element prop- The variables allow for dynamic control of the text display. erty 'Font variables'
}
\begin{tabular}{|c|c|}
\hline "Font name" & \begin{tabular}{l}
Variable (STRING). Includes the font of the text. \\
Example: PLC_PRG.stFontVar := 'Arial'; \\
The selection of fonts corresponds to the default "Font" dialog.
\end{tabular} \\
\hline "Size" & \begin{tabular}{l}
Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name. \\
- <pt>: Points (default) \\
Example: PLC_PRG.iFontHeight <pt> \\
Code: iFontHeight : INT := 12; \\
- <px> : Pixels \\
Example: PLC_PRG.iFontHeight <px> \\
Code: iFontHeight : INT := 19; \\
If you click in the value field, a drop-down list opens on the right for setting the unit. \\
Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property "Text property \(\rightarrow\) Font".
\end{tabular} \\
\hline "Flags" & \begin{tabular}{l}
Variable (DWORD). Contains the flags for displaying fonts. \\
Flags: \\
- 1: Italics \\
- 2: Bold \\
- 4: Underline \\
- 8: Strikethrough \\
Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: PLC_PRG.dwFontType :=6;
\end{tabular} \\
\hline "Character set" & \begin{tabular}{l}
Variable (DWORD). Contains a character set number for the font. \\
The selection of character set numbers corresponds to the "Script" setting of the standard "Font" dialog.
\end{tabular} \\
\hline "Color" & \begin{tabular}{l}
Variable (DWORD). Includes the color of the text. \\
Example: PLC_PRG.dwColorFont:= 16\#FF000000;
\end{tabular} \\
\hline "Flags for text alignment" & \begin{tabular}{l}
Variable (integer data type). Contains the coding for text alignment. \\
Example: PLC_PRG.dwTextAlignment. \\
Coding: \\
- 0 : Top left \\
- 1: Horizontal center \\
- 2: Right \\
- 4: Vertical center \\
- 8: Bottom \\
Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: PLC_PRG.dwFontType := 5;
\end{tabular} \\
\hline
\end{tabular}

\section*{Fixed values for displaying texts are set in "Text properties".}

See also
- « "Element property 'Text properties'" on page 3044

Element prop- The Element property is used as an interface for project variables to dynamically control colors erty 'Color varia- at runtime. bles'
\begin{tabular}{|c|c|}
\hline "Toggle color" & \begin{tabular}{l}
The property controls the toggled color at runtime. \\
Value assignment: \\
- FALSE: The element is displayed with the color specified in the "Color" property. \\
- TRUE: The element is displayed with the color specified in the "Alarm color" property. \\
Assigning the property: \\
- Placeholder for the user input variable \\
- "<toggle/tap variable>" \\
- "<NOT toggle/tap variable>" \\
The color change is not controlled by its own variable, but by a user input variable. \\
Note: Specify a variable for the mouse events "Tap" or "Toggle" in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both "Toggle" and "Tap", then the variable specified in "Tap" is used. \\
Hint: Click the symbol to insert the placeholder "<toggle/tap variable>". When you activate the "Inputconfiguration", "Tap FALSE" property, then the "<NOT toggle/tap variable>" placeholder is displayed. \\
- Instance path of a project variable (BOOL) \\
Example: PLC_PRG.xColorIsToggeled \\
Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.
\end{tabular} \\
\hline "Color" & \begin{tabular}{l}
Color variable for the frame \\
- Variable (DWORD) for the color \\
Example: PLC_PRG.dwColor \\
- Color literal \\
Example of gray and opaque: \(16 \#\) FF888888 \\
Requirement: "Show frame" property is activated. \\
Please note that the normal state is in effect if the expression in the "Colorvariables \(\rightarrow\) Toggle color" property is not defined or it has the value FALSE.
\end{tabular} \\
\hline "Alarm color" & \begin{tabular}{l}
Color variable for the frame in alarm state \\
- Variable (DWORD) for the alarm color \\
Example: PLC_PRG.dwAlarmColor \\
- Color literal \\
Example of red and opaque: 16\#FFFF0000 \\
Please note that the alarm state is in effect if the expression in the "Colorvariables \(\rightarrow\) Toggle color" property has the value TRUE.
\end{tabular} \\
\hline
\end{tabular}

The transparency part of the color value is evaluated only if the "Activate semitransparent drawing" option of the visualization manager is selected.

See also
- « Chapter 6.4.5.10.4 "Animating a color display" on page 2914
- \& Chapter 6.4.5.21.4.2 "Object 'Visualization manager'" on page 3398

Element property 'Appearance variables'
\begin{tabular}{|l|l|}
\hline "Line width" & \begin{tabular}{l} 
Variable (integer data type). Contains the line weight (in pixels). \\
Note: The values 0 and 1 both result in a line weight of one pixel. If no line \\
should be displayed, then the "Line style" property must be set to the option \\
"Invisible".
\end{tabular} \\
\hline "Line style" & Variable (DWORD). Controls the line style. \\
& Coding: \\
& - \(0:\) Solid line \\
& - 1: Dashed line \\
& - \(2:\) Dotted line \\
& - 3: Line type "Dash Dot" \\
& - 3: Line type "Dash Dot Dot" \\
& - 8: Invisible: The line is not drawn. \\
\hline
\end{tabular}

Fixed values can be set in the "Appearance" property. These values can be overwritten by dynamic variables at runtime.

See also
- « "Element property 'Appearance"' on page 3043

Element prop- The variables control the element behavior dynamically. erty 'State varia-
bles'
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \(:=\) FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain the configurations for the user input when using the mouse or keyboard. erty 'Input con- User input is a user event from the perspective of the element. figuration'

The "Configure" button opens the "Input configuration" dialog box for creating or modifying a user input configuration.
A configuration contains one or more input actions for the respective input event. Existing input actions are displayed below it.
Example: "Execute ST code": ßPLC_PRG.i_x := 0;
\begin{tabular}{|l|l|}
\hline "OnDialogClosed" & Input event: The user closes the dialog box. \\
\hline "OnMouseClick" & \begin{tabular}{l} 
Input event: A user clicks the element completely. The mouse button is clicked \\
and released.
\end{tabular} \\
\hline "OnMouseDown" & Input event: A user clicks down on the element only. \\
\hline "OnMouseEnter" & Input event: A user drags the mouse pointer to the element. \\
\hline "OnMouseLeave" & Input event: A user drags the mouse pointer away from the element. \\
\hline "OnMouseMove" & Input event: A user moves the mouse pointer over the element area. \\
\hline "OnMouseUp" & Input event: The user releases the mouse button over the element area. \\
\hline
\end{tabular}

See also
- \(\quad\) Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration'" on page 3370
\begin{tabular}{|c|c|}
\hline "Hotkeys" & \begin{tabular}{l}
Keyboard shortcut on the element for triggering specific input actions. \\
When the keyboard shortcut event occurs, the input actions in the "Event(s)" property are triggered.
\end{tabular} \\
\hline "Key" & \begin{tabular}{l}
Key pressed for input action. \\
Example: [T]
\end{tabular} \\
\hline "Event(s)" & \begin{tabular}{l}
- "None" \\
- "Mouse down": Pressing the key triggers the input actions that are configured in the "OnMouseDown" property. \\
- "Mouse up": Releasing the key triggers the input actions that are configured in the "OnMouseUp" property. \\
- "Mouse down/up": Pressing and releasing the key triggers the input actions that are configured in the "OnMouseDown" property and the "OnMouseUp" property.
\end{tabular} \\
\hline "Shift" & Combination with the Shift key Example: [Shift]+[T]. \\
\hline "Control" & Combination with the Ctrl key Example: [Ctrl] \(+[T]\). \\
\hline "Alt" & : Combination with the Alt key Example: \([A t t]+[T]\). \\
\hline
\end{tabular}

All keyboard shortcuts and their actions that are configured in the visualization are listed in the "Keyboard configuration" tab.

See also
- « \({ }^{2}\) Chapter 6.4.5.21.2.2 "Command 'Keyboard Configuration"" on page 3341

Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'
\begin{tabular}{|c|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages:
\end{tabular} \\
& \begin{tabular}{l} 
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
- "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline See also
\end{tabular}

See also
- \({ }^{\#}\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights"' on page 3366

\section*{See also}
- Project Settings - Visualization
- \& Chapter 6.4.5.21.2.10 "Command 'Background"' on page 3349
- Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Frame'}

Symbol:

Category: "Basic"
The element serves as a frame in which to display one or more already existing visualizations. You get a structured user interface. The size of the frame can be fixed or scaled. The display area of the referenced visualization then adapts itself to the frame size.

\section*{Element proper- \\ ties}
\begin{tabular}{|l|l|}
\hline "Element name" & Example: refVisUserInfo \\
\hline "Type of element" & "Frame" \\
\hline "Clipping" & \begin{tabular}{l} 
冋: Fixed size. Only that part of the referenced visualization that fits inside the \\
frame is displayed. \\
Requirement: The "Scaling type" property is "Fixed".
\end{tabular} \\
\hline "Show frame" & \begin{tabular}{l} 
Displays the frame \\
- "No frame": The displayed area of the frame does not have borders. \\
- "Frame": The displayed area of the frame has borders. \\
- "No frame with offset": The displayed area of the frame does not have a \\
border and the displayed area of the referenced visualization is reduced \\
inwards by one pixel as compared to the frame area. The gap prevents the \\
referenced visualization from touching any adjacent elements.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Scaling type" & \begin{tabular}{l} 
The method with which the height and width of the referenced visualization are \\
scaled. \\
- "Isotropic": The visualization is scaled to the size of the element. The visuali- \\
- zation retains its proportions with a fixed height/width ratio. \\
- Anisotropic": The visualization is scaled to the size of the element. The \\
height and width are adapted to the element independently of each other. \\
- "Fixed": the visualization is displayed in its original size without taking into \\
account the size of the element. \\
- "Fixed and scrollable": The visualization is displayed fixed in the element. If it \\
is larger than the element, the element will be provided with scrollbars. \\
Please note: assign varialos to the properties "Scoll position variable \\
horizontal"" "Scroll position variable vertical". You can then edit the data \\
of the scrollbar position in the application.
\end{tabular} \\
\hline
\end{tabular}

Element proper- The properties contain variables for the position of the scrollboxes in the scrollbars. You can ties 'Scrollbar then edit the data of the scrollbox position in the application. settings'
\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|l|}{Requirement: the property "Scaling type" is "fixed and scrollable".} \\
\hline "Scroll position variable horizontal" & \multirow[t]{2}{*}{Variable (integer data type, also as array). Contains the position of the horizontal or vertical scrollbox. The array contains the position for every display variant. If the visualization runs on several display variants, then the position changes are decoupled from each other.} \\
\hline "Scroll position variable & \\
\hline & Example: \\
\hline & PLC_PRG.iScrollHor [CURRENTCLIENTID] \\
\hline & PLC_PRG.iScrollVer [CURRENTCLIENTID] \\
\hline & The variable is declared as an array in the example. \\
\hline & iScrollHor: ARRAY[0..20] OF INT; \\
\hline & iScrollVer: ARRAY[0..20] OF INT; \\
\hline & CURRENTCLIENTID indexes the current display variant. \\
\hline
\end{tabular}


See also
- Unit conversion
\begin{tabular}{|l|l|}
\hline "Deactivation of the \\
background character" & \begin{tabular}{l} 
: The background is drawn. The non-animated element of the referenced visu- \\
alization is drawn as a background bitmap in order to optimize the performance \\
of the visualization. \\
Consequence: Elements can be displayed in an unexpected order at runtime. \\
For example, an animated element can push itself behind the Frame at runtime. \\
目: Background character is deactivated in order to avoid the behavior described \\
above.
\end{tabular} \\
\hline
\end{tabular}

Element prop- Contains the currently configured visualization references as a subnode
erty 'Referen-
ces'
\begin{tabular}{|l|l|}
\hline "References" & \begin{tabular}{l} 
Clicking "Configure" opens the "Frame Configuration" dialog. This is used to \\
manage the referenced visualizations. \\
Caution: Visualizations can be nested at any depth by means of Frame ele-- \\
ments. In order to use the "Switch to any visualization" Frame selection type \\
without any problems, a Frame must not contain more than 21 referenced \\
visualizations. For more information, see also the description for the "Input \\
configuration" of an element: Action "Switch Frame visualization".
\end{tabular} \\
\hline \begin{tabular}{l} 
List of the currently referenced \\
visualizations
\end{tabular} & \begin{tabular}{l} 
Visualizations that have a button also have this displayed as a subnode. Each \\
interface variable is listed with the currently assigned transfer parameters. \\
Example:
\end{tabular} \\
& \begin{tabular}{l} 
vis_FormA \\
- iDataToDisplay_1 : PLC_PRG. iVar1 \\
\\
\\
- iDataToDisplay_2 : PLC_PRG . iVar2 \\
Hint: You can change the assignment of the variables to an interface variable \\
here and edit the value field. Or click the "Configure" button instead.
\end{tabular} \\
\hline
\end{tabular}

See also
- « Chapter 6.4.5.21.2.1 "Command 'Interface Editor"' on page 3340
- EChapter 6.4.5.17 "Creating a structured user interface" on page 2940
- « "Input action 'Switch Frame Visualization"' on page 3377

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline "X" & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols ( ( - ) to other positions in the editor.

See also
- * Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X "\) & X-coordinate of the point of rotation \\
\hline " \(Y\) " & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus})\) to other positions in the editor.

Element prop- The properties contain fixed values for the colors. erty 'Colors'
\begin{tabular}{|l|l|}
\hline "Color"" & \begin{tabular}{l} 
Color of the frame \\
v: Selection list with style colors appears \\
\(\ldots . .: S t a n d a r d ~ d i a l o g ~ " C o l o r " ~ o p e n s ~ f o r ~ s e l e c t i n g ~ a ~ c o l o r . ~\)
\end{tabular} \\
& \begin{tabular}{l} 
Please note: the normal state is when the boolean variable in the property \\
"Color variables \(\rightarrow\) Toggle color" is not defined or its value is FALSE.
\end{tabular} \\
\hline "Alarm color" & \begin{tabular}{l} 
Color with which the element is filled during the alarm state. \\
Please note: Alarm state is when the value of the boolean variable in the prop- \\
erty "Color variables \(\rightarrow\) Toggle color" is FALSE.
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l} 
Integer number (value range from 255 to 0). Specifies the transparency of the \\
associated color. \\
\(255: ~ T h e ~ c o l o r ~ i s ~ o p a q u e . ~\)
\end{tabular} \\
\(0:\) The color is fully transparent. \\
Please note: If the color is a style color and already contains a transparency \\
value, then this property is write-protected.
\end{tabular}

See also
- ② Chapter 6.4.5.5.4 "Assigning a color" on page 2876

Element prop- The properties contain fixed values for setting the look of the element. erty 'Appearance'
\begin{tabular}{|c|c|}
\hline "Line width" & \begin{tabular}{l}
Value in pixels \\
Example: 2 \\
Note: The values 0 and 1 both result in a line weight of one pixel. If no line should be displayed, then the "Line style" property must be set to the option "Invisible".
\end{tabular} \\
\hline "Line style" & \begin{tabular}{l}
Type of line representation \\
- "Solid" \\
- "Dashes" \\
- "Dots" \\
- "Dash Dot" \\
- "Dash Dot Dot" \\
- "not visible"
\end{tabular} \\
\hline
\end{tabular}

You can assign variables in the "Appearance variables" property for controlling the appearance dynamically. The fixed values are defined here.

See also
- * "Element property 'Appearance variables"' on page 3064

Element prop- The properties contains character strings for labeling the element. The character string can also erty 'Texts' contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.

CODESYS accepts the specified texts automatically into the "GlobalTextList" text list. Therefore, these texts can be localized.
\begin{tabular}{|l|l|}
\hline "Text" & \begin{tabular}{l} 
Character string (without single straight quotation marks) for the labeling the \\
element. Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter]. \\
Example: Accesses: \%i \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Text".
\end{tabular} \\
\hline "Tooltip" & \begin{tabular}{l} 
Character string (without single straight quotation marks) that is displayed as the \\
tooltip of an element. \\
Example: Number of valid accesses. \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Tooltip".
\end{tabular} \\
\hline
\end{tabular}

See also
- «y "Element property 'Text variables'" on page 3060
- \(\xi^{\star}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872
- \({ }^{4}\) Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329

Element prop- The properties contain fixed values for the text properties.
erty 'Text prop-
erties'
\begin{tabular}{|c|c|}
\hline "Horizontal alignment" & Horizontal alignment of the text within the element. \\
\hline "Vertical alignment" & Vertical alignment of the text within the element. \\
\hline "Text format" & \begin{tabular}{l}
Definition for displaying texts that are too long \\
- "Default": The long text is truncated. \\
- "Line break": The text is split into parts. \\
- "Ellipsis": The visible text ends with "..." indicating that it is not complete.
\end{tabular} \\
\hline "Font" & Example: "Default"
: The "Font" dialog box opens.
Drop-down list with style fonts. \\
\hline "Font color" & Example: "Black"
: The "Color" dialog box opens.
: Drop-down list with style colors. \\
\hline "Transparency" & \begin{tabular}{l}
Whole number (value range from 0 to 255). This determines the transparency of the respective color. \\
Example: 255: The color is opaque. \\
0 : The color is completely transparent. \\
Please note: If the color is a style color and already has a transparency value, then this property is write-protected.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline \multicolumn{1}{|l|}{ "Movement" } \\
\hline " \(X "\) & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline " \(Y\) " & \begin{tabular}{l} 
Variable (numeric data type). Defines the \(Y\) position (in pixels). \\
Example: PLC_PRG. iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle_1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\oplus\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
"Scaling" & \begin{tabular}{l} 
Variable (integer data type). Causes centric stretching. \\
Example: PLC_PRG. iScaling. \\
The reference point is the "Center" property. \\
The value 1 shrinks the element by a factor of 0.001 . The value 1000 returns the element \\
to its original size.
\end{tabular} \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

Element property 'Relative movement'

The properties "X", "Y", "Rotation", and "Interior rotation" are supported by the "Client Animation" functionality.

\section*{See also}
\begin{tabular}{|l|l|}
\hline "Movement top-left" & \\
\hline " \(X\) " & \begin{tabular}{l} 
Variable (integer data type). It contains the number (in pixels) that the left edge \\
is moved horizontally. Incrementing the value moves the element to the right. \\
Example: pLC_PRG.iDeltax
\end{tabular} \\
\hline " \(Y\) " & \begin{tabular}{l} 
Variable (integer data type). It contains the number (in pixels) that the top edge \\
is moved vertically. Incrementing the value moves the element to the down. \\
Example: PLC_PRG.iDeltaY
\end{tabular} \\
\hline "Movement bottom-right" & \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline\(" X "\) & \begin{tabular}{l} 
Variable (integer data type). It contains the number (in pixels) that the right edge \\
is moved horizontally. Incrementing the value moves the element to the right. \\
Example: PLC_PRG.iDeltaWidth
\end{tabular} \\
\hline\(Y "\) & \begin{tabular}{l} 
Variable (integer data type). It contains the number (in pixels) that the bottom \\
edge is moved vertically. Incrementing the value moves the element to the down. \\
Example: PLC_PRG.iDeltaHeight
\end{tabular} \\
\hline
\end{tabular}

See also
- «乡 "Element property 'Absolute movement"' on page 3016

Element prop- These properties are variables with contents that replace a format definition. erty 'Text variables'
\begin{tabular}{|l|l|}
\hline "Text variable" & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses
\end{tabular} \\
& \begin{tabular}{l} 
Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Text". \\
Note: If you specify a variable of type enumeration with text list support, then \\
the name of the enumeration data type is added automatically in angle brackets \\
after the variable name. Example: PLC_PRG.enVar <enumeration name>. \\
Then the symbolic value of the enumeration component is printed instead of the \\
numeric value when text is printed. Refer to the help page for the enumerations.
\end{tabular} \\
\hline "Tooltip variable" & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses InTooltip \\
Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Tooltip".
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{\text {• }}\) Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329
- « "Element property 'Texts'" on page 3057
- \(\sum^{*}\) Chapter 6.4.1.20.5.18 "Enumerations" on page 2263

Element prop- Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is erty 'Dynamic texts'
\begin{tabular}{|c|c|}
\hline "Text list" & \begin{tabular}{l}
Variable (string) or name of the text list as a fixed string in single straight quotation marks. \\
Example: 'Errorlist' \\
v: Drop-down list with the dialogs available in the text lists.
\end{tabular} \\
\hline "Text index" & \begin{tabular}{l}
Text list ID. This refers to the desired output text. \\
- As fixed string with the ID in single straight quotation marks. Example: ' 1 ' \\
- As a variable (STRING) for dynamically controlling the text output. \\
Example: strTextID \\
Sample assignment: PLC_PRG.strTextID := '1';
\end{tabular} \\
\hline "Tooltip index" & \begin{tabular}{l}
Text list ID. This refers to the desired output text. \\
- As fixed string with the ID in single straight quotation marks. Example: '2' \\
- As a variable (STRING) for dynamically controlling the text output. \\
Example: strToolTipID \\
Sample assignment: PLC_PRG.strToolTipID := '2';
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{\circ}\) Chapter 6.4.1.21.2.28 "Object 'Text List"' on page 2532

Element prop- The variables allow for dynamic control of the text display. erty 'Font variables'
\begin{tabular}{|c|c|}
\hline "Font name" & \begin{tabular}{l}
Variable (STRING). Includes the font of the text. \\
Example: PLC_PRG.stFontVar := 'Arial'; \\
The selection of fonts corresponds to the default "Font" dialog.
\end{tabular} \\
\hline "Size" & \begin{tabular}{l}
Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name. \\
- <pt>: Points (default) \\
Example: PLC_PRG.iFontHeight <pt> \\
Code: iFontHeight : INT := 12; \\
- <px>: Pixels \\
Example: PLC_PRG.iFontHeight <px> \\
Code: iFontHeight : INT := 19; \\
If you click in the value field, a drop-down list opens on the right for setting the unit. \\
Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property "Text property \(\rightarrow\) Font".
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Flags" & \begin{tabular}{l}
Variable (DWORD). Contains the flags for displaying fonts. \\
Flags: \\
- 1: Italics \\
- 2: Bold \\
- 4: Underline \\
- 8: Strikethrough \\
Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: PLC_PRG.dwFontType := 6;
\end{tabular} \\
\hline "Character set" & \begin{tabular}{l}
Variable (DWORD). Contains a character set number for the font. \\
The selection of character set numbers corresponds to the "Script" setting of the standard "Font" dialog.
\end{tabular} \\
\hline "Color" & \begin{tabular}{l}
Variable (DWORD). Includes the color of the text. \\
Example: PLC_PRG.dwColorFont:= 16\#FF000000;
\end{tabular} \\
\hline "Flags for text alignment" & \begin{tabular}{l}
Variable (integer data type). Contains the coding for text alignment. \\
Example: PLC_PRG.dwTextAlignment. \\
Coding: \\
- 0 : Top left \\
- 1: Horizontal center \\
- 2: Right \\
- 4: Vertical center \\
- 8: Bottom \\
Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: PLC_PRG.dwFontType := 5;
\end{tabular} \\
\hline
\end{tabular}

Fixed values for displaying texts are set in "Text properties".

Element prop- The Element property is used as an interface for project variables to dynamically control colors erty 'Color varia- at runtime. bles'
\begin{tabular}{|c|c|}
\hline "Toggle color" & \begin{tabular}{l}
The property controls the toggled color at runtime. \\
Value assignment: \\
- FALSE: The element is displayed with the color specified in the "Color" property. \\
- TRUE: The element is displayed with the color specified in the "Alarm color" property. \\
Assigning the property: \\
- Placeholder for the user input variable \\
- "<toggle/tap variable>" \\
- "<NOT toggle/tap variable>" \\
The color change is not controlled by its own variable, but by a user input variable. \\
Note: Specify a variable for the mouse events "Tap" or "Toggle" in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both "Toggle" and "Tap", then the variable specified in "Tap" is used. \\
Hint: Click the symbol to insert the placeholder "<toggle/tap variable>". When you activate the "Inputconfiguration", "Tap FALSE" property, then the "<NOT toggle/tap variable>" placeholder is displayed. \\
- Instance path of a project variable (BOOL) \\
Example: PLC_PRG.xColorIsToggeled \\
Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.
\end{tabular} \\
\hline "Color" & \begin{tabular}{l}
Color variable for the Frame \\
- Variable (DWORD) for the color \\
Example: PLC_PRG.dwColor \\
- Color literal \\
Example of gray and opaque: 16\#FF888888 \\
Requirement: "Show Frame" property is activated. \\
Please note that the normal state is in effect if the expression in the "Colorvariables \(\rightarrow\) Toggle color" property is not defined or it has the value FALSE.
\end{tabular} \\
\hline "Alarm color" & \begin{tabular}{l}
Color variable for the Frame in alarm state \\
- Variable (DWORD) for the alarm color \\
Example: PLC_PRG.dwAlarmColor \\
- Color literal \\
Example of red and opaque: 16\#FFFF0000 \\
Please note that the alarm state is in effect if the expression in the "Colorvariables \(\rightarrow\) Toggle color" property has the value TRUE.
\end{tabular} \\
\hline
\end{tabular}

The transparency part of the color value is evaluated only if the "Activate semitransparent drawing" option of the visualization manager is selected.

Select the "Advanced" option in the toolbar of the properties view. Then all element properties are visible.

See also
- \# Chapter 6.4.5.10.4 "Animating a color display" on page 2914
- © Chapter 6.4.5.21.4.2 "Object 'Visualization manager'" on page 3398

Element prop- The properties contain variables for controlling the appearance of the element dynamically. erty 'Appearance variables'
\begin{tabular}{|l|l|}
\hline "Line width" & \begin{tabular}{l} 
Variable (integer data type). Contains the line weight (in pixels). \\
Note: The values 0 and 1 both result in a line weight of one pixel. If no line \\
should be displayed, then the "Line style" property must be set to the option \\
"Invisible".
\end{tabular} \\
\hline "Line style" & Variable (DWORD). Controls the line style. \\
& Coding: \\
& - \(0:\) Solid line \\
& - 1: Dashed line \\
& - 2: Dotted line \\
& - 3: Line type "Dash Dot" \\
& - 3: Line type "Dash Dot Dot" \\
& - 8: Invisible: The line is not drawn. \\
\hline
\end{tabular}

Fixed values can be set in the "Appearance" property. These values can be overwritten by dynamic variables at runtime.

See also
- © "Element property 'Appearance"' on page 3057

Element property 'Switch frame variable'

The variable controls the switching of the referenced visualizations. This variable indexes one of the referenced frame visualizations and this is displayed in the frame. When the value of the variable changes, it switches to the recently indexed visualization.
\begin{tabular}{|c|c|}
\hline "Variable" & \begin{tabular}{l}
- Variable (integer data type) that contains the index of the active visualization Example: PLC_PRG.uiIndexVisu \\
Hint: The "Frame Configuration" dialog includes a list of referenced visualizations. The visualizations are automatically numerically indexed via the order in the list. \\
Note: This variant of switching usually affects all connected display variants. \\
- Array element (integer data type) for index access via CURRENTCLIENTID Example: PLC_PRG.aIndexVisu [CURRENTCLIENTID] \\
Note: This variant of switching applies to the current client only, and therefore only on one display variant. That is the display variant where the value change was triggered (for example, by means of user input).
\end{tabular} \\
\hline
\end{tabular}

See also
- * Chapter 6.4.5.21.2.9 "Command 'Frame Selection"' on page 3348

\footnotetext{
Element prop- The variables control the element behavior dynamically.
erty 'State varia-
bles'
}
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \(:=\) FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
\\
\\
\begin{tabular}{l} 
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", "X", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain the configurations for the user input when using the mouse or keyboard. erty 'Input con- A user input defines an event and one or more actions that are executed when an event occurs. figuration'

The "Configure" button opens the "Input Configuration" dialog. There you can create or edit user inputs.
Configured user inputs are listed below the events. They each include the action that is triggered and the setting in short form.
Example: "Execute ST Code": 3 PLC_PRG.i_x := 0;
\begin{tabular}{|l|l|}
\hline "OnDialogClosed" & Input event: The user closes the dialog. \\
\hline "OnMouseClick" & \begin{tabular}{l} 
Input event: The user clicks the mouse button completely in the element area. \\
The mouse button is clicked and released.
\end{tabular} \\
\hline "OnMouseDown" & Input event: The user clicks down on the mouse button. \\
\hline "OnMouseEnter" & Input event: The user drags the mouse pointer to the element. \\
\hline "OnMouseLeave" & Input event: The user drags the mouse pointer away from the element. \\
\hline "OnMouseMove" & Input event: The user moves the mouse pointer over the element area. \\
\hline "OnMouseUp" & \begin{tabular}{l} 
Input events: \\
- \begin{tabular}{l} 
The user releases the mouse button within the element area. It is irrelevant \\
whether the user has previously pressed the mouse button inside or outside \\
the element area.
\end{tabular} \\
\hline
\end{tabular}
\end{tabular}
- The user presses the mouse button within the element area, leaves the element area, and then releases the mouse button.
Note: This CODESYS-specific triggering behavior guarantees that actions for key elements are completed. A key element starts an action for "OnMouseDown" and ends the action for "OnMouseUp".
Example: A visualization user presses the mouse button within the element area of the key element and then moves the cursor position so that it lies outside the element area. The action is ended anyway because "OnMouseUp" is triggered.
\begin{tabular}{|l|l|}
\hline "Tap" & \begin{tabular}{l} 
When a mouse click event occurs, the variable defined in "Variable" is described \\
in the application. The coding depends on the "Tap FALSE" and "Tap on enter if \\
captured" options.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL) that is set on mouse click event. \\
Example: PLC_PRG. bIsTapped \\
TRUE: A mouse click event exists. It lasts as long as the user presses the mouse \\
button over the element. It ends when the button is released. \\
FALSE: A mouse click event does not exist. \\
Requirement: The "Tap FALSE" option is not activated.
\end{tabular} \\
\hline "Tap FALSE" & \begin{tabular}{l} 
V: The mouse click event leads to a complementary value in "Variable". \\
TRUE: A mouse click event does not exist. \\
FALSE: While the mouse click event exists.
\end{tabular} \\
\hline "Tap on enter if captured"" & \begin{tabular}{l} 
V: During user input, it is also taken into consideration whether the mouse \\
pointer is dragged within the element area or not while the mouse button is \\
pressed. \\
TRUE: While the mouse click event exists and the mouse pointer is moved over \\
the element area. \\
FALSE: A mouse click event does not exist. Or the user moves the mouse \\
pointer outside of the element area while the mouse button is pressed. \\
The value is TRUE again as soon as the user moves the pointer back to the \\
element area. The mouse is then captured.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Toggle" & \begin{tabular}{l} 
With the onset of a mouse click event, the variable is set; when the mouse click \\
event is completed, the variable is reset.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL). Its value toggled when the mouse click event is ended. This is \\
when the user releases the mouse button while the mouse pointer is over the \\
element area. \\
If the user releases the mouse button while the mouse pointer is outside of the \\
element area, then the mouse click event is not ended and the value is not \\
toggled. \\
Hint: The user can cancel a started toggle input by dragging the mouse pointer \\
out of the element area.
\end{tabular} \\
\hline "Toggle on up if captured" & \begin{tabular}{l} 
V: The value toggles regardless of where the mouse pointer is when the mouse \\
button is released. The mouse is then captured.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Hotkey" & \begin{tabular}{l}
Keyboard shortcut on the element for triggering specific input actions. \\
When the keyboard shortcut event occurs, the input actions in the "Events" property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.
\end{tabular} \\
\hline "Key" & \begin{tabular}{l}
Key pressed for input action. \\
Example: [T] \\
Note: The following properties appear when a key is selected.
\end{tabular} \\
\hline "Events" & \begin{tabular}{l}
- "None" \\
- "Mouse down": Pressing the key triggers the input actions that are configured in the "OnMouseDown" property. \\
- "Mouse up": Releasing the key triggers the input actions that are configured in the "OnMouseUp" property. \\
- "Mouse down/up": Pressing and releasing the key triggers the input actions that are configured in the "OnMouseDown" property and the "OnMouseUp" property.
\end{tabular} \\
\hline "Shift" & Combination with the Shift key Example: [Shift]+[T]. \\
\hline "Control" & Combination with the Ctrl key Example: [Ctrl]+[T]. \\
\hline "Alt" & Combination with the Alt key Example: [AIt]+[T]. \\
\hline
\end{tabular}


All keyboard shortcuts and their actions that are configured in the visualization are listed on the "Keyboard Configuration" tab.

See also
- « Chapter 6.4.5.21.2.2 "Command 'Keyboard Configuration"' on page 3341
- \& Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration"' on page 3370

Element property 'Access
rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages: \\
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
\\
- "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\#\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights"" on page 3366

See also
- * Chapter 6.4.5.17 "Creating a structured user interface" on page 2940
- « Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration"' on page 3370

\section*{Visualization Element 'Label'}

Symbol: \(\square\)

\section*{Category: "Common Controls"}

The element is used to label visualizations.

\section*{Element proper- \\ ties}
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Optional \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list. \\
Example: Header_Parameter
\end{tabular} \\
\hline "Type of element" & "Label" \\
\hline
\end{tabular}

Element prop- The property requires a character string.
erty 'Texts'
This text is entered automatically into the Global TextList text list and can be localized there.
\begin{tabular}{|l|l|}
\hline "Text" & \begin{tabular}{l} 
Character string (without single straight quotation marks) \\
Example: Main page
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{2}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872
- Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329

Element prop-
erty 'Position'
The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline "X" & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (■-) to other positions in the editor.

See also
- Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center' rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline " \(Y\) " & \(Y\)-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus}\) ) to other positions in the editor.

Element prop- The properties contain fixed values for the text properties. erty 'Text properties'
\begin{tabular}{|c|c|}
\hline "Horizontal alignment" & Horizontal alignment of the text within the element. \\
\hline "Vertical alignment" & Vertical alignment of the text within the element. \\
\hline "Text format" & \begin{tabular}{l}
Definition for displaying texts that are too long \\
- "Default": The long text is truncated. \\
- "Line break": The text is split into parts. \\
- "Ellipsis": The visible text ends with "..." indicating that it is not complete.
\end{tabular} \\
\hline "Font" & Example: "Default"
The "Font" dialog box opens.
: Drop-down list with style fonts. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Font color" & \begin{tabular}{l} 
Example: "Black" \\
\\
\\
\hline... : The "Color" dialog box opens. \\
マ: Drop-down list with style colors.
\end{tabular} \\
\hline & \begin{tabular}{l} 
Whole number (value range from 0 to 255 ). This determines the transparency of \\
the respective color. \\
Example: \(255:\) The color is opaque. \\
\(0:\) The color is completely transparent. \\
Please note: If the color is a style color and already has a transparency value, \\
then this property is write-protected.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG. iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG. iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\oplus\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
\hline "Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol. \\
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

The "X", " \(Y\) ", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also
-

Element prop- The variables control the element behavior dynamically. erty 'State variables'
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- Requirement: User management is set up for the visualization. erty 'Access
rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages:
\end{tabular} \\
& \begin{tabular}{l} 
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
\(\bullet\) \\
"Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

See also
- ※y Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

See also
- \(\Leftrightarrow_{2}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Combo Box, Integer'}

Symbol:


\section*{Category: "Common Controls"}

The element shows values as a list box. When the user clicks an entry, the ID of the entry is written to an integer variable. The entries in the list box can be from a list and contain images from an image pool.

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Example: List of product numbers \\
Optional \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list.
\end{tabular} \\
\hline "Type of element" & "Combo Box, Integer" \\
\hline
\end{tabular}

Element prop- The position defines the location and size of the element in the visualization window. These erty 'Position' are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X\) " & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (區) to other positions in the editor.

See also
- \({ }^{4}\) Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center' rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline " \(Y\) " & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

\begin{tabular}{|c|c|}
\hline "Variable" & \begin{tabular}{l}
At runtime, the text list ID of the list entry that the user clicks is saved at runtime. If only one image pool is displayed, then the image ID is saved. \\
Property value \\
- Variable (integer data type) \\
Example: PLC_PRG.iIDComboboxEntry \\
- Enumeration variable with text list support \\
Example: PLC PRG.eMyCombobox<COMBO>
\end{tabular} \\
\hline "Text List" & \begin{tabular}{l}
Displayed as a combo box. Every text list entry becomes a combo box entry. \\
Note: A maximum of 32766 entries can be displayed. \\
Transfer value \\
- Text list identifier as string \\
Example: 'TextList_A' \\
Note: The IDs of the text list have to be within the range of values of DWORD or DINT. \\
- Blank \\
- When an enumeration variable with text list support is specified in the "Variable" property \\
- When only one image pool is displayed
\end{tabular} \\
\hline "Image Pool" & \begin{tabular}{l}
Displayed as a combo box. Every image in the image pool becomes a combo box entry. \\
Example: 'ImagePool_A'
\end{tabular} \\
\hline
\end{tabular}

See also
- Enumerations
- \({ }^{\wedge}\) Chapter 6.4.5.8 "Setting Up Multiple Languages" on page 2906

Element prop- Displayed list that expands when a visualization user clicks into the element. erty 'Settings of the list'
\begin{tabular}{|c|c|}
\hline "Number of rows setting" & \begin{tabular}{l}
- "From style": \\
- "Explicit": Then the "Number of visible rows" property appears below it.
\end{tabular} \\
\hline "Number of visible rows" & \begin{tabular}{l}
Number of visible lines of the combo box drop-down list defined here \\
- Integer literal \\
Example: 5 \\
- Variable (integer data type) \\
Example: PLC_PRG.iNumberOfVisibleRows \\
Note: The property is available when the "Number of rows setting" property is set to "Explicit".
\end{tabular} \\
\hline "Row height" & \begin{tabular}{l}
- "From style": \\
- Literal \\
Example: 20
\end{tabular} \\
\hline "Height of image" & \begin{tabular}{l}
Image height (in pixels) of the image displayed in the drop-down list entry \\
- "From style": \\
- Integer literal \\
Example: 30 \\
Note: Images are displayed only when a value is specified in the "Image pool" property.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Width of image" & \begin{tabular}{l}
Image width (in pixels) of the image displayed in the drop-down list entry \\
- "From style": \\
- Literal Example: 30 \\
Note: Images are displayed only when a value is specified in the "Image pool" property.
\end{tabular} \\
\hline "Offset of image" & \begin{tabular}{l}
Makes the images in the selection list appear offset (in pixels) from the left margin. An offset of 0 means that the images are displayed directly on the margin. \\
- "From style": \\
- Literal \\
Example: 4 \\
Note: Images are displayed only when a value is specified in the "Image pool" property.
\end{tabular} \\
\hline "Scrollbar size" & \begin{tabular}{l}
Size of the scrollbar (in pixels). The scrollbar is displayed when more entries are specified in the drop-down list than in "Number of visible rows". \\
Default: 20
\end{tabular} \\
\hline
\end{tabular}

\section*{Element prop- \\ erty 'Texts'}
\begin{tabular}{|l|l|}
\hline "Tooltip" & \begin{tabular}{l} 
Character string (without single straight quotation marks) that is displayed as the \\
tooltip of an element in runtime mode \\
Example: Products of customer A \\
Hint: The text is accepted automatically into the "GlobalTextList" text list and can \\
be localized there.
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\xi^{4}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Element prop- \\ erty 'Value \\ range'}
\begin{tabular}{|l|l|}
\hline "Limit valuerange" & \begin{tabular}{l} 
Limits the text list to one subrange. This subrange is displayed by the combo \\
box. \\
Requirement: A value is specified in the "Text list" property. \\
目: Only the subrange that is defined by the "Minimum value" "Maximum value" \\
properties is displayed as a drop-down list.
\end{tabular} \\
\hline "Minimum value" & \begin{tabular}{l} 
ID of the text list entry from which a combo box entry is displayed \\
- Literal (ANY_NUM) \\
Example: 5
\end{tabular} \\
& \begin{tabular}{l} 
Variable (integer data type) \\
Example: PLC_PRG. iFirstEntry
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Maximum value" & \begin{tabular}{l}
ID of the text list entry up to which combo box entries are displayed \\
- Literal (ANY_NUM) \\
Example: 10 \\
- Variable (integer data type) \\
Example: PLC_PRG.iLastEntry
\end{tabular} \\
\hline "Filter missing textentries" & Text list is refreshed and any unused texts (IDs) are removed. Requirement: A value is specified in the "Text list" property. \\
\hline
\end{tabular}

Element prop- The properties contain fixed values for the text properties.
erty 'Text prop-
erties'
\begin{tabular}{|l|l|}
\hline "Usage of" & \begin{tabular}{l} 
- "Default style values": The values of the visualization style are used. \\
- "Individual settings": The "Individual text properties" property group is shown \\
The values can be customized here.
\end{tabular} \\
\hline \begin{tabular}{l} 
"Individual text properties" \\
Requirement: The "Individual settings" text property is defined.
\end{tabular} \\
\hline "Horizontal alignment" & Horizontal alignment of the text within the element. \\
\hline "Vertical alignment" & Vertical alignment of the text within the element. \\
\hline "Font" & \begin{tabular}{l} 
Example: "Default" \\
\(\ldots . .:\) :The "Font" dialog box opens. \\
v: Drop-down list with style fonts.
\end{tabular} \\
\hline "Font color" & \begin{tabular}{l} 
Example: "Black" \\
"..: The "Color" dialog box opens. \\
v: Drop-down list with style colors.
\end{tabular} \\
\hline & \begin{tabular}{l} 
Whole number (value range from 0 to 255). This determines the transparency of \\
the respective color. \\
Example: \(255:\) The color is opaque. \\
0: The color is completely transparent.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain IEC variables for controlling the position of the element dynamically. The erty 'Absolute movement'
reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" & \multicolumn{1}{|l|}{} \\
\hline "X" & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\oplus\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
"Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PPRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\boldsymbol{\oplus}\) symbol.
\end{tabular} \\
\begin{tabular}{l} 
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular}
\end{tabular}

You can link the variables to a unit conversion.

The " \(X\) ", " "", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

\section*{See also}

\section*{Element prop- The variables control the element behavior dynamically. erty 'State varia- \\ bles'}
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL := FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

\section*{The "Invisible" property is supported by the "Client Animation" functionality.}

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- Requirement: User management is set up for the visualization. erty 'Access
rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages:
\end{tabular} \\
& \begin{tabular}{l} 
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
\(\bullet \quad\) "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

\section*{See also}
- \(\boldsymbol{y y}^{\circ}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Combo Box, Array'}

Symbol:

\section*{Category: "Common Controls"}

The element shows values of an array as a list box. When the visualization user clicks an entry, the array index of the entry is written to an integer variable.

\section*{Element proper- \\ ties}
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Optional \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list. \\
Example: List_Product_Number
\end{tabular} \\
\hline "Type of element" & "Combo Box, Array" \\
\hline
\end{tabular}

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline "X" & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols ( ( \(^{-}\)) to other positions in the editor.

See also
- \({ }^{4}\) Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center' rotation is shown as the \(\oplus\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline " \(Y\) " & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols (ఉ) to other positions in the editor.
\begin{tabular}{|c|c|}
\hline "Variable" & \begin{tabular}{l}
The array index of the list entry that the user clicks is saved at runtime. \\
Property value \\
- Variable (integer data type) \\
Example: PLC_PRG.iIndexComboboxEntry \\
- Enumeration variable with text list support \\
Example: PLC_PRG.eMyCombobox<COMBO> \\
Note: Value range of the enumeration value that lies within the DWORD or DINT value range
\end{tabular} \\
\hline "Data array" & \begin{tabular}{l}
Displayed as a combo box. Every array component becomes a combo box entry. \\
Property value \\
- Array variable (ARRAY [...] OF) \\
Example: PLC_PRG.astrCombobox \\
Declaration: astrCombobox : ARRAY[0..4] OF STRING := \\
['First', 'Second', 'Third', 'Fourth'];
\end{tabular} \\
\hline
\end{tabular}

See also
- Enumerations
- \(\Rightarrow\) Chapter 6.4.5.8 "Setting Up Multiple Languages" on page 2906

Element property 'Columns'

The "Combo box - Array" element visualizes an array variable or structure variable in a tabular view. The index of array elements or structure members is shown in a column or row. Twodimensional arrays or structure arrays are shown in several columns. You specify the visualized variable in the "Data array" property. If a variable is assigned there, then you can specify the display of the table columns where the array elements are shown. You can customize each column that is assigned to an index \([<n>]\).
\begin{tabular}{|c|c|}
\hline \begin{tabular}{l}
"Columns" \\
- [<n>]
\end{tabular} & \begin{tabular}{l}
Due to the structure of the variable that is defined in "Data array", CODESYS determines the number of columns and defines them with the index <n>. \\
Example: StringTable : ARRAY [0..2, 0..4] OF STRING := ['BMW','Audi','Mercedes','VW','Fiat', \\
'150','150','150','150','100','blue','gray','silver','blue' \\
, 'red'];: three columns are formed [0], [1] and [2].
\end{tabular} \\
\hline "Max. array index" & Optional. Variable (integer data type) or value. Defines up to which array index the data is displayed. \\
\hline "Row height" & Height of the rows (in pixels). \\
\hline "Number visible rows" & Optional. If the array is larger than the number of visible rows, then a scrollbar is included. \\
\hline "Scrollbar size" & Width of the vertical scrollbar (in pixels). \\
\hline
\end{tabular}

Table 629: "Element property 'Columns: Column [<n>]""
\begin{tabular}{|l|l|}
\hline "Width" & Column width (in pixels). \\
\hline "Image column" & \begin{tabular}{l} 
V: Images can be displayed in the column. Images are used from the global \\
image pool or user-defined image pools. The image IDs are shown in the cells of \\
the table as defined in the image pool.
\end{tabular} \\
\hline \multicolumn{4}{|l|}{ "Image configuration" } & \begin{tabular}{l} 
- "Fill cell" \\
\begin{tabular}{l} 
The image resizes to the dimensions of the cell without fixing the height/ \\
width ratio. \\
"Centered" \\
The image is centered in the cell and retains its proportions (height-width \\
ratio).
\end{tabular} \\
\hline "Fill mode" \\
\hline "V: The color that is specified in "Transparent color" is displayed as transparent.
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l} 
When the "Transparent" property is enabled, the color specified here is not \\
displayed. Pixels with this color are transparent.
\end{tabular} \\
\hline "Text alignment in column" & \begin{tabular}{l} 
- "Left" \\
- "Centered" \\
- "Right"
\end{tabular} \\
\hline
\end{tabular}

\section*{Element prop-}
erty 'Texts'
\begin{tabular}{|l|l|}
\hline "Tooltip" & \begin{tabular}{l} 
Character string (without single straight quotation marks) that is displayed as the \\
tooltip of an element in runtime mode \\
Example: Products of customer A \\
Hint: The text is accepted automatically into the "GlobalTextList" text list and can \\
be localized there.
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{4}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

Element prop- The properties contain fixed values for the text properties.
erty 'Text prop-
erties'
\begin{tabular}{|c|c|}
\hline "Usage of" & \begin{tabular}{l}
- "Default style values": The values of the visualization style are used. \\
- "Individual settings": The "Individual text properties" property group is shown The values can be customized here.
\end{tabular} \\
\hline \multicolumn{2}{|l|}{"Individual text properties"} \\
\hline \multicolumn{2}{|l|}{Requirement: The "Individual settings" text property is defined.} \\
\hline \multirow[t]{3}{*}{"Font"} & Example: "Default" \\
\hline & ...): The "Font" dialog opens. \\
\hline & - : List box with style fonts \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Font color" & \begin{tabular}{l} 
Example: "Black" \\
\\
\\
\hline... : The "Color" dialog opens. \\
マ: List box with style colors
\end{tabular} \\
\hline & \begin{tabular}{l} 
Integer (value range from 0 to 255). This determines the transparency of the \\
respective color. \\
\(255:\) The color is opaque. \\
\(0:\) The color is completely transparent. \\
Note: If the color is a style color and already has a transparency value, then this \\
property is write-protected.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG. iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG. iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\oplus\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
\hline "Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol. \\
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
\hline
\end{tabular}

\section*{You can link the variables to a unit conversion.}

X, "Y", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also

Element prop- The variables control the element behavior dynamically. erty 'State variables'
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \(=\) FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{\text {a }}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Tabs'}

Symbol:


\section*{Category: "Common Controls"}

The element displays selected visualizations in tabs. The tabs can be used by means of the tab header without having to configure an input configuration. A visualization user switches between visualizations by clicking the tab header.

\section*{Element proper- \\ ties}
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Example: Assembly A \\
Optional \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list.
\end{tabular} \\
\hline "Type of element" & "Tabs" \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Tab width" & \begin{tabular}{l} 
Width of the tab (in pixels). If there is not space for all tab headers, then a scroll \\
bar is added. \\
Example: 30
\end{tabular} \\
\hline "Tab height" & \begin{tabular}{l} 
Height of the tab (in pixels) \\
- Integer literal \\
Example: 15 \\
- "From style"
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Scaling type" & \begin{tabular}{l} 
The method with which the height and width of the referenced visualization are \\
scaled. \\
- "Isotropic": The visualization is scaled to the size of the element. The visuali- \\
- zation retains its proportions with a fixed height/width ratio. \\
- "Anisotropic": The visualization is scaled to the size of the element. The \\
height and width are adapted to the element independently of each other. \\
- "Fixed": the visualization is displayed in its original size without taking into \\
account the size of the element. \\
- "Fixed and scrollable": The visualization is displayed fixed in the element. If it \\
is larger than the element, the element will be provided with scrollbars. \\
Please note: assign variables to the properties ""croll position variable \\
horizontal": "Scroll position variable vertical". You can then edit the data \\
of the scrollbar position in the application.
\end{tabular} \\
\hline
\end{tabular}

\section*{"Deactivate background drawing"}
\(\square\) : The non-animated elements of the referenced visualization are displayed as
background images in order to optimize the performance of the visualization.
Result: At runtime, the elements can be displayed in any order, for example
when an element moves behind the frame at runtime.
包: Deactivates the background display in order to prevent the behavior
described above
The property is not available for the following settings:
- The "Scaling type" property is set to "Fixed and scrollable"
- The client animation functionality is enabled.

Element prop- The properties include variables for the position of the scroll boxes in the scroll bars. You can erty 'Scroll bar process the data for the scroll box position in the application.
settings'
\begin{tabular}{|l|l|}
\hline Requirement: The "Scaling type" property is "Fixed and scrollable". \\
\hline \begin{tabular}{ll}
\hline "Scroll position variable \\
horizontal"
\end{tabular} & \begin{tabular}{l} 
Variable (integer data type, also array). Includes the position of the horizontal or \\
vertical scroll box. The array contains the position for each display variant. If the \\
vertical" \\
\hline "Scroll position variable, \\
visualization is running on multiple display variants, then the position changes \\
are disconnected from each other. \\
Example:
\end{tabular} \\
& PLC_PRG.iScrollHor [CURRENTCLIENTID] \\
& PLC_PRG.iScrollVer[CURRENTCLIENTID] \\
In this example, the variable is declared as an array: \\
& \begin{tabular}{l} 
iScrollHor: ARRAY [0..20] OF INT; \\
iScrollVer: ARRAY [0..20] OF INT;
\end{tabular} \\
& CURRENTCLIENTID indicates the current display variant. \\
\hline
\end{tabular}

See also
- Unit conversion

\section*{Element property 'References'}
\begin{tabular}{|l|l|}
\hline "References" & \begin{tabular}{l} 
Clicking "Configure" opens the "Frame Configuration" dialog. You can select an \\
existing visualization there. \\
Selected visualization references are shown in the properties. \\
Selected visualization references are listed here as subordinate properties.
\end{tabular} \\
\hline \begin{tabular}{l} 
Name pf the visualization refer- \\
ence (example: PLC_PRG.S1)
\end{tabular} & Tab caption (example: Panel) \\
\hline "Heading" & \begin{tabular}{l} 
Image ID in the theme <image pool name>.<ID> \\
Example: Imagepoo__A.1 for the image with ID 1 in Imagepool_A
\end{tabular} \\
\hline "Image ID" & \begin{tabular}{l} 
If the visualization has an interface, then their parameters are displayed here as \\
subordinate properties. \\
Variable (data type conforms to data type of the interface parameter). Includes \\
the initialization value for the instantiation of the visualization.
\end{tabular} \\
\hline \begin{tabular}{l} 
Interface parameter of the visu \\
alization reference \\
Example: ix
\end{tabular} &
\end{tabular}

See also
- ※y Chapter 6.4.5.17 "Creating a structured user interface" on page 2940
- * Chapter 6.4.5.21.2.1 "Command 'Interface Editor'" on page 3340
- \(\boldsymbol{y y}^{\boldsymbol{y}}\) Chapter 6.4.5.21.2.9 "Command 'Frame Selection'" on page 3348

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X "\) & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (ए.) to other positions in the editor.

See also
- \({ }^{*}>\) Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline\(" X "\) & X-coordinate of the point of rotation \\
\hline\(" Y "\) & \(Y\)-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus}\) ) to other positions in the editor.

\section*{Element property 'Switch frame variable'}
\begin{tabular}{|l|l|}
\hline "Variable" & Variable (integer data type). Specifies the index of the active visualization. \\
& Example: PLC_PRG. uiActiveVisuID. \\
& Tip: The "Frame Configuration" dialog box includes a list of selected visualiza- \\
tions. The visualizations are ordered automatically in numeric order in the list. \\
\hline
\end{tabular}

See also
- \({ }^{4}\) Chapter 6.4.5.21.2.9 "Command 'Frame Selection'" on page 3348

Element prop- The properties contain IEC variables for controlling the position of the element dynamically. The erty 'Absolute movement'
reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline " \(Y\) " & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG. iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|}
\hline "Rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle1. \\
The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol. \\
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
\end{tabular} &  \\
\hline "Interior rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle2. \\
In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. \\
The rotation point is shown as the \(\boldsymbol{\oplus}\) symbol. \\
Note: If a static angle of rotation is specified in the "Position \(\rightarrow\) Angle" property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.
\end{tabular} &  \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

The " \(X\) ", " \(Y\) ", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

\section*{Element prop- The variables control the element behavior dynamically. erty 'State varia- \\ bles'}
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \(=\) FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

\section*{The "Invisible" property is supported by the "Client Animation" functionality.}

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

See also
- « \(\boldsymbol{y y}^{2}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Button'}

Symbol:


\section*{Category: "Common Controls"}

The element triggers an action, such as setting a variable.

\section*{Element proper- \\ ties}
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Optional \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list. \\
Example: Voltage_on
\end{tabular} \\
\hline "Type of element" & "Button" \\
\hline
\end{tabular}

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X "\) & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (ए़-) to other positions in the editor.

See also
- \({ }^{4}\) Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline " \(Y\) " & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus}\) ) to other positions in the editor.

Element prop- The properties contain fixed values for setting colors. erty 'Colors'
\begin{tabular}{|l|l|}
\hline "Color" & \begin{tabular}{l} 
Color for the element in its normal state. \\
Please note that the normal state is in effect if the expression in the \\
"Color variables \(\rightarrow\) Toggle color" property is not defined or it has the value \\
FALSE.
\end{tabular} \\
\hline "Alarm color" & \begin{tabular}{l} 
Color for the element in alarm state. \\
Please note that the alarm state is in effect if the expression in the
\end{tabular} \\
"Color variables \(\rightarrow\) Toggle color" property has the value TRUE.
\end{tabular}, \begin{tabular}{l} 
Value (0 to 255) for defining the transparency of the selected color. \\
Example 255: The color is opaque. 0: The color is completely transparent.
\end{tabular}, \begin{tabular}{l} 
"Transparency" The element is displayed with a color gradient.
\end{tabular}

See also
- \({ }^{\star}\) Chapter 6.4.5.21.3.5 "Dialog 'Gradient Editor"' on page 3369
- Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Element prop-}
erty 'Image'
\begin{tabular}{|c|c|}
\hline "Static ID" & \begin{tabular}{l}
Reference to an image in an image pool of the format <name of image pool>.<image ID> (example: image_pool. GreenButton). \\
If the image is from the "GloballmagePool", then you can omit the name of the image pool because CODESYS always searches this pool first.
\(\qquad\) The "Input Assistant" dialog box opens and lists all available image pools and images in the entire project.
\end{tabular} \\
\hline "Scale type" & \begin{tabular}{l}
Behavior of the image when resizing the button. \\
- "Isotropic": The image retains its proportions. The ratio of height to width is retained, even if you change the height or width of the button separately. \\
- "Anisotropic": The image resizes to the dimensions of the button. \\
- "Fixed": The image retains its original size, even if you change the size of the button.
\end{tabular} \\
\hline "Transparency" & The visualization displays the image with the transparency color that is selected in "Transparency color". \\
\hline "Transparency color" & \begin{tabular}{l}
Color that is transparent in the image (example: "White"). if the image background that is reference by "Static ID" is white, then this background is displayed transparent. Clicking ... opens a color selection dialog. \\
Requirement: The "Transparency" option is activated.
\end{tabular} \\
\hline "Horizontal alignment" & \begin{tabular}{l}
Horizontal alignment of the image \\
- "Left" \\
- "Centered" \\
- "Right"
\end{tabular} \\
\hline "Vertical alignment" & \begin{tabular}{l}
Vertical alignment of the image \\
- "Top" \\
- "Centered" \\
- "Bottom"
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contains character strings for labeling the element. The character string can also erty 'Texts' contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.

CODESYS accepts the specified texts automatically into the "GlobalTextList" text list. Therefore, these texts can be localized.
\begin{tabular}{|l|l|}
\hline "Text" & \begin{tabular}{l} 
Character string (without single straight quotation marks) for the labeling the \\
element. Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter]. \\
Example: Accesses: \%i \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Text".
\end{tabular} \\
\hline "Tooltip" & \begin{tabular}{l} 
Character string (without single straight quotation marks) that is displayed as the \\
tooltip of an element. \\
Example: Number of valid accesses. \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Tooltip".
\end{tabular} \\
\hline
\end{tabular}

See also
- «4 "Element property 'Text variables"" on page 3094
- \(y^{*}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872
- \({ }^{4}\) Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329

Element prop- The properties contain fixed values for the text properties.
erty 'Text prop-
erties'
\begin{tabular}{|c|c|}
\hline "Horizontal alignment" & Horizontal alignment of the text within the element. \\
\hline "Vertical alignment" & Vertical alignment of the text within the element. \\
\hline "Text format" & \begin{tabular}{l}
Definition for displaying texts that are too long \\
- "Default": The long text is truncated. \\
- "Line break": The text is split into parts. \\
- "Ellipsis": The visible text ends with "..." indicating that it is not complete.
\end{tabular} \\
\hline "Font" & \begin{tabular}{l}
Example: "Default"
The "Font" dialog box opens. \\
*: Drop-down list with style fonts.
\end{tabular} \\
\hline "Font color" & Example: "Black"
The "Color" dialog box opens.
Drop-down list with style colors. \\
\hline "Transparency" & \begin{tabular}{l}
Whole number (value range from 0 to 255). This determines the transparency of the respective color. \\
Example: 255: The color is opaque. \\
0 : The color is completely transparent. \\
Please note: If the color is a style color and already has a transparency value, then this property is write-protected.
\end{tabular} \\
\hline
\end{tabular}

\section*{Element property 'Absolute movement'}

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|l|}{"Movement"} \\
\hline " \(\times\) " & \begin{tabular}{l}
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline " \(Y\) " & \begin{tabular}{l}
Variable (numeric data type). Defines the \(Y\) position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline "Rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle1. \\
The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol. \\
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
\end{tabular} \\
\hline "Scaling" & \begin{tabular}{l}
Variable (integer data type). Causes centric stretching. \\
Example: PLC_PRG.iScaling. \\
The reference point is the "Center" property. \\
The value 1 shrinks the element by a factor of 0.001 . The value 1000 returns the element to its original size.
\end{tabular} \\
\hline
\end{tabular}

\section*{You can link the variables to a unit conversion.}

Element property 'Relative movement'

The properties "X", "Y", "Rotation", and "Interior rotation" are supported by the "Client Animation" functionality.

\section*{See also}
\begin{tabular}{|l|l|}
\hline "Movement top-left" & \\
\hline "X" & \begin{tabular}{l} 
Variable (integer data type). It contains the number (in pixels) that the left edge \\
is moved horizontally. Incrementing the value moves the element to the right. \\
Example: PLC_PRG. iDeltax
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline " \(Y\) " & \begin{tabular}{l} 
Variable (integer data type). It contains the number (in pixels) that the top edge \\
is moved vertically. Incrementing the value moves the element to the down. \\
Example: PLC_PRG. ideltaY
\end{tabular} \\
\hline "Movement bottom-right" & \\
\hline\(" X "\) & \begin{tabular}{l} 
Variable (integer data type). It contains the number (in pixels) that the right edge \\
is moved horizontally. Incrementing the value moves the element to the right. \\
Example: PLC_PRG.iDeltaWidth
\end{tabular} \\
\hline\(Y "\) & \begin{tabular}{l} 
Variable (integer data type). It contains the number (in pixels) that the bottom \\
edge is moved vertically. Incrementing the value moves the element to the down. \\
Example: PLC_PRG.iDeltaHeight
\end{tabular} \\
\hline
\end{tabular}

See also
- \& "Element property 'Absolute movement"' on page 3016

Element prop- These properties are variables with contents that replace a format definition. erty 'Text variables'
\begin{tabular}{|l|l|}
\hline "Text variable" & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses
\end{tabular} \\
& \begin{tabular}{l} 
Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Text". \\
Note: If you specify a variable of type enumeration with text list support, then \\
the name of the enumeration data type is added automatically in angle brackets \\
after the variable name. Example: PLC_PRG.enVar <enumeration name>. \\
Then the symbolic value of the enumeration component is printed instead of the \\
numeric value when text is printed. Refer to the help page for the enumerations.
\end{tabular} \\
\hline "Tooltip variable" & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses InTooltip \\
Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Tooltip".
\end{tabular} \\
\hline
\end{tabular}

See also
- \& Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329
- © "Element property 'Texts"' on page 3091
- Elapter 6.4.1.20.5.18 "Enumerations" on page 2263

Element prop- Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is erty 'Dynamic texts' currently indexed in the variable.
\begin{tabular}{|c|c|}
\hline "Text list" & \begin{tabular}{l}
Variable (string) or name of the text list as a fixed string in single straight quotation marks. \\
Example: 'Errorlist' \\
v: Drop-down list with the dialogs available in the text lists.
\end{tabular} \\
\hline "Text index" & \begin{tabular}{l}
Text list ID. This refers to the desired output text. \\
- As fixed string with the ID in single straight quotation marks. Example: ' 1 ' \\
- As a variable (STRING) for dynamically controlling the text output. \\
Example: strTextID \\
Sample assignment: PLC_PRG.strTextID := '1';
\end{tabular} \\
\hline "Tooltip index" & \begin{tabular}{l}
Text list ID. This refers to the desired output text. \\
- As fixed string with the ID in single straight quotation marks. Example: '2' \\
- As a variable (STRING) for dynamically controlling the text output. \\
Example: strToolTipID \\
Sample assignment: PLC_PRG.strToolTipID := '2';
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{\circ}\) Chapter 6.4.1.21.2.28 "Object 'Text List"' on page 2532

Element prop- The variables allow for dynamic control of the text display. erty 'Font variables'
\begin{tabular}{|c|c|}
\hline "Font name" & \begin{tabular}{l}
Variable (STRING). Includes the font of the text. \\
Example: PLC_PRG.stFontVar := 'Arial'; \\
The selection of fonts corresponds to the default "Font" dialog.
\end{tabular} \\
\hline "Size" & \begin{tabular}{l}
Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name. \\
- <pt>: Points (default) \\
Example: PLC_PRG.iFontHeight <pt> \\
Code: iFontHeight : INT := 12; \\
- <px>: Pixels \\
Example: PLC_PRG.iFontHeight <px> \\
Code: iFontHeight : INT := 19; \\
If you click in the value field, a drop-down list opens on the right for setting the unit. \\
Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property "Text property \(\rightarrow\) Font".
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Flags" & \begin{tabular}{l}
Variable (DWORD). Contains the flags for displaying fonts. \\
Flags: \\
- 1: Italics \\
- 2: Bold \\
- 4: Underline \\
- 8: Strikethrough \\
Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: PLC_PRG.dwFontType := 6;
\end{tabular} \\
\hline "Character set" & \begin{tabular}{l}
Variable (DWORD). Contains a character set number for the font. \\
The selection of character set numbers corresponds to the "Script" setting of the standard "Font" dialog.
\end{tabular} \\
\hline "Color" & \begin{tabular}{l}
Variable (DWORD). Includes the color of the text. \\
Example: PLC_PRG.dwColorFont:= 16\#FF000000;
\end{tabular} \\
\hline "Flags for text alignment" & \begin{tabular}{l}
Variable (integer data type). Contains the coding for text alignment. \\
Example: PLC_PRG.dwTextAlignment. \\
Coding: \\
- 0 : Top left \\
- 1: Horizontal center \\
- 2: Right \\
- 4: Vertical center \\
- 8: Bottom \\
Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: PLC_PRG.dwFontType := 5;
\end{tabular} \\
\hline
\end{tabular}

Fixed values for displaying texts are set in "Text properties".

See also
- «» "Element property 'Text properties'" on page 3092

Element prop- The Element property is used as an interface for project variables to dynamically control colors erty 'Color varia- at runtime. bles'
\begin{tabular}{|c|c|}
\hline "Toggle color" & \begin{tabular}{l}
The property controls the toggled color at runtime. \\
Value assignment: \\
- FALSE: The element is displayed with the color specified in the "Color" property. \\
- TRUE: The element is displayed with the color specified in the "Alarm color" property. \\
Assigning the property: \\
- Placeholder for the user input variable \\
- "<toggle/tap variable>" \\
- "<NOT toggle/tap variable>" \\
The color change is not controlled by its own variable, but by a user input variable. \\
Note: Specify a variable for the mouse events "Tap" or "Toggle" in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both "Toggle" and "Tap", then the variable specified in "Tap" is used. \\
Hint: Click the symbol to insert the placeholder "<toggle/tap variable>". When you activate the "Inputconfiguration", "Tap FALSE" property, then the "<NOT toggle/tap variable>" placeholder is displayed. \\
- Instance path of a project variable (BOOL) \\
Example: PLC_PRG.xColorIsToggeled \\
Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.
\end{tabular} \\
\hline "Color" & \begin{tabular}{l}
Color variable for the Frame \\
- Variable (DWORD) for the color \\
Example: PLC_PRG.dwColor \\
- Color literal \\
Example of gray and opaque: 16\#FF888888 \\
Requirement: "Show Frame" property is activated. \\
Please note that the normal state is in effect if the expression in the "Colorvariables \(\rightarrow\) Toggle color" property is not defined or it has the value FALSE.
\end{tabular} \\
\hline "Alarm color" & \begin{tabular}{l}
Color variable for the Frame in alarm state \\
- Variable (DWORD) for the alarm color \\
Example: PLC_PRG.dwAlarmColor \\
- Color literal \\
Example of red and opaque: 16\#FFFF0000 \\
Please note that the alarm state is in effect if the expression in the "Colorvariables \(\rightarrow\) Toggle color" property has the value TRUE.
\end{tabular} \\
\hline
\end{tabular}

The transparency part of the color value is evaluated only if the "Activate semitransparent drawing" option of the visualization manager is selected.

Select the "Advanced" option in the toolbar of the properties view. Then all element properties are visible.

See also
- « Chapter 6.4.5.10.4 "Animating a color display" on page 2914
- \& Chapter 6.4.5.21.4.2 "Object 'Visualization manager'" on page 3398

Element prop- The variables control the element behavior dynamically. erty 'State variables'
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \(=\) FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
\begin{tabular}{l} 
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular} \\
\hline
\end{tabular}

> The "Invisible" property is supported by the "Client Animation" functionality.

\section*{Element prop- \\ erty 'Button \\ state variable'}
\begin{tabular}{|c|c|}
\hline "Digital variable" & \begin{tabular}{l}
At runtime, the property controls whether the Button is displayed as pressed or not. \\
Values: \\
- FALSE: The Button is displayed as not pressed. \\
- TRUE: The Button is displayed as pressed. \\
Argument passed to the property: \\
- Placeholder for the user input variable to couple the representation of the Button with the input variable. \\
- "<toggle/tap variable>" \\
- "<NOT toggle/tap variable>" \\
Note: Specify a variable for the mouse events "Tap" or "Toggle" in the input configuration of the Button. Only then is the placeholder set. If you configure a variable in both "Toggle" and "Tap", then the variable specified in "Tap" is used. \\
Hint: Click the symbol i to insert the placeholder "<toggle/tap variable>". When you activate the "Inputconfiguration", "Tap FALSE" property, then the "<NOT toggle/tap variable>" placeholder is displayed. \\
- Instance path of a project variable (BOOL) \\
Example: prgA. xButtonState \\
Note: Implement a value assignment in the code for the variable specified here.
\end{tabular} \\
\hline
\end{tabular}

\section*{Element property 'Image ID variable'}
\begin{tabular}{|l|l|}
\hline "Image ID" & \begin{tabular}{l} 
Variable (STRING). Contains the image ID. The contents of the string corre- \\
sponds to the description of the "Static ID" property. \\
Example: PLC_PRG.stImageID \(:=\) 'ImagePool_A.Image3';
\end{tabular} \\
\hline
\end{tabular}

See also
- H Chapter 6.4.5.20.1.5 "Visualization Element 'Image"' on page 3038
- \({ }^{\text {• }}\), Chapter 6.4.1.21.2.15 "Object 'Image Pool'" on page 2468

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain the configurations for the user input when using the mouse or keyboard. erty 'Input con- A user input defines an event and one or more actions that are executed when an event occurs. figuration'
\begin{tabular}{|l|l|}
\hline \multicolumn{1}{|l|}{ The "Configure" button opens the "Input Configuration" dialog. There you can create or edit user inputs. } \\
Configured user inputs are listed below the events. They each include the action that is triggered and the setting \\
in short form. \\
Example: "Execute ST Code": 3 PLC_PRG.i_x \(:=0 ;\) \\
\hline "OnDialogClosed" & Input event: The user closes the dialog. \\
\hline "OnMouseClick" & \begin{tabular}{l} 
Input event: The user clicks the mouse button completely in the element area. \\
The mouse button is clicked and released.
\end{tabular} \\
\hline "OnMouseDown" & Input event: The user clicks down on the mouse button. \\
\hline "OnMouseEnter" & Input event: The user drags the mouse pointer to the element. \\
\hline "OnMouseLeave" & Input event: The user drags the mouse pointer away from the element. \\
\hline
\end{tabular}
\(\left.\begin{array}{|l|l|}\hline \text { "OnMouseMove" } & \text { Input event: The user moves the mouse pointer over the element area. } \\ \hline \text { "OnMouseUp" } & \begin{array}{l}\text { Input events: } \\ \text { - } \\ \text { The user releases the mouse button within the element area. It is irrelevant } \\ \text { whether the user has previously pressed the mouse button inside or outside } \\ \text { the element area. }\end{array} \\ \text { - The user presses the mouse button within the element area, leaves the } \\ \text { element area, and then releases the mouse button. } \\ \text { Note: This CODESYS-specific triggering behavior guarantees that actions for } \\ \text { key elements are completed. A key element starts an action for "OnMouseDown" } \\ \text { and ends the action for "OnMouseUp". } \\ \text { Example: A visualization user presses the mouse button within the element area } \\ \text { of the key element and then moves the cursor position so that it lies outside the } \\ \text { element area. The action is ended anyway because "OnMouseUp" is triggered. }\end{array}\right\}\)
\begin{tabular}{|l|l|}
\hline "Tap" & \begin{tabular}{l} 
When a mouse click event occurs, the variable defined in "Variable" is described \\
in the application. The coding depends on the "Tap FALSE" and "Tap on enter if \\
captured" options.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL) that is set on mouse click event. \\
Example: PLC_PRG . bI sTapped \\
TRUE: A mouse click event exists. It lasts as long as the user presses the mouse \\
button over the element. It ends when the button is released.
\end{tabular} \\
& \begin{tabular}{l} 
FALSE: A mouse click event does not exist. \\
Requirement: The "Tap FALSE" option is not activated.
\end{tabular} \\
\hline "Tap FALSE" & \begin{tabular}{l} 
V: The mouse click event leads to a complementary value in "Variable". \\
TRUE: A mouse click event does not exist.
\end{tabular} \\
\hline "Tap on enter if captured" & \begin{tabular}{l} 
FALSE: While the mouse click event exists.
\end{tabular} \\
\hline \begin{tabular}{l} 
pointer is dragged within the element area or not while the mouse button is \\
pressed. \\
TRUE: While the mouse click event exists and the mouse pointer is moved over \\
the element area. \\
FALSE: A mouse click event does not exist. Or the user moves the mouse \\
pointer outside of the element area while the mouse button is pressed. \\
The value is TRUE again as soon as the user moves the pointer back to the \\
element area. The mouse is then captured.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Toggle" & \begin{tabular}{l} 
With the onset of a mouse click event, the variable is set; when the mouse click \\
event is completed, the variable is reset.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL). Its value toggled when the mouse click event is ended. This is \\
when the user releases the mouse button while the mouse pointer is over the \\
element area. \\
If the user releases the mouse button while the mouse pointer is outside of the \\
element area, then the mouse click event is not ended and the value is not \\
toggled. \\
Hint: The user can cancel a started toggle input by dragging the mouse pointer \\
out of the element area.
\end{tabular} \\
\hline "Toggle on up if captured" & \begin{tabular}{l} 
V: The value toggles regardless of where the mouse pointer is when the mouse \\
button is released. The mouse is then captured.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Hotkey" & Keyboard shortcut on the element for triggering specific input actions. When the keyboard shortcut event occurs, the input actions in the "Events" property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action. \\
\hline "Key" & \begin{tabular}{l}
Key pressed for input action. \\
Example: [T] \\
Note: The following properties appear when a key is selected.
\end{tabular} \\
\hline "Events" & \begin{tabular}{l}
- "None" \\
- "Mouse down": Pressing the key triggers the input actions that are configured in the "OnMouseDown" property. \\
- "Mouse up": Releasing the key triggers the input actions that are configured in the "OnMouseUp" property. \\
- "Mouse down/up": Pressing and releasing the key triggers the input actions that are configured in the "OnMouseDown" property and the "OnMouseUp" property.
\end{tabular} \\
\hline "Shift" & Combination with the Shift key Example: [Shift]+[T]. \\
\hline "Control" & Combination with the Ctrl key Example: [Ctrl]+[T]. \\
\hline "Alt" & \(\square\) : Combination with the Alt key Example: \([A I t]+[T]\). \\
\hline
\end{tabular}

All keyboard shortcuts and their actions that are configured in the visualization are listed on the "Keyboard Configuration" tab.

\section*{See also}
- ⓨ Chapter 6.4.5.21.2.2 "Command 'Keyboard Configuration'" on page 3341
- \(\xi^{2}\) Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration'" on page 3370

Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'
\begin{tabular}{|c|c|}
\hline "Access rights" & \begin{tabular}{l}
Opens the "Access rights" dialog. There you can edit the access privileges for the element. \\
Status messages: \\
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
- "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline \multicolumn{2}{|l|}{See also} \\
\hline & 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366 \\
\hline
\end{tabular}

\section*{Visualization Element 'Group Box'}

Symbol:


\section*{Category: "Common Controls"}

The element provides a visual grouping of visualization elements. The group box can have multiple levels of nesting.

You can also use drag\&drop to add elements to a "Group Box". To do this, drag the element to the window area of the "Group Box". The appearance of the cursor changes (a small plus sign is displayed). When you click the [Shift] key at the same time, the element is not added.
You can remove elements from the "Group Box" by dragging them out of the window area.

\section*{Element proper- \\ ties}
\begin{tabular}{|l|l|}
\hline "Element name" & Example: Parameter axis 1 \\
& \begin{tabular}{l} 
Optional \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list.
\end{tabular} \\
\hline "Type of element" & "Group Box" \\
\hline "Clipping" & \(\boxed{\square}:\) Elements that protrude beyond the size of the group box are clipped. \\
\hline
\end{tabular}

Element prop- The position defines the location and size of the element in the visualization window. These erty 'Position' are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X\) " & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (ए-) to other positions in the editor.

See also
- 出 Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center' rotation is shown as the \(\boldsymbol{\epsilon}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline "Y" & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus}\) ) to other positions in the editor.

Element prop- The properties contains character strings for labeling the element.
CODESYS accepts the specified texts automatically into the "GlobalTextList" text list. Therefore, these texts can be localized.
\begin{tabular}{|l|l|}
\hline "Text" & \begin{tabular}{l} 
Character string (without single straight quotation marks) for the labeling the \\
element. \\
Example: Axis 1.
\end{tabular} \\
\hline "Tooltip" & \begin{tabular}{l} 
Character string (without single straight quotation marks) that is displayed as the \\
tooltip of an element. \\
Example: Parameters of Axis 1.
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\xi^{4}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

Element prop- The properties contain fixed values for the text properties.
erty 'Text properties'
\begin{tabular}{|l|l|}
\hline "Font" & \begin{tabular}{l} 
Example: "Default" \\
\(\ldots . .:\) The "Font" dialog box opens. \\
v: Drop-down list with style fonts.
\end{tabular} \\
\hline "Font color" & \begin{tabular}{l} 
Example: "Black" \\
\(\ldots . .\). The "Color" dialog box opens. \\
v: Drop-down list with style colors.
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l} 
Whole number (value range from 0 to 255 ). This determines the transparency of \\
the respective color. \\
Example: \(255:\) The color is opaque. \\
\(0:\) The color is completely transparent. \\
Please note: If the color is a style color and already has a transparency value, \\
then this property is write-protected.
\end{tabular} \\
\hline
\end{tabular}

\section*{Element property 'Absolute movement'}

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|c|c|c|}
\hline \multicolumn{3}{|l|}{"Movement"} \\
\hline "X" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular}} \\
\hline "Y" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the \(Y\) position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular}} \\
\hline "Rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle1. \\
The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol. \\
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
\end{tabular} &  \\
\hline "Scaling" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (integer data type). Causes centric stretching. \\
Example: PLC_PRG.iScaling. \\
The reference point is the "Center" property. \\
The value 1 shrinks the element by a factor of 0.001 . The value 1000 returns the element to its original size.
\end{tabular}} \\
\hline "Interior rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle2. \\
In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol. \\
Note: If a static angle of rotation is specified in the property "Position \(\rightarrow\) Angle", then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.
\end{tabular} &  \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

See also
erty 'State varia-
bles'
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \(=\) FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

See also
- Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Table'}

\section*{Symbol:}


\section*{Category: "Common Controls"}

The element displays data that can be represented as an array in a table. Therefore, the data type of the visualizing variable can be 1) a one-dimensional array, 2) a maximum two-dimensional array, 3) an array of an array, 4) an array of structures, or 5) an array of a function block.

\section*{Element proper-}
ties
\begin{tabular}{|c|c|}
\hline "Element name" & \begin{tabular}{l}
Example: Data set component 1 \\
Optional \\
Hint: Assign individual names for elements so that they are found faster in the element list.
\end{tabular} \\
\hline "Type of element" & Table \\
\hline "Data array" & \begin{tabular}{l}
Array whose data is visualized as a table \\
Variable (ARRAY) whose data type determines the number of columns and rows in the table \\
Array types \\
- One-dimensional array: The table has one column. \\
- Two-dimensional array: The second dimension determines the number of columns. \\
- Array of an array: The number of array elements of the back array determines the number of columns. \\
- Array of a structure: The number of structure members determines the number of columns. \\
- Array of a function block: The number of local variables determines the number of columns. \\
Example: PLC_PRG.aiTable \\
Declaration: aiTable : ARRAY[0..3, 0..4] OF INT := [4(1, 2, 3, 4, 5)]; \\
Hint: If the declaration of the array changes, then the table can be refreshed by placing the cursor in the data array value field and pressing the [Enter] key.
\end{tabular} \\
\hline "Max. array index" & \begin{tabular}{l}
Top index limit for the displayed table. Limits the number of displayed rows. The index begins at 0 . \\
- Variable (integer data type) \\
Example: PLC_PRG.iUpperIndexBoundToDisplay \\
- Integer literal \\
Example: 4 is displayed as 5 in the row of the table.
\end{tabular} \\
\hline
\end{tabular}

See also
- Data Type 'ARRAY'

Element property 'Columns'

The "Table" element shows the values of a variable in a tabular view. The array elements of structure members are shown in a column or in a row. Two-dimensional arrays or arrays of a structure are shown in multiple columns. The visualized variable is defined in the "Data array" property. When a variable is assigned there, you can specify the display of the Table columns where the array elements are shown. An individual configuration is possible for each column that is assigned to an index [ \(<n>\) ].
\begin{tabular}{|l|l|}
\hline "Show row header" & \begin{tabular}{l} 
: The row header is visible. \\
\\
Example: For an array, the index of the array element is displayed in the header.
\end{tabular} \\
\hline "Show column header" & \(\boxed{\square}:\) The column label is visible. \\
\hline "Row height" & Height of the rows (in pixels) \\
\hline "Row header width" & Width of the row label \\
\hline "Scroll bar size" " & Size of the scroll bar (in pixels) \\
\hline
\end{tabular}

Table 630: "Element property 'Columns: Column [<n>]""
\(\left.\begin{array}{|l|l|}\hline \text { "Column header" } & \begin{array}{l}\text { By default, the name of the array or structure is applied as the heading with } \\
\text { the index or structure member for the column. If an array of a function block } \\
\text { has been selected for "Data array", then the name of the array is applied to the } \\
\text { column header with the local variables of the function block that belong to the } \\
\text { column. } \\
\text { The column label can be changed here by specifying a new title. }\end{array} \\
\hline \text { "Width" } & \text { Column width (in pixels) }\end{array} \right\rvert\,\)\begin{tabular}{l|l|l|}
\hline \multicolumn{1}{l}{ "Image pool or custom image pools. The image IDs are shown in the cells of the } \\
Table as they are defined in the image pool.
\end{tabular}\(\}\)

See also
- \# Chapter 6.4.5.20.1.1 "Visualization Element 'Rectangle', 'Rounded Rectangle', 'Ellipse'" on page 2988

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X "\) & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline " \(Y\) " & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (炠) to other positions in the editor.

See also
- Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center' rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline "X" & X-coordinate of the point of rotation \\
\hline\(" Y\) " & \(Y\)-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus}\) ) to other positions in the editor.

Element prop- The properties contain fixed values for the text properties.
erty 'Text prop-
erties'
\begin{tabular}{|l|l|}
\hline "Horizontal alignment" & Horizontal alignment of the text within the element. \\
\hline "Vertical alignment" & Vertical alignment of the text within the element. \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Font" & Example: "Default"
The "Font" dialog box opens.
: Drop-down list with style fonts. \\
\hline "Font color" & \begin{tabular}{l}
Example: "Black"
The "Color" dialog box opens. \\
*: Drop-down list with style colors.
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l}
Whole number (value range from 0 to 255). This determines the transparency of the respective color. \\
Example: 255: The color is opaque. \\
0 : The color is completely transparent. \\
Please note: If the color is a style color and already has a transparency value, then this property is write-protected.
\end{tabular} \\
\hline
\end{tabular}

Element prop- Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is erty 'Dynamic currently indexed in the variable. texts'
\begin{tabular}{|c|c|}
\hline "Text list" & \begin{tabular}{l}
Variable (string) or name of the text list as a fixed string in single straight quotation marks. \\
Example: 'Errorlist' \\
- : Drop-down list with the dialogs available in the text lists.
\end{tabular} \\
\hline "Text index" & \begin{tabular}{l}
Text list ID. This refers to the desired output text. \\
- As fixed string with the ID in single straight quotation marks. Example: '1' \\
- As a variable (STRING) for dynamically controlling the text output. \\
Example: strTextID \\
Sample assignment: PLC_PRG.strTextID := '1';
\end{tabular} \\
\hline "Tooltip index" & \begin{tabular}{l}
Text list ID. This refers to the desired output text. \\
- As fixed string with the ID in single straight quotation marks. Example: '2' \\
- As a variable (STRING) for dynamically controlling the text output. \\
Example: strToolTipID \\
Sample assignment: PLC_PRG.strToolTipID := '2';
\end{tabular} \\
\hline
\end{tabular}

See also
- \& Chapter 6.4.1.21.2.28 "Object 'Text List"' on page 2532

Element prop- The variables enable dynamic control of the text display. erty 'Font varia-
bles'
\begin{tabular}{|l|l|}
\hline "Font name" & \begin{tabular}{l} 
Variable (STRING). Includes the font of the text. \\
Example: PLC_PRG.stFontVar \(:=~ ' A r i a l ' ~ ; ~\)
\end{tabular} \\
The selection of fonts corresponds to the default "Font" dialog box.
\end{tabular}\(|\)\begin{tabular}{ll} 
VSize" & \begin{tabular}{l} 
Variable (integer data type). Contains the font size (in pixels). \\
The selection of font sizes corresponds to the default "Font" dialog box.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Flags" & \begin{tabular}{l}
Variable (DWORD). Contains the flags for displaying fonts. \\
Flags: \\
- 1: Italics \\
- 2: Bold \\
- 4: Underline \\
- 8: Strikethrough \\
Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: PLC_PRG.dwFontType := 6;
\end{tabular} \\
\hline "Charset" & \begin{tabular}{l}
Variable (DWORD). Contains a character set number for the font. \\
The selection of character set numbers corresponds to the "Script" setting of the standard "Font" dialog box.
\end{tabular} \\
\hline "Color" & \begin{tabular}{l}
Variable (DWORD). Includes the color of the text. \\
Example: PLC_PRG.dwColorFont:= 16\#FF000000;
\end{tabular} \\
\hline "Flags for text alignment" & \begin{tabular}{l}
Variable (integer data type). Contains the coding for text alignment. \\
Example: PLC_PRG.dwTextAlignment. \\
Coding: \\
- 0: Top left \\
- 1: Horizontal center \\
- 2: Right \\
- 4: Vertical center \\
- 8: Bottom \\
Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: PLC_PRG.dwFontType := 5;
\end{tabular} \\
\hline
\end{tabular}

\section*{Fixed values for displaying texts are set in "Text properties".}

\section*{See also}
- «ヶ "Element property 'Text properties'" on page 3116

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline \multicolumn{1}{|l|}{ "Movement" } & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline\(" Y "\) & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\oplus\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
"Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PPRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\boldsymbol{\oplus}\) symbol.
\end{tabular} \\
\begin{tabular}{l} 
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular}
\end{tabular}

You can link the variables to a unit conversion.

The " \(X\) ", " "", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

\section*{See also}

\section*{Element prop- The variables control the element behavior dynamically. erty 'State varia- \\ bles'}
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL := FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

\section*{The "Invisible" property is supported by the "Client Animation" functionality.}

Element prop-
erty 'Selection'
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
"Background color on \\
selection"
\end{tabular} & Fill color of the selected row. \\
\hline "Selection font color" & Font color of the selected row. \\
\hline "Selection type" & \begin{tabular}{l} 
Selection when clicking the table row. \\
- \\
- No selection: No selection \\
- Cell select ion: The clicked cell only. \\
- Row selection: Row of the clicked cell. \\
- Column selection: Column of the clicked cell. \\
- Row and col umn select ion: Row and column of the clicked cell.
\end{tabular} \\
\hline "Frame around selected cells" & V: A frame is drawn around the selected cells.
\end{tabular}

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|l|l|}
\hline "Animation duration" & \begin{tabular}{l} 
Defines the duration (in milliseconds) in which the element runs an animation \\
- \begin{tabular}{l} 
Variable (integer value) \\
Example: Menu. tContent with VAR tContent : INT \(:=500 ;\) \\
END_VAR \\
- Integer literal \\
Example: 500
\end{tabular} \\
Animatable properties \\
- "Absolute movement", "Movement", "X", " \(Y\) " \\
- "Absolute movement", ""otation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable \\
property has changed. The movement then executed is not jerky, but is smooth \\
within the specified animation duration. The visualization element travels to the \\
specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline \begin{tabular}{l} 
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIs InForeground with VAR bIsInForeground : BooL \(:=\) \\
FALSE; END_VAR
\end{tabular} \\
\hline "Move to foreground" \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it \\
was inserted in the visualization editor.
\end{tabular}

Element prop- Requirement: User management is set up for the visualization. erty 'Access
rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages:
\end{tabular} \\
& \begin{tabular}{l} 
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
\(\bullet \quad\) "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

See also
- ※y Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

See also
- 乡y Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872- 乡y Chapter 6.4.5.11.2 "Displaying Array Variables in Tables" on page 2917
- Data Type 'ARRAY'

\section*{Visualization Element 'Text Field'}

Symbol:

Category: "Common Controls"

The element is used for the following purposes:
- Static output of text. The contents of a variable can be part of the text.
- Showing a tooltip. The text is managed as static text and can also be defined so that the contents of a variable are also displayed.
- Dynamic output of text. Texts of a text list are displayed dynamically.
- Input of text. For example, a user can input a number or a text literal.

See also
- \(\quad{ }^{2}>\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Element proper- \\ ties}
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Optional \\
Example: FileName_A \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list.
\end{tabular} \\
\hline "Type of element" & "Text Field" \\
\hline
\end{tabular}

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X "\) & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (ए़-) to other positions in the editor.

See also
- \(\Rightarrow\) Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

\section*{Element property 'Colors'}
\begin{tabular}{|l|l|}
\hline "Normal state" & \begin{tabular}{l} 
The normal state is in effect if the variable in "Color variables \(\boldsymbol{\rightarrow}\) Toggle color" is \\
not defined or it has the value FALSE.
\end{tabular} \\
\hline "Frame color" & Frame and fill color for the corresponding state of the variable. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Fill color" & \\
\hline "Transparency" & \begin{tabular}{l} 
Transparency value (0 to 255\()\) for defining the transparency of the selected color. \\
Example: 255: The color is opaque. 0: The color is completely transparent.
\end{tabular} \\
\hline "Alarm state" & \begin{tabular}{l} 
The alarm state is in effect if the variable in "Color variables \(\rightarrow\) Toggle color" has \\
the value TRUE.
\end{tabular} \\
\hline
\end{tabular}

See also
- « Chapter 6.4.5.21.3.5 "Dialog 'Gradient Editor'" on page 3369

Element prop- The properties contain fixed values for setting the look of the element. erty 'Appearance'
\begin{tabular}{|c|c|}
\hline "Line width" & \begin{tabular}{l}
Value in pixels \\
Example: 2 \\
Note: The values 0 and 1 both result in a line weight of 1 pixel. If no line should be displayed, then the "Line style" property must be set to the option "Invisible".
\end{tabular} \\
\hline "Fill attributes" & \begin{tabular}{l}
The way in which the element is filled. \\
- "Filled":The element is filled with the color from property "Colors \(\rightarrow\) Fill color". \\
- "Invisible": The fill color is invisible.
\end{tabular} \\
\hline "Line style" & \begin{tabular}{l}
Type of line representation \\
- "Solid" \\
- "Dashes" \\
- "Dots" \\
- "Dash Dot" \\
- "Dash Dot Dot" \\
- "not visible"
\end{tabular} \\
\hline
\end{tabular}

You can assign variables in the "Appearance variables" property for controlling the appearance dynamically. The fixed values here are overwritten.

\section*{See also}
- « "Element property 'Appearance variables'" on page 3064

Element property 'Texts'

The properties contains character strings for labeling the element. The character string can also contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.
CODESYS accepts the specified texts automatically into the "GlobalTextList" text list. Therefore, these texts can be localized.
\begin{tabular}{|l|l|}
\hline "Text" & \begin{tabular}{l} 
Character string (without single straight quotation marks) for the labeling the \\
element. Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter]. \\
Example: Accesses: \%i \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Text".
\end{tabular} \\
\hline "Tooltip" & \begin{tabular}{l} 
Character string (without single straight quotation marks) that is displayed as the \\
tooltip of an element. \\
Example: Number of valid accesses. \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Tooltip".
\end{tabular} \\
\hline
\end{tabular}

See also
- をy "Element property 'Text variables'" on page 3116
- \(\xi^{\star}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872
- \({ }^{4}\) Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329

Element prop- The properties contain fixed values for the text properties. erty 'Text properties'
\begin{tabular}{|l|l|}
\hline "Horizontal alignment" & Horizontal alignment of the text within the element. \\
\hline "Vertical alignment" & Vertical alignment of the text within the element. \\
\hline "Text format" & \begin{tabular}{l} 
Definition for displaying texts that are too long \\
- "Default": The long text is truncated. \\
- "Line break": The text is split into parts. \\
- "Ellipsis": The visible text ends with "..." indicating that it is not complete.
\end{tabular} \\
\hline "Font" & \begin{tabular}{l} 
Example: "Default" \\
...: The "Font" dialog box opens. \\
-: Drop-down list with style fonts.
\end{tabular} \\
\hline "Font color" & \begin{tabular}{l} 
Example: "Black" \\
(..: The "Color" dialog box opens. \\
v: Drop-down list with style colors.
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l} 
Whole number (value range from 0 to 255). This determines the transparency of \\
the respective color. \\
Example: \(255:\) The color is opaque. \\
\(0:\) The color is completely transparent.
\end{tabular} \\
\hline
\end{tabular}

\footnotetext{
Element prop- These properties are variables with contents that replace a format definition. erty 'Text variables'
}
\begin{tabular}{|l|l|}
\hline "Text variable" & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses
\end{tabular} \\
& \begin{tabular}{l} 
Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Text". \\
Note: If you specify a variable of type enumeration with text list support, then \\
the name of the enumeration data type is added automatically in angle brackets \\
after the variable name. Example: PLC_PRG.enVar <enumeration name>. \\
Then the symbolic value of the enumeration component is printed instead of the \\
numeric value when text is printed. Refer to the help page for the enumerations.
\end{tabular} \\
\hline "Tooltip variable" & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses InTooltip \\
Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Tooltip".
\end{tabular} \\
\hline
\end{tabular}

See also
- Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329
- © "Element property 'Texts"' on page 3115
- E Chapter 6.4.1.20.5.18 "Enumerations" on page 2263

Element prop- Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is erty 'Dynamic currently indexed in the variable. texts'
\begin{tabular}{|c|c|}
\hline "Text list" & \begin{tabular}{l}
Variable (string) or name of the text list as a fixed string in single straight quotation marks. \\
Example: 'Errorlist' \\
: Drop-down list with the dialogs available in the text lists.
\end{tabular} \\
\hline "Text index" & \begin{tabular}{l}
Text list ID. This refers to the desired output text. \\
- As fixed string with the ID in single straight quotation marks. Example: '1' \\
- As a variable (STRING) for dynamically controlling the text output. Example: strTextID \\
Sample assignment: PLC_PRG.strTextID := '1';
\end{tabular} \\
\hline "Tooltip index" & \begin{tabular}{l}
Text list ID. This refers to the desired output text. \\
- As fixed string with the ID in single straight quotation marks. \\
Example: ' 2 ' \\
- As a variable (STRING) for dynamically controlling the text output. \\
Example: strToolTipID \\
Sample assignment: PLC_PRG.strToolTipID := '2';
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{*}\) Chapter 6.4.1.21.2.28 "Object 'Text List"' on page 2532

\footnotetext{
Element prop- The variables allow for dynamic control of the text display. erty 'Font variables'
}
\begin{tabular}{|c|c|}
\hline "Font name" & \begin{tabular}{l}
Variable (STRING). Includes the font of the text. \\
Example: PLC_PRG.stFontVar := 'Arial'; \\
The selection of fonts corresponds to the default "Font" dialog.
\end{tabular} \\
\hline "Size" & \begin{tabular}{l}
Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name. \\
- <pt>: Points (default) \\
Example: PLC_PRG.iFontHeight <pt> \\
Code: iFontHeight : INT := 12; \\
- <px> : Pixels \\
Example: PLC_PRG.iFontHeight <px> \\
Code: iFontHeight : INT := 19; \\
If you click in the value field, a drop-down list opens on the right for setting the unit. \\
Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property "Text property \(\rightarrow\) Font".
\end{tabular} \\
\hline "Flags" & \begin{tabular}{l}
Variable (DWORD). Contains the flags for displaying fonts. \\
Flags: \\
- 1: Italics \\
- 2: Bold \\
- 4: Underline \\
- 8: Strikethrough \\
Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: PLC_PRG.dwFontType :=6;
\end{tabular} \\
\hline "Character set" & \begin{tabular}{l}
Variable (DWORD). Contains a character set number for the font. \\
The selection of character set numbers corresponds to the "Script" setting of the standard "Font" dialog.
\end{tabular} \\
\hline "Color" & \begin{tabular}{l}
Variable (DWORD). Includes the color of the text. \\
Example: PLC_PRG.dwColorFont:= 16\#FF000000;
\end{tabular} \\
\hline "Flags for text alignment" & \begin{tabular}{l}
Variable (integer data type). Contains the coding for text alignment. \\
Example: PLC_PRG.dwTextAlignment. \\
Coding: \\
- 0 : Top left \\
- 1: Horizontal center \\
- 2: Right \\
- 4: Vertical center \\
- 8: Bottom \\
Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: PLC_PRG.dwFontType := 5;
\end{tabular} \\
\hline
\end{tabular}

Fixed values for displaying texts are set in "Text properties".

\section*{See also}
- « "Element property 'Text properties'" on page 3116

Element prop- The Element property is used as an interface for project variables to dynamically control colors erty 'Color varia- at runtime.

\section*{bles'}
\begin{tabular}{|c|c|}
\hline "Toggle color" & \begin{tabular}{l}
The property controls the toggled color at runtime. \\
Value assignment: \\
- FALSE: The element is displayed with the color specified in the "Color" property. \\
- TRUE: The element is displayed with the color specified in the "Alarm color" property. \\
Assignment options: \\
- Placeholder for the user input variable \\
- "<toggle/tap variable>" \\
- "<NOT toggle/tap variable>" \\
The color change is not controlled by its own variable, but by a user input variable. \\
Note: Specify a variable for the mouse events "Tap" or "Toggle" in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both "Toggle" and "Tap", then the variable specified in "Tap" is used. \\
Hint: Click the symbol to insert the placeholder "<toggle/tap variable>". When you activate the "Inputconfiguration", "Tap FALSE" property, then the "<NOT toggle/tap variable>" placeholder is displayed. \\
- Instance path of a project variable (BOOL) \\
Example: PLC_PRG.xColorIsToggeled \\
Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.
\end{tabular} \\
\hline \begin{tabular}{l}
"Normal state" \\
"Alarm state"
\end{tabular} & The properties listed below control the color depending on the state. The normal state is in effect if the variable in "Color variables", "Toggle colo"" is not defined or it has the value FALSE. The alarm state is in effect if the variable in "Colorvariables", "Toggle color" has the value TRUE. \\
\hline "Frame color" & \begin{tabular}{l}
Assignment options: \\
- Variable (DWORD) for the frame color Example: PLC_PRG.dwBorderColor \\
- Color literal Example of green and opaque: 16\#FF00FF00
\end{tabular} \\
\hline "Filling color" & \begin{tabular}{l}
Assignment options: \\
- Variable (DWORD) for the fill color Example: PLC_PRG.dwFillColor \\
- Color literal Example of gray and opaque: 16\#FF888888
\end{tabular} \\
\hline
\end{tabular}


See also
- \({ }^{4}\) Chapter 6.4.5.10.4 "Animating a color display" on page 2914

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" & \multicolumn{1}{|l|}{\begin{tabular}{|l|}
\hline Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular}} \\
\hline " \(Y\) " & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\mp\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol.
\end{tabular} \\
\begin{tabular}{l} 
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

The "X", " \(Y\) ", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also
-

Element prop- The variables control the element behavior dynamically.
erty 'State varia-
bles'
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
\\
\\
\hline TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \(=\) FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Selection and caret configuration'

The "Invisible" property is supported by the "Client Animation" functionality.


The variables allow for controlling the caret position and the selection of the text.

\begin{tabular}{|l|l|}
\hline "Caret position" & Variable (integer data type). Contains the position of the cursor. \\
\hline "Selection start" & \begin{tabular}{l} 
Variable (integer data type). Contains the position of the first selected character. \\
Example: PLC_PRG. iSelStart
\end{tabular} \\
\hline "Selection end" & \begin{tabular}{l} 
Variable (integer data type). Contains the position of the last selected character. \\
Example: PLC_PRG.iSelEnd
\end{tabular} \\
\hline "All selected" & \begin{tabular}{l} 
Variable (BOOL). Toggles the selection of the entered text. \\
TRUE: The text in the text field is selected.
\end{tabular} \\
\begin{tabular}{l} 
FALSE: The selection starts with the value in "Selection start" and ends with \\
"Selection end".
\end{tabular} \\
\hline
\end{tabular}

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain the configurations for the user input when using the mouse or keyboard. erty 'Input con- A user input defines an event and one or more actions that are executed when an event occurs. figuration'

The "Configure" button opens the "Input Configuration" dialog. There you can create or edit user inputs.
Configured user inputs are listed below the events. They each include the action that is triggered and the setting in short form.
Example: "Execute ST Code": ЗPLC_PRG.i_x := 0;
\begin{tabular}{|l|l}
\hline "OnDialogClosed" & Input event: The user closes the dialog.
\end{tabular}
\begin{tabular}{|l|l|}
\hline ＂OnMouseClick＂ & \begin{tabular}{l} 
Input event：The user clicks the mouse button completely in the element area． \\
The mouse button is clicked and released．
\end{tabular} \\
\hline ＂OnMouseDown＂ & Input event：The user clicks down on the mouse button． \\
\hline ＂OnMouseEnter＂ & Input event：The user drags the mouse pointer to the element． \\
\hline ＂OnMouseLeave＂ & Input event：The user drags the mouse pointer away from the element． \\
\hline ＂OnMouseMove＂ & \begin{tabular}{l} 
Input event：The user moves the mouse pointer over the element area． \\
－The user releases the mouse button within the element area．It is irrelevant \\
whether the user has previously pressed the mouse button inside or outside \\
the element area． \\
－The user presses the mouse button within the element area，leaves the \\
element area，and then releases the mouse button．
\end{tabular} \\
\hline ＂OnMouseUp＂ & \begin{tabular}{l} 
Note：This CODESYS－specific triggering behavior guarantees that actions for \\
key elements are completed．A key element starts an action for＂OnMouseDown＂ \\
and ends the action for＂OnMouseUp＂． \\
Example：A visualization user presses the mouse button within the element area \\
of the key element and then moves the cursor position so that it lies outside the \\
element area．The action is ended anyway because＂OnMouseUp＂is triggered．
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline ＂Tap＂ & \begin{tabular}{l} 
When a mouse click event occurs，the variable defined in＂Variable＂is described \\
in the application．The coding depends on the＂Tap FALSE＂and＂Tap on enter if \\
captured＂options．
\end{tabular} \\
\hline ＂Variable＂ & \begin{tabular}{l} 
Variable（BOOL）that is set on mouse click event． \\
Example：PLC＿PRG．bIsTapped \\
TRUE：A mouse click event exists．It lasts as long as the user presses the mouse \\
button over the element．It ends when the button is released． \\
FALSE：A mouse click event does not exist． \\
Requirement：The＂Tap FALSE＂option is not activated．
\end{tabular} \\
\hline ＂Tap FALSE＂ & \begin{tabular}{l} 
⿴囗 ：The mouse click event leads to a complementary value in＂Variable＂． \\
TRUE：A mouse click event does not exist． \\
FALSE：While the mouse click event exists．
\end{tabular} \\
\hline ＂Tap on enter if captured＂ & \begin{tabular}{l} 
⿴囗：During user input，it is also taken into consideration whether the mouse \\
pointer is dragged within the element area or not while the mouse button is \\
pressed． \\
TRUE：While the mouse click event exists and the mouse pointer is moved over \\
the element area． \\
FALSE：A mouse click event does not exist．Or the user moves the mouse \\
pointer outside of the element area while the mouse button is pressed． \\
The value is TRUE again as soon as the user moves the pointer back to the \\
element area．The mouse is then captured．
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Toggle" & \begin{tabular}{l} 
With the onset of a mouse click event, the variable is set; when the mouse click \\
event is completed, the variable is reset.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (BOoL). Its value toggled when the mouse click event is ended. This is \\
when the user releases the mouse button while the mouse pointer is over the \\
element area. \\
If the user releases the mouse button while the mouse pointer is outside of the \\
element area, then the mouse click event is not ended and the value is not \\
toggled. \\
Hint: The user can cancel a started toggle input by dragging the mouse pointer \\
out of the element area.
\end{tabular} \\
\hline "Toggle on up if captured" & \begin{tabular}{l} 
V: The value toggles regardless of where the mouse pointer is when the mouse \\
button is released. The mouse is then captured.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Hotkey" & \begin{tabular}{l}
Keyboard shortcut on the element for triggering specific input actions. \\
When the keyboard shortcut event occurs, the input actions in the "Events" property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.
\end{tabular} \\
\hline "Key" & \begin{tabular}{l}
Key pressed for input action. \\
Example: [T] \\
Note: The following properties appear when a key is selected.
\end{tabular} \\
\hline "Events" & \begin{tabular}{l}
- "None" \\
- "Mouse down": Pressing the key triggers the input actions that are configured in the "OnMouseDown" property. \\
- "Mouse up": Releasing the key triggers the input actions that are configured in the "OnMouseUp" property. \\
- "Mouse down/up": Pressing and releasing the key triggers the input actions that are configured in the "OnMouseDown" property and the "OnMouseUp" property.
\end{tabular} \\
\hline "Shift" & Combination with the Shift key Example: [Shift]+[T]. \\
\hline "Control" & Combination with the Ctrl key Example: [Ctrl]+[T]. \\
\hline "Alt" & Combination with the Alt key Example: [AIt]+[T]. \\
\hline
\end{tabular}


All keyboard shortcuts and their actions that are configured in the visualization are listed on the "Keyboard Configuration" tab.

See also
- « Chapter 6.4.5.21.2.2 "Command 'Keyboard Configuration"' on page 3341
- © Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration"' on page 3370

Element property 'Access
rights'

Requirement: User management is set up for the visualization.
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages: \\
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
\\
- "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

\section*{See also}
- \(\sum^{*}\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

\section*{Visualization Element 'Scroll Bar'}

Symbol:


\section*{Category: "Common Controls"}

The element sets the value of a variable, depending on the position of the scroll bar.

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & Example: Speed Conveyor Belt 1 \\
Optional \\
& \begin{tabular}{l} 
Hint: Assign individual names for elements so that they are found faster in the \\
element list.
\end{tabular} \\
\hline "Type of element" & "Scroll Bar" \\
\hline
\end{tabular}

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center' rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline "Y" & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus}\) ) to other positions in the editor.
\begin{tabular}{|l|l|}
\hline "Value" & Variable as type integer that includes the position of the scroll bar. \\
\hline "Minimum value" & Smallest value of the scroll bar (fixed value or variable). \\
\hline "Maximum value" & Largest value of the scroll bar (fixed value or variable). \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Page size" & Page size \\
& - As a fixed value, for example 10 \\
& - As a variable of data type integer \\
Requirement: Visible when the "Move to click" property is not selected. \\
\hline "Move to click" & \begin{tabular}{l} 
Behavior of the scroll bar at visualization runtime when it is clicked: \\
\\
\\
\\
\\
\\
\(\square\) \\
\(\square:\) : The scrollbar moves to the clicked position.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The position defines the location and size of the element in the visualization window. These erty 'Position' are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X\) " & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (吊) to other positions in the editor.

\section*{See also}
- \({ }^{4}\) Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The property defines the representation of scaling and direction of travel. erty 'Bar'
\begin{tabular}{|l|l|}
\hline "Orientation" & \begin{tabular}{l} 
Alignment of the slider; defined by the ratio of width to height. \\
- "Horizontal" \\
- "Vertical" \\
You can modify the alignment in the visualization editor by using the pointing \\
device to adjust the width and height of the Scroll Bar.
\end{tabular} \\
\hline "Running direction" & \begin{tabular}{l} 
The drop-down list varies depending on the alignment of the slider. \\
Horizontal \\
- "Left to right": Scale starts at the left. \\
- "Right to left": Scale starts at the right. \\
Vertical \\
- "Bottom to top": Scale starts at the bottom. \\
- "Top to bottom": Scale starts at the top.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain fixed values for setting colors.
erty 'Colors'
\begin{tabular}{|l|l|}
\hline "Color" & \begin{tabular}{l} 
Color for the element in its normal state. \\
Please note that the normal state is in effect if the expression in the \\
"Color variables \(\rightarrow\) Toggle color" property is not defined or it has the value \\
FALSE.
\end{tabular} \\
\hline "Alarm color" & \begin{tabular}{l} 
Color for the element in alarm state. \\
Please note that the alarm state is in effect if the expression in the \\
"Color variables \(\rightarrow\) Toggle color" property has the value TRUE.
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l} 
Value (0 to 255\()\) for defining the transparency of the selected color. \\
Example \(255:\) The color is opaque. \(0:\) The color is completely transparent.
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\Leftrightarrow_{2}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

Element prop- The properties contains character strings for labeling the element. The character string can also erty 'Texts' contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.
CODESYS accepts the specified texts automatically into the "GlobalTextList" text list. Therefore, these texts can be localized.
\begin{tabular}{|l|l|}
\hline "Text" & \begin{tabular}{l} 
Character string (without single straight quotation marks) for the labeling the \\
element. Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter]. \\
Example: Accesses: \%i \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Text".
\end{tabular} \\
\hline "Tooltip" & \begin{tabular}{l} 
Character string (without single straight quotation marks) that is displayed as the \\
tooltip of an element. \\
Example: Number of valid accesses. \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Tooltip".
\end{tabular} \\
\hline
\end{tabular}

See also
- «3 "Element property 'Text variables"' on page 3128
- \& Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872
- 4 Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329

Element prop- The properties contain fixed values for the text properties. erty 'Text properties'
\begin{tabular}{|c|c|}
\hline "Horizontal alignment" & Horizontal alignment of the text within the element. \\
\hline "Vertical alignment" & Vertical alignment of the text within the element. \\
\hline "Font" & Example: "Default"
The "Font" dialog box opens.
Drop-down list with style fonts. \\
\hline "Font color" & \begin{tabular}{l}
Example: "Black"
: The "Color" dialog box opens. \\
- : Drop-down list with style colors.
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l}
Whole number (value range from 0 to 255). This determines the transparency of the respective color. \\
Example: 255: The color is opaque. \\
0 : The color is completely transparent. \\
Please note: If the color is a style color and already has a transparency value, then this property is write-protected.
\end{tabular} \\
\hline
\end{tabular}

Element prop- These properties are variables with contents that replace a format definition. erty 'Text variables'
\begin{tabular}{|l|l|}
\hline "Text variable" & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses
\end{tabular} \\
& \begin{tabular}{l} 
Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Text". \\
Note: If you specify a variable of type enumeration with text list support, then \\
the name of the enumeration data type is added automatically in angle brackets \\
after the variable name. Example: PLC PRG. enVar <enume ration name>. \\
Then the symbolic value of the enumeration component is printed instead of the \\
numeric value when text is printed. Refer to the help page for the enumerations.
\end{tabular} \\
\hline "Tooltip variable" & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses InTooltip
\end{tabular} \\
Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Tooltip".
\end{tabular}.

See also
- \& Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329
- © "Element property 'Texts"' on page 3127
- © Chapter 6.4.1.20.5.18 "Enumerations" on page 2263

Element property 'Dynamic texts'

Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is currently indexed in the variable.
\begin{tabular}{|c|c|}
\hline "Text list" & \begin{tabular}{l}
Variable (string) or name of the text list as a fixed string in single straight quotation marks. \\
Example: 'Errorlist' \\
: Drop-down list with the dialogs available in the text lists.
\end{tabular} \\
\hline "Text index" & \begin{tabular}{l}
Text list ID. This refers to the desired output text. \\
- As fixed string with the ID in single straight quotation marks. \\
Example: '1' \\
- As a variable (STRING) for dynamically controlling the text output. \\
Example: strTextID \\
Sample assignment: PLC_PRG.strTextID := '1';
\end{tabular} \\
\hline "Tooltip index" & \begin{tabular}{l}
Text list ID. This refers to the desired output text. \\
- As fixed string with the ID in single straight quotation marks. \\
Example: '2' \\
- As a variable (STRING) for dynamically controlling the text output. \\
Example: strToolTipID \\
Sample assignment: PLC_PRG.strToolTipID := '2';
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{\text {• }}\), Chapter 6.4.1.21.2.28 "Object 'Text List"' on page 2532

Element prop- The variables allow for dynamic control of the text display. erty 'Font variables'
\begin{tabular}{|c|c|}
\hline "Font name" & \begin{tabular}{l}
Variable (STRING). Includes the font of the text. \\
Example: PLC_PRG.stFontVar := 'Arial'; \\
The selection of fonts corresponds to the default "Font" dialog.
\end{tabular} \\
\hline "Size" & \begin{tabular}{l}
Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name. \\
- <pt>: Points (default) \\
Example: PLC_PRG.iFontHeight <pt> \\
Code: iFontHeight : INT := 12; \\
- <px> : Pixels \\
Example: PLC_PRG.iFontHeight <px> \\
Code: iFontHeight : INT := 19; \\
pt \\
pt \\
px \\
If you click in the value field, a drop-down list opens on the right for setting the unit. \\
Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property "Text property \(\rightarrow\) Font".
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Flags" & \begin{tabular}{l}
Variable (DWORD). Contains the flags for displaying fonts. \\
Flags: \\
- 1: Italics \\
- 2: Bold \\
- 4: Underline \\
- 8: Strikethrough \\
Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: PLC_PRG.dwFontType := 6;
\end{tabular} \\
\hline "Character set" & \begin{tabular}{l}
Variable (DWORD). Contains a character set number for the font. \\
The selection of character set numbers corresponds to the "Script" setting of the standard "Font" dialog.
\end{tabular} \\
\hline "Color" & \begin{tabular}{l}
Variable (DWORD). Includes the color of the text. \\
Example: PLC_PRG.dwColorFont:= 16\#FF000000;
\end{tabular} \\
\hline "Flags for text alignment" & \begin{tabular}{l}
Variable (integer data type). Contains the coding for text alignment. \\
Example: PLC_PRG.dwTextAlignment. \\
Coding: \\
- 0 : Top left \\
- 1: Horizontal center \\
- 2: Right \\
- 4: Vertical center \\
- 8: Bottom \\
Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: PLC_PRG.dwFontType := 5;
\end{tabular} \\
\hline
\end{tabular}

Fixed values for displaying texts are set in "Text properties".

See also
- « "Element property 'Text properties'" on page 3116

Element prop- The Element property is used as an interface for project variables to dynamically control colors erty 'Color varia- at runtime. bles'
\begin{tabular}{|c|c|}
\hline "Toggle color" & \begin{tabular}{l}
The property controls the toggled color at runtime. \\
Value assignment: \\
- FALSE: The element is displayed with the color specified in the "Color" property. \\
- true: The element is displayed with the color specified in the "Alarm color" property. \\
Assignment options: \\
- Placeholder for the user input variable \\
- "<toggle/tap variable>" \\
- "<NOT toggle/tap variable>" \\
The color change is not controlled by its own variable, but by a user input variable. \\
Note: Specify a variable for the mouse events "Tap" or "Toggle" in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both "Toggle" and "Tap", then the variable specified in "Tap" is used. \\
Hint: Click the symbol to insert the placeholder "<toggle/tap variable>". When you activate the "Inputconfiguration", "Tap FALSE" property, then the "<NOT toggle/tap variable>" placeholder is displayed. \\
- Instance path of a project variable (BOOL) \\
Example: PLC_PRG.xColorIsToggeled \\
Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.
\end{tabular} \\
\hline \begin{tabular}{l}
"Normal state" \\
"Alarm state"
\end{tabular} & The properties listed below control the color depending on the state. The normal state is in effect if the variable in "Color variables", "Toggle color" is not defined or it has the value FALSE. The alarm state is in effect if the variable in "Colorvariables", "Toggle color" has the value tRUE. \\
\hline "Frame color" & \begin{tabular}{l}
Assignment options: \\
- Variable (DWORD) for the frame color \\
Example: PLC_PRG.dwBorderColor \\
- Color literal Example of green and opaque: 16\#FF00FF00
\end{tabular} \\
\hline "Filling color" & \begin{tabular}{l}
Assignment options: \\
- Variable (DWORD) for the fill color \\
Example: PLC_PRG.dwFillColor \\
- Color literal \\
Example of gray and opaque: 16\#FF888888
\end{tabular} \\
\hline
\end{tabular}

The transparency part of the color value is evaluated only if the "Activate semitransparent drawing" option of the visualization manager is selected.

Select the "Advanced" option in the toolbar of the properties view. Then all element properties are visible.

\section*{See also}
- \& Chapter 6.4.5.10.4 "Animating a color display" on page 2914

\section*{Element property 'Absolute movement'}

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\boldsymbol{\oplus}\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
\hline "Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol. \\
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

The " \(X\) ", " \(Y\) ", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

\section*{See also}

Element prop- The variables control the element behavior dynamically. erty 'State variables'
\begin{tabular}{|l|l|}
\hline "Invisible"" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \(=\) FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

\footnotetext{
Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'
}
\begin{tabular}{|c|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages: \\
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
- "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline See also
\end{tabular}

See also
- \({ }^{\#}\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights"' on page 3366

\section*{See also}
- \(\Leftrightarrow\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Slider'}

\section*{Symbol:}

\section*{Category: "Common Controls"}

The element changes the value of a variable, depending on the position of the slider within the slider bar. You define the value range of the slider bar by means of the scale start and scale end.

\section*{Element proper- \\ ties}
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Example: Speed controller conveyor belt 1 \\
Optional \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list.
\end{tabular} \\
\hline "Type of element" & "Slider" \\
\hline
\end{tabular}

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline "X" & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols ( ( \(^{-}\)) to other positions in the editor.

See also
- Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center'
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline " \(Y\) " & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus})\) to other positions in the editor.
\begin{tabular}{|c|c|}
\hline "Variable" & \begin{tabular}{l}
Variable (numeric data type) \\
Example: PLC_PRG.rSlider \\
When executed, the variable assigns a value that corresponds to the position of the slider in the bar.
\end{tabular} \\
\hline "Page size" & \begin{tabular}{l}
Page size \\
- As a fixed value, for example 10 \\
- As an IEC variable of data type integer \\
Requirement: The "Move to click" element property is not selected.
\end{tabular} \\
\hline "Move to click" & Behavior of the slider at visualization runtime when it is clicked:
The slider moves to the clicked position.
\(\square\) The slider moves to the value (defined in the "Page size" element property) in the direction of the click. \\
\hline
\end{tabular}

\section*{Element prop-}
erty 'Scale'
\begin{tabular}{|l|l|}
\hline "Show scale" & \begin{tabular}{l} 
: The element has a visible scale. \\
Note: This option is available for the "Slider" only.
\end{tabular} \\
\hline "Scale start" & \begin{tabular}{l} 
Least value of the scale and the lower limit of the value range for the element. \\
Example: 0 \\
0 The property "Variable" is shown below.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the scale start. \\
Example: PLC_PRG.iScaleStart \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iScaleStart : INT := 0;
END_VAR

```
\end{tabular} \\
\hline "Scale end" & \begin{tabular}{l}
Greatest value of the scale and the upper limit of the value range for the element. \\
Example: 100 \\
*: The property "Variable" is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the scale end. \\
Example: PLC_PRG.iScaleEnd \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iScaleEnd : INT := 120;
END_VAR

```
\end{tabular} \\
\hline "Main scale" & \begin{tabular}{l}
Distance between two tick marks on the rough scale. \\
Example: 10 \\
*: The property "Variable" is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the distance. \\
Example: PLC_PRG.iMainScale \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iMainScale : INT := 20;
END_VAR

```
\end{tabular} \\
\hline "Subscale" & \begin{tabular}{l}
Distance between two dashes on the fine scale. You can hide the fine scale by setting the value to 0 . \\
Example: 2 \\
*: The property "Variable" is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the distance. \\
Example: PLC_PRG.iSubScale \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iMainScale : INT := 5;
END_VAR

```
\end{tabular} \\
\hline "Scale format (C Syntax)" & \begin{tabular}{l}
Formatting of the scale label (example: \%d \%s) \\
Note: This property is available for the Slider only.
\end{tabular} \\
\hline "Scale proportion" & Size of the scale (in \%) of the total size \\
\hline
\end{tabular}

Element prop- The property defines the representation of scaling and direction of travel. erty 'Bar'
\begin{tabular}{|l|l|}
\hline "Diagram type" & \begin{tabular}{l} 
The drop-down list varies depending on the alignment of the diagram. \\
Horizontal \\
- "Top": Scale is above the slider. \\
- \\
- "Bottom": Scale is below the slider. \\
- "Top and bottom": Two scales frame the slider above and below. \\
Vertical \\
- Left: Scale is left of the slider. \\
- Right: Scale is right of the slider. \\
- Left and right: Two scales frame the slider on the left and the right.
\end{tabular} \\
\hline "Orientation" \begin{tabular}{l} 
Alignment of the slider; defined by the ratio of width to height. \\
- "Horizontal" \\
- "Vertical" \\
You can modify the alignment in the visualization editor by using the pointing \\
device to adjust the width and height of the scrollbar.
\end{tabular} \\
\hline \begin{tabular}{l} 
The drop-down list varies depending on the alignment of the slider. \\
Horizontal \\
- "Left to right": Scale starts at the left. \\
- "Right to left": Scale starts at the right. \\
Vertical \\
- "Bottom to top": Scale starts at the bottom. \\
- "Top to bottom": Scale starts at the top.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline " \(" \mathrm{Y}\) " & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG. iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\oplus\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
\hline "Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol.
\end{tabular} \\
\begin{tabular}{l} 
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

The " \(X\) ", " "", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

\section*{Element prop- The variables control the element behavior dynamically. erty 'State varia- \\ bles'}
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \(=\) FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

\section*{The "Invisible" property is supported by the "Client Animation" functionality.}

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- Requirement: User management is set up for the visualization. erty 'Access
rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages: \\
\\
\\
\(\bullet\) - "Not set. Full rights.": Access rights for all user groups : "operable"" \\
• "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

See also
- \(y^{y}\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

\section*{See also}
- \(y^{y}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Spin Box'}

Symbol:


\section*{Category: "Common Controls"}

The element increments or decrements the value of a variable in defined intervals.

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Example: Speed controller conveyor belt \\
Optional \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list.
\end{tabular} \\
\hline "Type of element" & "Spin Box" \\
\hline
\end{tabular}

Element prop- The position defines the location and size of the element in the visualization window. These erty 'Position' are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x -axis runs to the right. The positive vertical y -axis runs downwards.
\begin{tabular}{|l|l|}
\hline "X" & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols ( ( - ) to other positions in the editor.

\section*{See also}
- Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline\(Y "\) & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\mathbf{\oplus})\) to other positions in the editor.
\begin{tabular}{|l|l|}
\hline "Variable" & \begin{tabular}{l} 
Variable (numeric data type) \\
Example: PLC_PRG.iTemp
\end{tabular} \\
\hline "Number format" & \begin{tabular}{l} 
Format of the value in printf syntax \\
Example: \%d, \%5.2f
\end{tabular} \\
\hline "Interval" & Interval used for modification of the value \\
\hline
\end{tabular}

Element property 'Value
range'
\begin{tabular}{|l|l|}
\hline "Minimum value" & \begin{tabular}{l} 
Lower limit of the output value \\
- fixed value \\
- Variable (INT)
\end{tabular} \\
\hline "Maximum value" & \begin{tabular}{l} 
Upper limit of the output value \\
- fixed value \\
- Variable (INT)
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain fixed values for the text properties. erty 'Text properties'
\begin{tabular}{|c|c|}
\hline "Usage of" & \begin{tabular}{l}
- "Default style values": The values of the visualization style are used. \\
- "Individual settings": The "Individual text properties" property group is shown The values can be customized here.
\end{tabular} \\
\hline \multicolumn{2}{|l|}{\multirow[t]{2}{*}{\begin{tabular}{l}
"Individual text properties" \\
Requirement: The "Individual settings" text property is defined.
\end{tabular}}} \\
\hline & \\
\hline "Horizontal alignment" & Horizontal alignment of the text within the element. \\
\hline "Vertical alignment" & Vertical alignment of the text within the element. \\
\hline "Font" & \begin{tabular}{l}
Example: "Default"
The "Font" dialog box opens. \\
*: Drop-down list with style fonts.
\end{tabular} \\
\hline "Font color" & \begin{tabular}{l}
Example: "Black"
The "Color" dialog box opens. \\
- : Drop-down list with style colors.
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l}
Whole number (value range from 0 to 255). This determines the transparency of the respective color. \\
Example: 255: The color is opaque. \\
0 : The color is completely transparent. \\
Please note: If the color is a style color and already has a transparency value, then this property is write-protected.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The Element property is used as an interface for project variables to dynamically control colors erty 'Color varia- at runtime. bles'
\begin{tabular}{|c|c|}
\hline "Toggle color" & \begin{tabular}{l}
The property controls the toggled color at runtime. \\
Value assignment: \\
- FALSE: The element is displayed with the color specified in the "Color" property. \\
- TRUE: The element is displayed with the color specified in the "Alarm color" property. \\
Assigning the property: \\
- Placeholder for the user input variable \\
- "<toggle/tap variable>" \\
- "<NOT toggle/tap variable>" \\
The color change is not controlled by its own variable, but by a user input variable. \\
Note: Specify a variable for the mouse events "Tap" or "Toggle" in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both "Toggle" and "Tap", then the variable specified in "Tap" is used. \\
Hint: Click the symbol to insert the placeholder "<toggle/tap variable>". When you activate the "Inputconfiguration", "Tap FALSE" property, then the "<NOT toggle/tap variable>" placeholder is displayed. \\
- Instance path of a project variable (BOOL) \\
Example: PLC_PRG.xColorIsToggeled \\
Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.
\end{tabular} \\
\hline
\end{tabular}

The transparency part of the color value is evaluated only if the "Activate semitransparent drawing" option of the visualization manager is selected.

Select the "Advanced" option in the toolbar of the properties view. Then all element properties are visible.

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" \\
\hline " \(X "\) & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline " \(Y\) " & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG. iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\oplus\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
\hline "Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol. \\
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

The " \(X\) ", " \(Y\) ", "Rotation", and "Interior rotation" properties are supported by the
"Client Animation" functionality.

See also

Element prop- The variables control the element behavior dynamically.
erty 'State varia-
bles'
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \(:=\) FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain the configurations for the user input when using the mouse or keyboard. erty 'Input con- User input is a user event from the perspective of the element. figuration'


The "Configure" button opens the "Input configuration" dialog box for creating or modifying a user input configuration.
A configuration contains one or more input actions for the respective input event. Existing input actions are displayed below it.
\begin{tabular}{|l|l|}
\hline \multicolumn{1}{|l|}{ Example: "Execute ST code": } & PLC_PRG.i_x \(:=0 ;\) \\
\hline "OnDialogClosed" & Input event: The user closes the dialog box. \\
\hline "OnMouseClick" & \begin{tabular}{l} 
Input event: The user clicks the element completely. The mouse button is clicked \\
and released.
\end{tabular} \\
\hline "OnMouseDown" & Input event: The user clicks down on the element only. \\
\hline "OnMouseEnter" & Input event: The user drags the mouse pointer to the element. \\
\hline "OnMouseLeave" & Input event: The user drags the mouse pointer away from the element. \\
\hline "OnMouseMove" & Input event: The user moves the mouse pointer over the element area. \\
\hline "OnMouseUp" & Input event: The user releases the mouse button over the element area. \\
\hline
\end{tabular}

See also
- \({ }^{*}\) ) Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration'" on page 3370
\begin{tabular}{|l|l|}
\hline "Tap" & \begin{tabular}{l} 
When a mouse click event occurs, the variable defined in "Variable" is described \\
in the application. The coding depends on the options "Tap FALSE" and "Tap on \\
enter if captured".
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL). Contains the information whether a mouse click event exists. \\
Example: PLC_PRG. bIsTapped \\
TRUE: A mouse click event exists. It lasts while the user presses the mouse \\
button over the element. It ends when the button is released. \\
FALSE: A mouse click event does not exist. \\
Requirement: The "Tap FALSE" option is not activated.
\end{tabular} \\
\hline "Tap FALSE" & \begin{tabular}{l} 
V: The mouse click event leads to a complementary value in "Variable". \\
TRUE: A mouse click event does not exist.
\end{tabular} \\
\hline FALSE: While the mouse click event exists.
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Shift" & \begin{tabular}{l} 
When a mouse click event occurs, the variable here is described in the applica- \\
tion. When the mouse click event ends, its value is toggled with the "Toggle on \\
up if captured" option.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL). Its value toggled when the mouse click event is ended. This is \\
when the user releases the mouse button while the mouse pointer is over the \\
element area. \\
If the user releases the mouse button while the mouse pointer is outside of the \\
element area, then the mouse click event is not ended and the value is not \\
toggled. \\
Tip: The user can cancel a started toggle input by dragging the mouse pointer \\
out of the element area.
\end{tabular} \\
\hline "Toggle on up if captured" & \begin{tabular}{l} 
V: The value toggles regardless of where the mouse pointer is when the mouse \\
button is released. The mouse is then captured.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Hotkeys" & \begin{tabular}{l}
Keyboard shortcut on the element for triggering specific input actions. \\
When the keyboard shortcut event occurs, the input actions in the "Event(s)" property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.
\end{tabular} \\
\hline "Key" & \begin{tabular}{l}
Key pressed for input action. \\
Example: [T] \\
Note: The following properties appear when a key is selected.
\end{tabular} \\
\hline "Event(s)" & \begin{tabular}{l}
- "None" \\
- "Mouse down": Pressing the key triggers the input actions that are configured in the "OnMouseDown" property. \\
- "Mouse up": Releasing the key triggers the input actions that are configured in the "OnMouseUp" property. \\
- "Mouse down/up": Pressing and releasing the key triggers the input actions that are configured in the "OnMouseDown" property and the "OnMouseUp" property.
\end{tabular} \\
\hline "Shift" & Combination with the Shift key Example: [Shift]+[T]. \\
\hline "Control" & Combination with the Ctrl key Example: [Ctrl]+[T]. \\
\hline "Alt" & Combination with the Alt key Example: [AIt]+[T]. \\
\hline
\end{tabular}

All keyboard shortcuts and their actions that are configured in the visualization are listed in the "Keyboard configuration" tab.

\section*{See also}
- \({ }^{4}\) Chapter 6.4.5.21.2.2 "Command 'Keyboard Configuration'" on page 3341
- \(y^{*}\) Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration'" on page 3370

\footnotetext{
Element prop- Requirement: User management is set up for the visualization.
erty 'Access
rights'
}
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages: \\
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
- "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

\section*{See also}
- \({ }^{\#}\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights"" on page 3366

\section*{Visualization Element 'Invisible Input'}

\section*{Symbol:}


\section*{Category: "Common Controls"}

This element is displayed in the editor with a dashed line which is not visible in online mode. You define the behavior of the el in the input configuration.

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Optional \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list. \\
Example: Unsichtbare_Eingabe_1
\end{tabular} \\
\hline "Type of element" & "Invisible Input" \\
\hline
\end{tabular}

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x -axis runs to the right. The positive vertical y -axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X "\) & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}


See also
- H Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center'
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline "Y" & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|c|c|c|}
\hline \multicolumn{3}{|l|}{"Movement"} \\
\hline "X" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular}} \\
\hline "Y" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular}} \\
\hline "Rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle1. \\
The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol. \\
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
\end{tabular} &  \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol.
\end{tabular} \\
\begin{tabular}{l} 
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

The "X", " \(Y\) ", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also

Element prop- The variables control the element behavior dynamically. erty 'State varia-
bles'
\begin{tabular}{|l|l|}
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain the configurations for the user input when using the mouse or keyboard. erty 'Input con- A user input defines an event and one or more actions that are executed when an event occurs. figuration'

The "Configure" button opens the "Input Configuration" dialog. There you can create or edit user inputs.
Configured user inputs are listed below the events. They each include the action that is triggered and the setting in short form.

Example: "Execute ST Code": ßPLC_PRG.i_x := 0;
\begin{tabular}{|l|l|}
\hline "OnDialogClosed" & Input event: The user closes the dialog. \\
\hline "OnMouseClick" & \begin{tabular}{l} 
Input event: The user clicks the mouse button completely in the element area. \\
The mouse button is clicked and released.
\end{tabular} \\
\hline "OnMouseDown" & Input event: The user clicks down on the mouse button. \\
\hline "OnMouseEnter" & Input event: The user drags the mouse pointer to the element. \\
\hline "OnMouseLeave" & Input event: The user drags the mouse pointer away from the element. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "OnMouseMove" & Input event: The user moves the mouse pointer over the element area. \\
\hline "OnMouseUp" & \begin{tabular}{l} 
Input events: \\
- \begin{tabular}{l} 
The user releases the mouse button within the element area. It is irrelevant \\
whether the user has previously pressed the mouse button inside or outside \\
the element area.
\end{tabular} \\
- The user presses the mouse button within the element area, leaves the \\
element area, and then releases the mouse button. \\
Note: This CODESYS-specific triggering behavior guarantees that actions for \\
key elements are completed. A key element starts an action for "OnMouseDown"" \\
and ends the action for "OnMouseUp". \\
Example: A visualization user presses the mouse button within the element area \\
of the key element and then moves the cursor position so that it lies outside the \\
element area. The action is ended anyway because "OnMouseUp" is triggered.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Tap" & \begin{tabular}{l} 
When a mouse click event occurs, the variable defined in "Variable" is described \\
in the application. The coding depends on the "Tap FALSE" and "Tap on enter if \\
captured" options.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL) that is set on mouse click event. \\
Example: PLC_PRG . bI sTapped \\
TRUE: A mouse click event exists. It lasts as long as the user presses the mouse \\
button over the element. It ends when the button is released.
\end{tabular} \\
& \begin{tabular}{l} 
FALSE: A mouse click event does not exist. \\
Requirement: The "Tap FALSE" option is not activated.
\end{tabular} \\
\hline "Tap FALSE" & \begin{tabular}{l} 
V: The mouse click event leads to a complementary value in "Variable". \\
TRUE: A mouse click event does not exist.
\end{tabular} \\
\hline FALSE: While the mouse click event exists.
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Toggle" & \begin{tabular}{l} 
With the onset of a mouse click event, the variable is set; when the mouse click \\
event is completed, the variable is reset.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL). Its value toggled when the mouse click event is ended. This is \\
when the user releases the mouse button while the mouse pointer is over the \\
element area. \\
If the user releases the mouse button while the mouse pointer is outside of the \\
element area, then the mouse click event is not ended and the value is not \\
toggled. \\
Hint: The user can cancel a started toggle input by dragging the mouse pointer \\
out of the element area.
\end{tabular} \\
\hline "Toggle on up if captured" & \begin{tabular}{l} 
V: The value toggles regardless of where the mouse pointer is when the mouse \\
button is released. The mouse is then captured.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Hotkey" & \begin{tabular}{l}
Keyboard shortcut on the element for triggering specific input actions. \\
When the keyboard shortcut event occurs, the input actions in the "Events" property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.
\end{tabular} \\
\hline "Key" & \begin{tabular}{l}
Key pressed for input action. \\
Example: [T] \\
Note: The following properties appear when a key is selected.
\end{tabular} \\
\hline "Events" & \begin{tabular}{l}
- "None" \\
- "Mouse down": Pressing the key triggers the input actions that are configured in the "OnMouseDown" property. \\
- "Mouse up": Releasing the key triggers the input actions that are configured in the "OnMouseUp" property. \\
- "Mouse down/up": Pressing and releasing the key triggers the input actions that are configured in the "OnMouseDown" property and the "OnMouseUp" property.
\end{tabular} \\
\hline "Shift" & Combination with the Shift key Example: [Shift]+[T]. \\
\hline "Control" & Combination with the Ctrl key Example: [Ctrl]+[T]. \\
\hline "Alt" & Combination with the Alt key Example: \([A I t]+[T]\). \\
\hline
\end{tabular}

All keyboard shortcuts and their actions that are configured in the visualization are listed on the "Keyboard Configuration" tab.

\section*{See also}
- ⓨ Chapter 6.4.5.21.2.2 "Command 'Keyboard Configuration'" on page 3341
- \(\xi^{2}\) Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration'" on page 3370

Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages: \\
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
- "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\(\qquad\)\begin{tabular}{l} 
See also \\
- «3 Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366
\end{tabular}
\end{tabular}

See also
- \({ }^{4}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Progress Bar'}

Symbol:

Category: "Common Controls"
The element displays the value of a variable as a progress bar.

Element proper-
ties
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Optional \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list. \\
Example: Progress_Data_Transfer
\end{tabular} \\
\hline "Progress Bar" \\
\hline "Type of element" ID" & \begin{tabular}{l} 
ID of the global text list \\
Requirement: Text is configured in the property "Texts \(\rightarrow\) Text".
\end{tabular} \\
\hline "Variable" & Variable (numeric data type). Represents the length of the progress bar. \\
\hline "Minimum value" & Value range of the variable \\
\hline "Maximum value" & \begin{tabular}{l} 
- "Blocks" \\
- "Bar"
\end{tabular} \\
\hline
\end{tabular}

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X\) " & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (ए-) to other positions in the editor.

See also
- \(\boldsymbol{y}^{\boldsymbol{y}}\) Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline "Y" & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus}\) ) to other positions in the editor.

Element prop-
erty 'Texts'
\begin{tabular}{|l|l|}
\hline "Text" & \begin{tabular}{l} 
String label for the element. \\
Example: Zoom
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain IEC variables for controlling the position of the element dynamically. The erty 'Absolute movement' reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" & \multicolumn{1}{l|}{\begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG. iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular}} \\
\hline "Y" & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\boldsymbol{\oplus}\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol.
\end{tabular} \\
\begin{tabular}{l} 
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

The "X", " \(Y\) ", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also

Element prop- The variables control the element behavior dynamically.
erty 'State varia-
bles'
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- Requirement: User management is set up for the visualization. erty 'Access
rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages:
\end{tabular} \\
& \begin{tabular}{l} 
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
\(\bullet\) \\
"Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{\wedge} \Rightarrow\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

See also
- *y Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Check Box'}

Symbol:


\section*{Category: "Common Controls"}

The element is used for setting and resetting a Boolean variable. The set state is identified by a check mark.

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Example: signal_tone_for_parts_deficit \\
Optional \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list.
\end{tabular} \\
\hline "Type of element" & "Check Box" \\
\hline "Text ID" & \begin{tabular}{l} 
ID for the text in the "GlobalTextList" \\
Example: 22 \\
The text ID cannot be changed. As soon as you specify and save a text in \\
"Texts"- "Text", CODESYS automatically creates an entry in the "GlobalTextList" \\
and displays the ID here.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline "X" & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (古) to other positions in the editor.

\section*{See also}
- \({ }^{3}\), Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center'
\begin{tabular}{|l|l|}
\hline\(" X "\) & X-coordinate of the point of rotation \\
\hline\(" Y "\) & \(Y\)-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\mathbf{\oplus})\) to other positions in the editor.
\begin{tabular}{|l|l|}
\hline "Variable" & \begin{tabular}{l} 
Variable of type BOOL \\
Example: "PLC_PRG.xlsTrue"
\end{tabular} \\
\hline "Frame size" & \begin{tabular}{l} 
Distance of the element to the edge \\
Example: "From style"
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contains character strings for labeling the element. erty 'Texts'

CODESYS accepts the specified texts automatically into the "GlobalTextList" text list. Therefore, these texts can be localized.
\begin{tabular}{|l|l|}
\hline "Text" & \begin{tabular}{l} 
Character string (without single straight quotation marks) for the labeling the \\
element. \\
Example: Axis 1.
\end{tabular} \\
\hline "Tooltip" & \begin{tabular}{l} 
Character string (without single straight quotation marks) that is displayed as the \\
tooltip of an element. \\
Example: Parameters of Axis 1.
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{\sharp}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

Element prop- The properties contain fixed values for the text properties.
erty 'Text properties'
\begin{tabular}{|l|l|}
\hline "Usage of" & \begin{tabular}{l} 
- "Default style values": The values of the visualization style are used. \\
- "Indiviviual settings": The "Individual text properties" property group is shown \\
The values can be customized here.
\end{tabular} \\
\hline \begin{tabular}{l} 
"Individual text properties" \\
Requirement: The "Individual settings" text property is defined.
\end{tabular} \\
\hline "Horizontal alignment" & Horizontal alignment of the text within the element. \\
\hline "Vertical alignment" & Vertical alignment of the text within the element. \\
\hline "Text format" & \begin{tabular}{l} 
Definition for displaying texts that are too long \\
- "Default": The long text is truncated. \\
- "Line brek": The text is split into parts. \\
- "Ellipsis": The visible text ends with "..." indicating that it is not complete.
\end{tabular} \\
\hline & \begin{tabular}{l} 
Example: "Default" \\
"..: The "Font" dialog box opens. \\
v: Drop-down list with style fonts.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline \multirow[t]{3}{*}{"Font color"} & Example: "Black" \\
\hline & ...): The "Color" dialog box opens. \\
\hline & - Drop-down list with style colors. \\
\hline \multirow[t]{4}{*}{"Transparency"} & Whole number (value range from 0 to 255). This determines the transparency of the respective color. \\
\hline & Example: 255: The color is opaque. \\
\hline & 0 : The color is completely transparent. \\
\hline & Please note: If the color is a style color and already has a transparency value, then this property is write-protected. \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\oplus\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
\hline "Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol. \\
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
\hline
\end{tabular}

You can link the variables to a unit conversion.
"X", "Y", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

\section*{See also}

Element prop- The variables control the element behavior dynamically. erty 'State variables'
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \(=\) FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|l|l|}
\hline "Animation duration" & \begin{tabular}{l} 
Defines the duration (in milliseconds) in which the element runs an animation \\
- \begin{tabular}{l} 
Variable (integer value) \\
Example: Menu. tContent with VAR tContent : INT \(:=500 ;\) \\
END_VAR \\
- Integer literal \\
Example: 500
\end{tabular} \\
Animatable properties \\
- "Absolute movement", "Movement", "X", " \(Y\) " \\
- "Absolute movement", ""otation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable \\
property has changed. The movement then executed is not jerky, but is smooth \\
within the specified animation duration. The visualization element travels to the \\
specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline \begin{tabular}{l} 
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIs InForeground with VAR bIsInForeground : BooL \(:=\) \\
FALSE; END_VAR
\end{tabular} \\
\hline "Move to foreground" \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it \\
was inserted in the visualization editor.
\end{tabular}

Element prop- Requirement: User management is set up for the visualization. erty 'Access
rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages:
\end{tabular} \\
& \begin{tabular}{l} 
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
\(\bullet\) \\
"Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

See also
- ※y Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

See also
- \(\Leftrightarrow_{2}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Radio Buttons'}

Symbol:


\section*{Category: "Common Controls"}

The element provides a series of radios buttons with an unlimited number of options.

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Optional \\
\\
\\
Hint: Assign individual names for elements so that they are found faster in the \\
element list. \\
Example: Morning Shift
\end{tabular} \\
\hline "Type of element" & "Radio Buttons" \\
\hline
\end{tabular}

Element prop- The position defines the location and size of the element in the visualization window. These erty 'Position' are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X\) " & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (區) to other positions in the editor.

See also
- \({ }^{\text {• }}\) Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874
\begin{tabular}{|l|l|}
\hline "Variable" & \begin{tabular}{l} 
Variable (integer data type) that gives the index of the radio button that the \\
visualization user has activated \\
Example: PLC_PRG.iNrOfActivatedRadioButton
\end{tabular} \\
\hline "Number of columns" & \begin{tabular}{l} 
Definition of the number of list boxes displayed in a row \\
Example: 2
\end{tabular} \\
\hline "Radio button order" & \begin{tabular}{l} 
"Left to right": The radio buttons are aligned by rows until the number of columns \\
is reached. \\
"Top to bottom": The radio buttons are aligned row by columns until the number \\
of columns is reached.
\end{tabular} \\
\hline "Frame size" & Defines the distance from the list boxes to the edge (in pixels). \\
\hline "Row height" & \begin{tabular}{l} 
Height of the row (in pixels) Modifying the height of the row also changes the \\
size of the list box.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain fixed values for the text properties.
erty 'Text prop-
erties'
\begin{tabular}{|l|l|}
\hline "Usage of" & \begin{tabular}{l} 
- "Default style values": The values of the visualization style are used. \\
- "Individual settings": The "Individual text properties" property group is shown \\
The values can be customized here.
\end{tabular} \\
\hline \begin{tabular}{l} 
"Individual text properties" \\
Requirement: The "Individual settings" text property is defined.
\end{tabular} \\
\hline "Horizontal alignment" & Horizontal alignment of the text within the element. \\
\hline "Vertical alignment" & \begin{tabular}{l} 
Vertical alignment of the text within the element.
\end{tabular} \\
\hline "Text format" & \begin{tabular}{l} 
Definition for displaying texts that are too long \\
- "Default": The long text is truncated. \\
- "Line brek": The text is split into parts. \\
- "Ellipsis": The visible text ends with "..." indicating that it is not complete.
\end{tabular} \\
\hline "Font" & \begin{tabular}{l} 
Example: "Default" \\
"..: The "Font" dialog box opens. \\
-: Drop-down list with style fonts.
\end{tabular} \\
\hline & \begin{tabular}{l} 
Example: "Black" \\
"..: The "Color" dialog box opens. \\
- : Drop-down list with style colors.
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l} 
Whole number (value range from 0 to 255). This determines the transparency of \\
the respective color. \\
Example: \(255:\) The color is opaque. \\
0: The color is completely transparent. \\
Please note: If the color is a style color and already has a transparency value, \\
then this property is write-protected.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" & \multicolumn{1}{|l|}{\begin{tabular}{|l|l|}
\hline\(X "\) & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline " \(Y\) " & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG. iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline
\end{tabular}} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\oplus\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
"Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\boldsymbol{\oplus}\) symbol.
\end{tabular} \\
\begin{tabular}{l} 
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular}
\end{tabular}

You can link the variables to a unit conversion.

The " \(X\) ", " "", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

\section*{Element prop- The variables control the element behavior dynamically. erty 'State varia- \\ bles'}
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \(=\) FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

\section*{The "Invisible" property is supported by the "Client Animation" functionality.}
\begin{tabular}{|c|c|c|}
\hline \multirow[t]{5}{*}{Element property 'Radio button settings'} & \begin{tabular}{l}
"Radio button" \\
- "Areas" \\
- " \([<n>]\) "
\end{tabular} & \begin{tabular}{l}
"Create new": Clicking this button creates a new selection button in the editor and lists an additional area in the properties editor. \\
For each radio button, an area is visible that records the settings. \\
- [<n>] \\
- "[<n>]": This number indicates the area. Clicking "Delete" will delete the associated radio button with its settings "Text", "Tooltip", and "Line spacing (in pixels)".
\end{tabular} \\
\hline & Areas: [<n>] & \\
\hline & "Text" & The button name is specified here. Default value: "Radio_button" \\
\hline & "Tooltip" & Text is specified here that is displayed in a tooltip. \\
\hline & "Line spacing (in pixels)" & The distance (in pixels) to the upper button can be specified here. \\
\hline
\end{tabular}

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

\footnotetext{
Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'
}
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages:
\end{tabular} \\
& \begin{tabular}{l} 
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
\(\bullet \quad\) "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

See also
- \(y^{y}\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

\section*{See also}
- ② Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Alarm Table'}

Symbol:


\section*{Category: "Alarm Manager"}

The element displays alarms in a list. In the element properties, you specify which information is shown. You define the appearance of the element and the variables that control the element behavior.

In online mode, you can sort an alarm table by a specific column - even in the classic view. Click into the column header. A small triangle indicates the current sort order (ascending, descending). Clicking the symbol reverses the order.

Sorting inside the column depends on the type of the contained information. The "Priority" column is sorted numerically, and the "Message" and "Class" columns alphabetically. The "Value" and "Latch" columns may contain different value types. In this case, sorting is first by type (blank, Boolean, numeric value, character string) and then either numerically or alphabetically depending on the type.

If an alarm history has been created, then you can programmatically delete it at runtime. The recording starts again from the time of deletion. See the help page for "Visualizing Alarm Management".

\section*{Element proper- \\ ties}
\begin{tabular}{|c|c|}
\hline "Element name" & Example: GenElemInst_1 \\
\hline "Type of element" & "Alarm Table" \\
\hline "Data source" & \begin{tabular}{l}
Selection of the device and the application where the data to be visualized and the alarms are generated \\
- Remote data source which accesses a remote device, accesses a remote application, and then transfers the data to the alarm configuration \\
Example: DataSource_A \\
Below the (now visible) "Application" property, the remote application is displayed as configured in the data source. \\
Example: App_A \\
Note: If the data source is accessed symbolically by means of a symbol file (CODESYS symbolic), then the required symbol file and the corresponding project have to be saved in the same folder. \\
- Local application below which the alarm configuration is located Example: ©<local application>"
\end{tabular} \\
\hline
\end{tabular}

See also
- Object 'Data Source'

Element prop-
erty 'Alarm con-
figuration'
\begin{tabular}{|l|l|}
\hline "Alarm groups" & \begin{tabular}{l} 
Opens the "Select Alarm Group" dialog where you define the alarm groups that \\
you want to display.
\end{tabular} \\
\hline "Priority from" & Least priority for alarm display. (0 to 255). \\
\hline "Priority to" & Greatest priority for alarm display. (0 to 255). \\
\hline "Alarm classes" & \begin{tabular}{l} 
Opens the "Select Class Group" dialog where you define the alarm classes that \\
you want to display.
\end{tabular} \\
\hline "Filter criterion" & \begin{tabular}{l} 
For the "Alarm Banner" element only \\
- "Most important": The alarm with the highest priority (lowest value) is dis- \\
played. \\
- "Newest": The most recent alarm is displayed.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Filter by latch 1" & \begin{tabular}{l}
The generated alarms (previous and current) can be filtered by the contents of "Latch variable 1", which is specified in the configuration of the alarm group. In "Filter type", you define whether or not the filtering is performed by a string value or a numerical value. \\
- "Filter variable": Indicates what the alarms are filtered by. Possible entries: Application variable of data type STRING or WSTRING, or a literal value directly. Examples:PLC_PRG.strFilterVariable, 'STRING'. \\
- "Filter type": Integer value that determines by which criteria the latch variable value is used for filtering. Possible entries: Numerical variable from the application (example: PLC_PRG.diFilterType, or a value directly (example: 2). \\
Possible values: \\
- 0: No filtering \\
- 1: Filter by alarms whose latch variable 1 contains the string specified in "Filter variable". Example: The filter variable contains 'Error 1 ' which is the latch variable 1 of different alarms of type STRING and has the value 'Error 1' ->. Only these alarms are displayed. \\
- 2: Filter by alarms whose latch variable 1 contains the typed literal specified in "Filter variable" according to IEC 61131-3. Examples: T\#1h2s, DINT\#15, REAL\#1.5, FALSE \\
- 3: Filter by alarms whose latch variable 1 contains the LINT literal value specified in "Filter variable". Therefore, the value of the latch variables has to be in the range of \(9,223,372,036,854,775,808\) to 9,223,372,036,854,775,807. \\
- All other values: The behavior is not defined and can change in the future.
\end{tabular} \\
\hline "Filter by time range" & \begin{tabular}{l}
The generated alarms (remote, historical, local) can be displayed for a specified time range. You use the "Filter type" to define whether filtering by time range is enabled or disabled. \\
- "Filter variable, from": Variable of data type DT or DATE_AND_TIME (example: PLC_PRG.filterTimeFrom) for the start time that the alarms are displayed. \\
- "Filter variable, to": Variable of data type DT or DATE_AND_TIME (example: PLC_PRG.filterTimeTo) for the end time that the alarms are displayed. \\
- "Filter type": Variable of integer data type that determines whether "Filter by time range" is enabled or disabled. \\
Possible values: \\
- 1: Filtering is enabled \\
- 0 : Filtering is disabled
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\quad \stackrel{y}{ }\) Chapter 6.4.5.21.3.17 "Dialog 'Selected Alarm Group'" on page 3390
- \(\Rightarrow\) Chapter 6.4.5.21.3.16 "Dialog 'Selected Alarm Class'" on page 3389

\section*{Element property 'General table configuration'}
\begin{tabular}{|l|l|}
\hline "Show row header" & \(\boxed{\text { : Display of the row number at the beginning of the row. }}\) \\
\hline "Show column header" & \(\boxed{ }\)
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Scrollbar size" & \begin{tabular}{l} 
Width of the scrollbar when it runs vertically. Width of the scrollbar when it runs \\
horizontally. Specified in pixels
\end{tabular} \\
\hline \begin{tabular}{l} 
"Automatic line break for alarm \\
message"
\end{tabular} & \begin{tabular}{l} 
: The message text is truncated at the end of the line. \\
\(\square\)
\end{tabular} \\
\hline
\end{tabular}

Element property 'Columns: Column [<n>]'

By default, columns [0] and [1] are configured: "Time stamp" and "Message text". You can create more columns by clicking the "Create new", and remove columns by clicking "Delete".

Animations (dynamic text, font variables), text, and tooltip are not supported.
\begin{tabular}{|c|c|}
\hline "Column header" & The standard header is set and changed here by specifying a new text. \\
\hline "Use text alignment in title" & The text in the column header is aligned according to the current definition in "Text alignment".
\(\square\) : The text in the column header is centered. \\
\hline "Width" & Width of the column (in pixels). \\
\hline "Data type" & \begin{tabular}{l}
Notice about time stamps: For use in a TargetVisu or WebVisu, you can control the date and time format with the help of the global string variables from the library Alarmmanager. library: AlarmGlobals.g_sDateFormat (example: AlarmGlobals.g_sDateFormat := 'MM.yyyy') and AlarmGlobals.g_sTimeFormat (example: AlarmGlobals.g_sTimeFormat := 'HH:mm'). \\
Define the information to be displayed in the column. \\
- "Symbol" \\
- "Time stamp": Date and time of the last status change of the alarm. \\
- "Time stamp active": Date and time of the last activation of the alarm. \\
- "Time stamp inactive": Date and time of the last deactivation of the alarm. \\
- "Time stamp acknowledge": Date and time of the last acknowledgment. \\
- "Value": Current value of the printout \\
- "Message text": Output of the message text \\
- "Priority": Alarm priority \\
- "Class": Alarm class \\
- "State": Alarm state \\
- "Latch Variable <n>": Value of the selected latch variables
\end{tabular} \\
\hline "Text alignment" & \begin{tabular}{l}
Alignment of the text in this column \\
- "Left" \\
- "Centered" \\
- "Right"
\end{tabular} \\
\hline "Color settings" & \begin{tabular}{l}
- "Activate color settings": Boolean variable for activating and deactivating the color settings defined here. Example: PLC_PRG.bColorSettings \\
- "Cell fill color": \\
- "Color variable": Variable for the cell fill color, example: dwCellColor (hexadecimal color definition: 16\#TTRRGGBB) \\
- "Use color also for column header": \(\square\) : The color defined via "Color variable" is used in the column header as well. \\
- "Text color": \\
- "Color variable": Variable for the definition of the text color in the column, example: dwTextColor (hexadecimal color definition: 16\#TTRRGGBB) \\
- "Use color also for column header": \(\square\) : The color defined via "Color variable" is used in the column header as well.
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{4}\) Chapter 6.4.5.10.4 "Animating a color display" on page 2914

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X "\) & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (匝-) to other positions in the editor.

See also
- *y Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline " \(Y\) " & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\Psi}\) ) to other positions in the editor.

Element prop- The properties contain fixed values for the text properties. erty 'Text properties'
\begin{tabular}{|c|c|}
\hline "Font" & Example: "Default"
The "Font" dialog box opens.
Drop-down list with style fonts. \\
\hline "Font color" & \begin{tabular}{l}
Example: "Black"
The "Color" dialog box opens. \\
- : Drop-down list with style colors.
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l}
Whole number (value range from 0 to 255). This determines the transparency of the respective color. \\
Example: 255: The color is opaque. \\
0 : The color is completely transparent. \\
Please note: If the color is a style color and already has a transparency value, then this property is write-protected.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Selection'
\begin{tabular}{|c|c|}
\hline "Selection color" & Fill color of the selected row \\
\hline "Selection font color" & Font color of the selected row \\
\hline "Frame around selected cells" & \(\square\) : A frame is drawn around the selected cells at runtime. \\
\hline "Variable for selected alarm group" & Name of the affected alarm group; type: STRING, WSTRING \\
\hline "Variable for selected alarm ID" & Alarm ID of the affected alarm group; type: STRING, WSTRING \\
\hline "Variable for selected line" & Index of the selected alarm line (0-based). The index can be read and written; integer data type \\
\hline "Variable for valid selection" & \begin{tabular}{l}
TRUE: An alarm line is selected. \\
FALSE: The selection is invalid. For example, for an empty alarm table or when an alarm is not selected yet.
\end{tabular} \\
\hline "Variable for selected alarm information" & \begin{tabular}{l}
Information about the selected alarm. Type AlarmSelectionInfo \\
For easy usage, the function block AlarmSelectionInfoDefault is provided. This FB fills the structure with the memory for 10 messages and 10 latch variables. \\
Example: myAlarmSelectionInfoDefault.AlarmSelectionInfo \\
The following information is available: \\
- sAlarmgroup \\
- uialarmID \\
- timeStampActive \\
- timeStampInactive \\
- timeStampAcknowledge \\
- timeStampLast \\
- paLatchVariables \\
- iLatchVariablesCount \\
- papwsAlarmMessages \\
- dwAlarmMessageTextBufferSize \\
- iAlarmMessagesCount \\
- iSelectionChangedCounter
\end{tabular} \\
\hline
\end{tabular}

\section*{Element prop- Boolean variables are defined here for executing specific actions in the table can be executed at erty 'Control runtime. \\ variables'}
\begin{tabular}{|c|c|}
\hline "Acknowledge selected" & \begin{tabular}{l}
Variable (BOOL) \\
Example: PLC_PRG.bAckSelectedAlarms \\
If the assigned variable is TRUE, then the selected alarm is acknowledged.
\end{tabular} \\
\hline "Acknowledge all visible" & \begin{tabular}{l}
Variable (BOOL) \\
Example: PLC_PRG.bAckVisibleAlarms \\
If the assigned variable is TRUE, then all alarms are acknowledged that are visible in the alarm table.
\end{tabular} \\
\hline "History" & \begin{tabular}{l}
Variable (BOOL) \\
Example: PLC_PRG.bShowHistory \\
If the assigned variable is TRUE, then the history alarms are displayed in addition to the active alarms. In the classic view, the same sort options apply as in normal mode. \\
Note: Acknowledgment is not possible in this view.
\end{tabular} \\
\hline "Freeze scroll position" & \begin{tabular}{l}
Variable (BOOL) \\
Example: PLC_PRG.bFreezeScrollPosition \\
If the assigned variable is TRUE, then the scroll position set in the "History" view is retained, even if a new alarm is active. If not, then the scroll position jumps to the first table row (the newest alarm).
\end{tabular} \\
\hline "Count alarms" & \begin{tabular}{l}
Variable (integer data type) \\
Example: PLC_PRG.iNumberOfAlarms. \\
Number of alarms that are currently displayed in the alarm table. Defined by the alarm table.
\end{tabular} \\
\hline "Count visible rows" & \begin{tabular}{l}
Variable (integer data type) \\
Example: PLC_PRG.iNumberVisibleLines \\
Number of alarms that can be displayed on one page of the alarm table. Defined by the alarm table.
\end{tabular} \\
\hline "Current scroll index" & \begin{tabular}{l}
Variable (integer data type) \\
Example: PLC_PRG.iScrollIndex \\
The index of the first visible row if the alarm table (0-based). The variable can be read and written.
\end{tabular} \\
\hline "Current column sorting" & \begin{tabular}{l}
Variable (integer data type) \\
Example: PLC_PRG.iColSort \\
The variable contains a value of the enumeration "VisuElemsAlarm.VisuEnumAlarmDataType". This value determines the column that sorts the alarm table.
\end{tabular} \\
\hline "Variable for sorting direction" & \begin{tabular}{l}
Variable (BOOL) \\
Example: PLC_PRG.xSortAscending \\
The variable determines the sort order for the entries in the alarm table (TRUE: ascending; FALSE: descending).
\end{tabular} \\
\hline
\end{tabular}

You can also use the "Insert Elements for Acknowledging Alarms" command to define buttons with predefined control variables.

See also
- \({ }^{*}>\) Chapter 6.4.5.21.2.23 "Command 'Add Elements for Alarm Acknowledgement"" on page 3365

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG. iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG. iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\oplus\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
\hline "Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the symbol. \\
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

See also

Element prop- The variables control the element behavior dynamically. erty 'State variables'
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
\\
TRUE: The element is not visible at runtime.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", "X", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

\footnotetext{
Element property 'Access
rights'
}
\begin{tabular}{|c|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages: \\
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
- "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline See also
\end{tabular}
- \(y^{3}\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

\section*{Visualization Element 'Alarm Banner'}

Symbol:


\section*{Category: "Alarm Manager"}

The element is a simplified version of the alarm table. It visualizes a single alarm only. In the element properties, you specify which information is shown. You define the appearance of the element and the variables that control the element behavior.

The alarm banner displays active alarms only. If the alarm is acknowledged, then it disappears from the alarm banner.

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & Example: GenElem Inst_1 \\
\hline "Type of element" & "Alarm Banner" \\
\hline "Data source" & \begin{tabular}{l} 
If you intend to use a remote alarm configuration, then you have to specify the \\
name of the remote application here. If you do not specify anything, the alarm \\
configuration will be located locally.
\end{tabular} \\
\hline
\end{tabular}

\section*{Element prop- \\ erty 'Alarm con- \\ figuration'}
\begin{tabular}{|l|l|}
\hline "Alarm groups" & \begin{tabular}{l} 
Opens the "Select Alarm Group" dialog where you define the alarm groups that \\
you want to display.
\end{tabular} \\
\hline "Priority from" & Least priority for alarm display. (0 to 255). \\
\hline "Priority to" & Greatest priority for alarm display. (0 to 255). \\
\hline "Alarm classes" & \begin{tabular}{l} 
Opens the "Select Class Group" dialog where you define the alarm classes that \\
you want to display.
\end{tabular} \\
\hline "Filter criterion" & \begin{tabular}{l} 
For the "Alarm Banner" element only \\
- "Most important": The alarm with the highest priority (lowest value) is dis- \\
played. \\
- "Newest": The most recent alarm is displayed.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Filter by latch 1" & \begin{tabular}{l}
The generated alarms (previous and current) can be filtered by the contents of "Latch variable 1", which is specified in the configuration of the alarm group. In "Filter type", you define whether or not the filtering is performed by a string value or a numerical value. \\
- "Filter variable": Indicates what the alarms are filtered by. Possible entries: Application variable of data type STRING or WSTRING, or a literal value directly. Examples:PLC_PRG.strFilterVariable, 'STRING'. \\
- "Filter type": Integer value that determines by which criteria the latch variable value is used for filtering. Possible entries: Numerical variable from the application (example: PLC_PRG.diFilterType, or a value directly (example: 2). \\
Possible values: \\
- 0: No filtering \\
- 1: Filter by alarms whose latch variable 1 contains the string specified in "Filter variable". Example: The filter variable contains 'Error 1 ' which is the latch variable 1 of different alarms of type STRING and has the value 'Error 1' ->. Only these alarms are displayed. \\
- 2: Filter by alarms whose latch variable 1 contains the typed literal specified in "Filter variable" according to IEC 61131-3. Examples: T\#1h2s, DINT\#15, REAL\#1.5, FALSE \\
- 3: Filter by alarms whose latch variable 1 contains the LINT literal value specified in "Filter variable". Therefore, the value of the latch variables has to be in the range of \(9,223,372,036,854,775,808\) to 9,223,372,036,854,775,807. \\
- All other values: The behavior is not defined and can change in the future.
\end{tabular} \\
\hline "Filter by time range" & \begin{tabular}{l}
The generated alarms (remote, historical, local) can be displayed for a specified time range. You use the "Filter type" to define whether filtering by time range is enabled or disabled. \\
- "Filter variable, from": Variable of data type DT or DATE_AND_TIME (example: PLC_PRG.filterTimeFrom) for the start time that the alarms are displayed. \\
- "Filter variable, to": Variable of data type DT or DATE_AND_TIME (example: PLC_PRG.filterTimeTo) for the end time that the alarms are displayed. \\
- "Filter type": Variable of integer data type that determines whether "Filter by time range" is enabled or disabled. \\
Possible values: \\
- 1: Filtering is enabled \\
- 0 : Filtering is disabled
\end{tabular} \\
\hline
\end{tabular}

See also
- 乡y Chapter 6.4.5.21.3.17 "Dialog 'Selected Alarm Group"' on page 3390
- \(\Rightarrow\) Chapter 6.4.5.21.3.16 "Dialog 'Selected Alarm Class'" on page 3389

Element prop- By default, columns [0] and [1] are preconfigured: "Time stamp" and "Message text". You create erty 'Columns: Column [<n>]'

Animations (dynamic text, font variables), texts, and tooltips are not supported.
\begin{tabular}{|c|c|}
\hline "Width" & Width of the column (in pixels) \\
\hline "Type of data" & \begin{tabular}{l}
About time stamps: When used in a TargetVisu or WebVisu, you can control the date and time format by means of the global string variables from the library Alarmmanager.library: AlarmGlobals.g_sDateFormat (example: AlarmGlobals.g_sDateFormat := 'MM.yyyy') and AlarmGlobals.g_sTimeFormat (example: AlarmGlobals.g_sTimeFormat := 'HH:mm'). \\
Here you define the information to be displayed in the column. \\
- "Bitmap" \\
- "Time stamp": Date and time of the last status change of the alarm \\
- "Time stamp active": Date and time of the last activation of the alarm \\
- "Time stamp inactive": Date and time of the last deactivation of the alarm \\
- "Time stamp acknowledge": Date and time of the last acknowledgement \\
- "Value": Actual value of the expression \\
- "Message": Output of the message text \\
- "Priority": Alarm priority \\
- "Class": Alarm class \\
- "State": Alarm state \\
- "Latch Variable <n>". Value of the selected latch variables
\end{tabular} \\
\hline "Text alignment" & \begin{tabular}{l}
Alignment of the contents in the column \\
- "Left" \\
- "Centered" \\
- "Right"
\end{tabular} \\
\hline
\end{tabular}

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline "X" & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (品) to other positions in the editor.

See also
- \({ }^{\text {® }}\), Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center' rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline "Y" & \(Y\)-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus}\) ) to other positions in the editor.

Element prop- The properties contain fixed values for the text properties. erty 'Text properties'
\begin{tabular}{|c|c|}
\hline "Font" & \begin{tabular}{l}
Example: "Default"
\(\qquad\) : The "Font" dialog box opens. \\
*: Drop-down list with style fonts.
\end{tabular} \\
\hline "Font color" & Example: "Black"
\(\qquad\) : The "Color" dialog box opens.
: Drop-down list with style colors. \\
\hline "Transparency" & \begin{tabular}{l}
Whole number (value range from 0 to 255). This determines the transparency of the respective color. \\
Example: 255: The color is opaque. \\
0 : The color is completely transparent. \\
Please note: If the color is a style color and already has a transparency value, then this property is write-protected.
\end{tabular} \\
\hline
\end{tabular}
"Acknowledge variable" \(\quad\) A rising edge of this variable acknowledges the alarm.

Handling of mul-
tiple active alarms
\begin{tabular}{|l|l|}
\hline "Automatic switch" & \begin{tabular}{l} 
V: The display in the alarm banner is \\
switched automatically according to the time \\
to the next alarm as configured in "Every N \\
second".
\end{tabular} \\
\hline "Every N second" & \begin{tabular}{l} 
Time period until the next switching. Available \\
only if "Automatic switch" is selected.
\end{tabular} \\
\hline "Next alarm" & \begin{tabular}{l} 
Variable for switching to the next alarm. Avail- \\
able only if "Automatic switch" is not selected.
\end{tabular} \\
\hline "Previous alarm" & \begin{tabular}{l} 
Variable for switching to the previous alarm. \\
Available only if "Automatic switch" is not \\
selected.
\end{tabular} \\
\hline "Multiple alarms active" & \begin{tabular}{l} 
Variable that has the value TRUE if multiple \\
alarms are active.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|c|c|c|}
\hline \multicolumn{3}{|l|}{"Movement"} \\
\hline " \(\times\) " & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular}} \\
\hline " \(Y\) " & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the \(Y\) position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular}} \\
\hline "Rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle1. \\
The midpoint of the element rotates at the "Center" point. This rotation point is shown as the \(\boldsymbol{\oplus}\) symbol. \\
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
\end{tabular} &  \\
\hline "Interior rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle2. \\
In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. \\
The rotation point is shown as the \(\boldsymbol{\oplus}\) symbol. \\
Note: If a static angle of rotation is specified in the "Position \(\rightarrow\) Angle" property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.
\end{tabular} &  \\
\hline
\end{tabular}

You can link the variables to a unit conversion.


See also

Element prop- The variables control the element behavior dynamically. erty 'State variables'
\begin{tabular}{|l|l}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
\\
TRUE: The element is not visible at runtime.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- Requirement: User management is set up for the visualization.
erty 'Access
rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages:
\end{tabular} \\
& \begin{tabular}{l} 
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
• "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\xi^{*}\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights"' on page 3366

\section*{Visualization Element 'Bar Display'}

Symbol:

Category: "Measurement Controls"
The element displays the value of a variable.
See also
- ⓨ Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & Example: GenElemInst_2 \\
\hline "Type of element" & "Bar Display" \\
\hline "Value" & \begin{tabular}{l} 
Variable (numeric data type) \\
The value of the variable is displayed as a bar length.
\end{tabular} \\
\hline
\end{tabular}
\(\begin{array}{ll}\text { Element prop- } & \text { The properties contain fixed values for the coordinates of the point of rotation. This point of } \\ \text { erty 'Center' } & \text { rotation is shown as the } \boldsymbol{\oplus} \text { symbol. The point is used as the center for rotating and scaling. }\end{array}\)
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline " \(Y\) " & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus}\) ) to other positions in the editor.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X\) " & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (ए-) to other positions in the editor.

\section*{See also}
- \({ }^{\wedge} \Rightarrow\) Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop-
erty 'Back-
ground'
\begin{tabular}{|c|c|}
\hline \multirow[t]{2}{*}{"Background color"} & Drop-down list with background colors \\
\hline & Note: This property depends on the style. For example, there are no heterochromatic background images for "FlatStyle" and "Whitestyle". \\
\hline "Own image" & \begin{tabular}{l}
- "image": Image ID of the background image. You select the background image from an image pool by clicking the \(\ldots .\). symbol. \\
Info: If you specify the "<default>" value or select the image from the "Default" category in the input assistant, then the original element background image is used. \\
- "Transparent color": Color of pixels that are displayed as transparent. Selection from drop-down list or input assistant.
\end{tabular} \\
\hline "Optimized drawing" & \begin{tabular}{l}
The background image is drawn one time. If there is a change in the foreground, then only the affected part of the image is redrawn.
\(\square\) The background image is redrawn in cycles. \\
Note: Deactivating this option is sensible only in certain exceptional cases.
\end{tabular} \\
\hline
\end{tabular}

\section*{Element prop- \\ erty 'Bar'}
\begin{tabular}{|c|c|}
\hline "Diagram type" & \begin{tabular}{l}
Position of the scale \\
- "Scale besides bar" \\
- "Scale in bar" \\
- "Bar in scale" \\
- "No scale"
\end{tabular} \\
\hline "Orientation" & \begin{tabular}{l}
Orientation depending on the ratio of width to height of the Bar Display: \\
- "Horizontal" \\
- "Vertical"
\end{tabular} \\
\hline "Running direction" & \begin{tabular}{l}
Direction the values are increased. \\
Drop-down list for "Orientation Horizontal": \\
- "Left to right" \\
- "Right to left" \\
Drop-down list for "Orientation Vertical": \\
- "Bottom to top" \\
- "Top to bottom"
\end{tabular} \\
\hline "Optimum size for bar" & \begin{tabular}{l}
The bar width requires the majority of the element surface. \\
Note: This property depends on the style. It is not provided for "FlatStyle" or "WhiteStyle".
\end{tabular} \\
\hline
\end{tabular}

\section*{Element prop- \\ erty 'Scale'}
\begin{tabular}{|c|c|}
\hline "Scale start" & \begin{tabular}{l}
Least value of the scale and the lower limit of the value range for the element. \\
Example: 0 \\
*: The property "Variable" is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the scale start. \\
Example: PLC_PRG.iScaleStart \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iScaleStart : INT := 0;
END_VAR

```
\end{tabular} \\
\hline "Scale end" & \begin{tabular}{l}
Greatest value of the scale and the upper limit of the value range for the element. \\
Example: 100 \\
*: The property "Variable" is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the scale end. \\
Example: PLC_PRG.iScaleEnd \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iScaleEnd : INT := 120;
END_VAR

```
\end{tabular} \\
\hline "Main scale" & \begin{tabular}{l}
Distance between 2 values on the rough scale. \\
Example: 10 \\
p: The property "Variable" is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the distance. \\
Example: PLC_PRG.iMainScale \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iMainScale : INT := 20;
END VAR

```
\end{tabular} \\
\hline "Subscale" & \begin{tabular}{l}
Distance between 2 values on the fine scale. \\
You can hide the fine scale by setting the value to 0 . \\
Example: 2 \\
中: The property "Variable" is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the spacing. \\
Example: PLC_PRG.iSubScale \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iSubScale : INT := 5;
END VAR

```
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Scale line width" & \begin{tabular}{l}
Specified in pixels. \\
Example: 3
\end{tabular} \\
\hline "Scale color" & \begin{tabular}{l}
Color of scale lines \\
- ...: \\
The "Color" dialog box opens. \\
- \(\quad\) : A drop-down list with color names opens.
\end{tabular} \\
\hline "Scale in 3D" & \begin{tabular}{l}
Tick marks are displayed with slight 3D shadowing. \\
Note: This property depends on the style. Not available for "FlatStyle".
\end{tabular} \\
\hline "Element frame" & \(\checkmark\) : A frame is drawn around the element. \\
\hline
\end{tabular}

Element property 'Label'
\begin{tabular}{|l|l|}
\hline "Unit" & \begin{tabular}{l} 
Text that is displayed in the element. \\
Example: Units displayed in m/s.
\end{tabular} \\
\hline "Font" & \begin{tabular}{l} 
Font for labels (example: scale numbering). \\
Selection from the drop-down list or by clicking the "" button. \\
\hline "...
\end{tabular} \\
\hline "Mcale format (C Syntax)" text width of labels" & \begin{tabular}{l} 
Values scaled in "printf" syntax \\
Examples: \%d, \%5.2f
\end{tabular} \\
\hline & \begin{tabular}{l} 
(optional) Value that redefines the maximum width of the scale label. The correct \\
value is normally set automatically. \\
Note: Change this value only if the automatic adjustment does not yield the \\
expected result.
\end{tabular} \\
\hline "Text height of labels" & \begin{tabular}{l} 
(optional) Value that redefines the maximum height of the scale label. The cor- \\
rect value is normally set automatically. \\
Note: Change this value only if the automatic adjustment does not yield the \\
expected result.
\end{tabular} \\
\hline "Font color" & Selection from the drop-down list or by clicking the ... button. \\
\hline
\end{tabular}

\section*{Element prop-}
erty 'Positio-
ning'
\begin{tabular}{|l|l|}
\hline "Horizontal offset" & \begin{tabular}{l} 
Distance from the scale (bar) to the horizontal element frame \\
Specified in pixels. \\
Used for achieving the exact position relative to the background image.
\end{tabular} \\
\hline "Vertical offset" & \begin{tabular}{l} 
Distance from the scale (bar) to the vertical element frame \\
Specified in pixels. \\
Used for achieving the exact position relative to the background image.
\end{tabular} \\
\hline "Horizontal scaling" & \begin{tabular}{l} 
Horizontal division of the scale \\
Specified in pixels. \\
Used for achieving the exact positioning relative to the background image.
\end{tabular} \\
\hline "Vertical scaling" & \begin{tabular}{l} 
Vertical division of the scale \\
Specified in pixels. \\
Used for achieving the exact positioning relative to the background image.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Colors'
\begin{tabular}{|c|c|}
\hline "Graph color" & Color of the bar \\
\hline "Bar background" & The background of the bar is black.
The background of the bar is white. \\
\hline "Frame color" & Color that the frames are drawn. \\
\hline "Switch whole color" & The total color of the bar is switched to the color of the color area of the current value. \\
\hline "Use gradient color for bar" & - Bar is displayed with a gradient. \\
\hline "Color range markers" & \begin{tabular}{l}
The color areas can be separated from each other inside the bar with a vertical mark. \\
- "No markers": No display. \\
- "Marker forwards": The color of the vertical mark corresponds to the color of the previous color area. \\
- "Marker backwards": The color of the vertical mark corresponds to the color of the next color area.
\end{tabular} \\
\hline \multicolumn{2}{|l|}{"Color areas"} \\
\hline "Create new" & A new color area is added. \\
\hline "Delete" & The color area is removed from the list. \\
\hline "Begin of area" & Start value of the color area \\
\hline "End of area" & End value of the color area \\
\hline "Color" & Color that is used for displaying the area. \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|c|c|c|}
\hline "Movement" & & \\
\hline " \(X\) " & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular}} \\
\hline "Y" & \begin{tabular}{l}
Variable (numeric data type). Defines the Y position Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the el
\end{tabular} & \begin{tabular}{l}
in pixels). \\
nent downwards.
\end{tabular} \\
\hline "Rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle1. \\
The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol. \\
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
\end{tabular} &  \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol.
\end{tabular} \\
\begin{tabular}{l} 
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

The "X", " \(Y\) ", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also

Element prop- The variables control the element behavior dynamically.
erty 'State varia-
bles'
\begin{tabular}{|l|l|}
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

Element prop- Requirement: User management is set up for the visualization. erty 'Access
rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages: \\
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
- "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

See also
- Chapter 6.4.5.21.3.1 "Dialog 'Access Rights"' on page 3366

\section*{Visualization Element 'Meter 90'}

Symbol:


\section*{Category: "Measurement Controls"}

The element displays the value of a variable. The needle is positioned according to the value of the assigned variable. A meter is used to represent a tachometer, for example.

\section*{Element proper- \\ ties}
\begin{tabular}{|l|l|}
\hline "Element name" & Example: GenElemInst_1 \\
\hline "Type of element" & "Meter 90"" \\
\hline "Value" & \begin{tabular}{l} 
Variable (numeric data type) \\
The variable value determines the pointer direction of the element.
\end{tabular} \\
\hline
\end{tabular}

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline "X" & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (■-) to other positions in the editor.

See also
- Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

\section*{Element prop- \\ erty 'Back- \\ ground'}
\begin{tabular}{|l|l|}
\hline "Image color" & List box containing background colors \\
\hline "Own image" & \begin{tabular}{l} 
- "Image": ID of the background image. \\
\\
\\
\\
\\
\\
\\
\\
\\
\\
You select the background image from an image pool by clicking you specify the value "<default>" or select the image from the \\
"Defautl" category in the Input Assistant, then the original element back- \\
ground image is used. \\
- "Transparency color": Selection from list box or Input Assistant.
\end{tabular} \\
\hline
\end{tabular}

\section*{Element prop- \\ erty 'Arrow'}
\begin{tabular}{|l|l|}
\hline "Hand style" & Drop-down list with different arrow types \\
\hline "Color" & - \(\quad . .\). The "Color" dialog box opens. \\
& - : Drop-down list with color names \\
\hline "Angle range" & Drop-down list for the alignment of the element \\
\hline "Additional arrow" & : An additional arrow is shown inside the scale. \\
\hline
\end{tabular}

\section*{Element property 'Scale'}
\begin{tabular}{|c|c|}
\hline "Sub scale position" & \begin{tabular}{l}
- "Outside": The subscale is displayed on the outer scale ring. ("Frame outside") \\
- "Inside": The subscale is displayed on the inner scale ring. ("Frame inside")
\end{tabular} \\
\hline "Scale type" & \begin{tabular}{l}
Type of scale \\
- "Lines" \\
- "Dots" \\
- "Squares"
\end{tabular} \\
\hline "Scale start" & \begin{tabular}{l}
Least value of the scale and the lower limit of the value range for the element Example: 0 \\
*: The "Variable" property is displayed in the line below this.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the scale start \\
Example: PLC_PRG.iScaleStart \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iScaleStart : INT := 0;
END_VAR

```
\end{tabular} \\
\hline "Scale end" & \begin{tabular}{l}
Greatest value of the scale and the upper limit of the value range for the element \\
Example: 100 \\
p: The "Variable" property is shown below this.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the scale end \\
Example: PLC_PRG.iScaleEnd \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iScaleEnd : INT := 120;
END_VAR

```
\end{tabular} \\
\hline "Main scale" & \begin{tabular}{l}
Distance between two values on the main scale \\
Example: 10 \\
p: The "Variable" property is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type) Contains the distance between two values on the main scale \\
Example: PLC_PRG.iMainScale \\
Declaration: \\
PROGRAM PLC_PRG \\
VAR \\
iMainScale : INT := 20;
\end{tabular} \\
\hline "Sub scale" & \begin{tabular}{l}
Distance between two values on the fine scale You can hide the fine scale by setting the value to 0 . \\
Example: 2 \\
*: The "Variable" property is shown below this.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type) Contains the distance between two values on the fine scale \\
Example: PLC_PRG.iSubScale \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iSubScale : INT := 5;
END VAR

```
\end{tabular} \\
\hline "Scale line width" & \begin{tabular}{l}
Specified in pixels \\
Example: 3
\end{tabular} \\
\hline "Scale color" & \begin{tabular}{l}
Color of scale lines \\
The "Color" dialog opens. \\
- - A list box with style colors opens.
\end{tabular} \\
\hline "Scale in 3D" & Scale lines are displayed with soft 3D shadowing. Note: This property is not displayed in "FlatStyle". \\
\hline "Show scale" & 四: The scale is displayed. \\
\hline "Frame inside" & \(\square\) : A frame is drawn at the inner end of the scale. \\
\hline "Frame outside" & \(\square\) : A frame is drawn at the outer end of the scale. \\
\hline
\end{tabular}

Element prop-
erty 'Label'
\begin{tabular}{|l|l|}
\hline "Label" & \begin{tabular}{l} 
Selection list \\
- "Outside": Scale values are placed outside of the scale. \\
- "Inside": Scale values are placed inside of the scale.
\end{tabular} \\
\hline "Unit" & \begin{tabular}{l} 
Text that is displayed in the element. \\
Example: Units displayed in m/s.
\end{tabular} \\
\hline "Font" & \begin{tabular}{l} 
Font for labels (example: scale numbering). \\
Selection from the drop-down list or by clicking the "" button. ...
\end{tabular} \\
\hline "Scale format (C Syntax)" & \begin{tabular}{l} 
Values scaled in "printf" syntax \\
Examples: \%d, \%5.2f
\end{tabular} \\
\hline "Max. text width of labels" & \begin{tabular}{l} 
(optional) Value that redefines the maximum width of the scale label. The correct \\
value is normally set automatically. \\
Note: Change this value only if the automatic adjustment does not yield the \\
expected result.
\end{tabular} \\
\hline "Text height of labels" & \begin{tabular}{l} 
(optional) Value that redefines the maximum height of the scale label. The cor- \\
rect value is normally set automatically. \\
Note: Change this value only if the automatic adjustment does not yield the \\
expected result.
\end{tabular} \\
\hline "Font color" & Selection from the drop-down list or by clicking the ■... button. \\
\hline
\end{tabular}

\section*{Element property 'Positioning'}
\begin{tabular}{|l|l|}
\hline "Usage of" & \begin{tabular}{l} 
- "Preset style values": Values from the current style \\
- "User-defined settings": The subnode "Positioning" appears.
\end{tabular} \\
\hline \begin{tabular}{l} 
"Positioning" \\
Requirement: "User-defined settings" is selected as "Usage of". \\
The displayed positioning settings depend on the type of needle instrument and Potentiometer, and partially on \\
whether a custom background image is selected. The following settings are used for achieving the exact position \\
relative to the background image.
\end{tabular} \\
\hline "Needle movement" & Length of the needle (in pixels) \\
\hline "Scale movement" & \begin{tabular}{l} 
Distance from the tick marks to the center (in pixels) \\
Requirement: A customer image is selected as "Background".
\end{tabular} \\
\hline "Scale length" & \begin{tabular}{l} 
Length of the tick marks (in pixels) \\
Requirement: A customer image is selected as "Background".
\end{tabular} \\
\hline "Label offset": & Distance from the labels to the tick marks (in pixels) \\
\hline "Unit offset": & Distance of the unit text "Label \(\mathbf{\rightarrow}\) Unit" from the upper scale edge (in pixels) \\
\hline "Origin offset" & \begin{tabular}{l} 
Offset of the element (in pixels) \\
Requirement: For the elements "Meter 180"" and "Meter 90"", this property is \\
displayed only if a custom image is selected as "Background".
\end{tabular} \\
\hline
\end{tabular}

\section*{Element prop-}
erty 'Colors'
\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|l|}{"Color areas"} \\
\hline "Durable color areas" & \begin{tabular}{l}
: All color areas are visible, regardless of the current value. \\
Only the color area is visible that includes the current value.
\end{tabular} \\
\hline "Use colors for scale" & \: Colors in the color area are used only for the scale and frame. \\
\hline "Color areas" & \\
\hline "Create new" & A new color area is added to the "Elements" view. \\
\hline "Delete" & The color area is removed from the list and the list is refreshed. \\
\hline "Begin of area" & \begin{tabular}{l}
Start value of the color area \\
Example: 20 \\
p: The property "Variable" is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the start value. \\
Example: PLC_PRG.iColorAreaStarto \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iColorAreaStart0 : INT := 80;
END_VAR

```
\end{tabular} \\
\hline "End of area" & \begin{tabular}{l}
End value of the color area \\
Example: 120 \\
: The property "Variable" is shown below.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Variable" & Variable (integer data type). Contains the end value. \\
& Example: iColorAreaEnd0 \\
& Declaration: \\
& \begin{tabular}{l} 
PROGRAM PLC_PRG \\
VAR iColorAreaEndo : INT \(:=100 ;\) \\
\\
\\
End_VAR
\end{tabular} \\
\hline "Color" & Color that is used for displaying the area. \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG. iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\oplus\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
\hline "Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol. \\
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
\hline
\end{tabular}


Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages:
\end{tabular} \\
& - "Not set. Full rights.": Access rights for all user groups : "operable" \\
- "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular}

See also
- \(\Leftrightarrow\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights"' on page 3366

See also
- « Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872 ""

\section*{Visualization Element 'Meter 180 \({ }^{\circ}\)}

Symbol:
The " \(X\) ", " \(Y\) ", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also
-


\section*{Category: "Measurement Controls"}

The element displays the value of a variable. The needle is positioned according to the value of the assigned variable on a scale. A meter is used to represent a tachometer, for example.

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & Example: GenElemInst_1 \\
\hline "Type of element" & "Meter 180" \\
\hline "Value" & \begin{tabular}{l} 
Variable (numeric data type) \\
The variable value determines the pointer direction of the element.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center' rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline " \(Y\) " & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus}\) ) to other positions in the editor.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The position defines the location and size of the element in the visualization window. These erty 'Position' are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X\) " & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols ( ( - ) to other positions in the editor.

See also
- © Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop-
erty 'Back-
ground'
\begin{tabular}{|c|c|}
\hline "Image color" & List box containing background colors \\
\hline "Own image" & \begin{tabular}{l}
- "Image": ID of the background image. \\
You select the background image from an image pool by clicking \(\square\) Info: If you specify the value "<default>" or select the image from the "Default" category in the Input Assistant, then the original element background image is used. \\
- "Transparency color": Selection from list box or Input Assistant.
\end{tabular} \\
\hline
\end{tabular}

\section*{Element prop-}
erty 'Arrow'
\begin{tabular}{|l|l|}
\hline "Hand style" & Drop-down list with different arrow types \\
\hline "Color" & - ...: The "Color" dialog box opens. \\
& - : Drop-down list with color names \\
\hline "Angle range" & Drop-down list for the alignment of the element \\
\hline "Additional arrow" & 目: An additional arrow is shown inside the scale. \\
\hline
\end{tabular}

\section*{Element prop-}
erty 'Scale'
\begin{tabular}{|l|l|}
\hline "Sub scale position" & \begin{tabular}{l} 
- "Outside": The subscale is displayed on the outer scale ring. ("Frame \\
outside") \\
- "Inside": The subscale is displayed on the inner scale ring. ("Frame inside")
\end{tabular} \\
\hline "Scale type" & \begin{tabular}{l} 
Type of scale \\
- "Lines" \\
- "Dots" \\
- "Squares"
\end{tabular} \\
\hline "Scale start" & \begin{tabular}{l} 
Least value of the scale and the lower limit of the value range for the element \\
Example: 0 \\
: The "Variable" property is displayed in the line below this.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the scale start \\
Example: PLC_PRG.iScaleStart \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iScaleStart : INT := 0;
END_VAR

```
\end{tabular} \\
\hline "Scale end" & \begin{tabular}{l}
Greatest value of the scale and the upper limit of the value range for the element Example: 100 \\
: The "Variable" property is shown below this.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the scale end \\
Example: PLC_PRG.iScaleEnd \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iScaleEnd : INT := 120;
END_VAR

```
\end{tabular} \\
\hline "Main scale" & \begin{tabular}{l}
Distance between two values on the main scale \\
Example: 10 \\
*: The "Variable" property is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type) Contains the distance between two values on the main scale \\
Example: PLC_PRG.iMainScale \\
Declaration: \\
PROGRAM PLC_PRG \\
VAR \\
iMainScale : INT := 20; \\
END VAR
\end{tabular} \\
\hline "Sub scale" & \begin{tabular}{l}
Distance between two values on the fine scale You can hide the fine scale by setting the value to 0 . Example: 2 \\
*: The "Variable" property is shown below this.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type) Contains the distance between two values on the fine scale \\
Example: PLC_PRG.iSubScale \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iSubScale : INT := 5;
END_VAR

```
\end{tabular} \\
\hline "Scale line width" & \begin{tabular}{l}
Specified in pixels \\
Example: 3
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Scale color" & \begin{tabular}{l}
Color of scale lines
The "Color" dialog opens. \\
- - A list box with style colors opens.
\end{tabular} \\
\hline "Scale in 3D" & Scale lines are displayed with soft 3D shadowing. Note: This property is not displayed in "FlatStyle". \\
\hline "Show scale" & - The scale is displayed. \\
\hline "Frame inside" & \(\square\) : A frame is drawn at the inner end of the scale. \\
\hline "Frame outside" & : A frame is drawn at the outer end of the scale. \\
\hline
\end{tabular}

Element property 'Label'
\begin{tabular}{|c|c|}
\hline "Label" & \begin{tabular}{l}
Selection list \\
- "Outside": Scale values are placed outside of the scale. \\
- "Inside": Scale values are placed inside of the scale.
\end{tabular} \\
\hline "Unit" & \begin{tabular}{l}
Text that is displayed in the element. \\
Example: Units displayed in \(\mathrm{m} / \mathrm{s}\).
\end{tabular} \\
\hline "Font" & \begin{tabular}{l}
Font for labels (example: scale numbering). \\
Selection from the drop-down list or by clicking the "" button.
\end{tabular} \\
\hline "Scale format (C Syntax)" & \begin{tabular}{l}
Values scaled in "printf" syntax \\
Examples: \%d, \(\% 5.2 f\)
\end{tabular} \\
\hline "Max. text width of labels" & \begin{tabular}{l}
(optional) Value that redefines the maximum width of the scale label. The correct value is normally set automatically. \\
Note: Change this value only if the automatic adjustment does not yield the expected result.
\end{tabular} \\
\hline "Text height of labels" & \begin{tabular}{l}
(optional) Value that redefines the maximum height of the scale label. The correct value is normally set automatically. \\
Note: Change this value only if the automatic adjustment does not yield the expected result.
\end{tabular} \\
\hline "Font color" & Selection from the drop-down list or by clicking the ... button. \\
\hline
\end{tabular}

\section*{Element prop- \\ erty 'Positio- \\ ning'}
\begin{tabular}{|l|l|}
\hline "Usage of" & - "Preset style values": Values from the current style \\
- "User-defined settings": The subnode "Positioning" appears.
\end{tabular}\(|\)\begin{tabular}{l|l|}
\hline \begin{tabular}{l} 
Requirement: "User-defined settings" is selected as "Usage of". \\
The displayed positioning settings depend on the type of needle instrument and Potentiometer, and partially on \\
whether a custom background image is selected. The following settings are used for achieving the exact position \\
relative to the background image.
\end{tabular} \\
\hline "Needle movement" & \multicolumn{2}{|l|}{ Length of the needle (in pixels) } \\
\hline "Scale movement" & \begin{tabular}{l} 
Distance from the tick marks to the center (in pixels) \\
Requirement: A customer image is selected as "Background".
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Scale length" & \begin{tabular}{l} 
Length of the tick marks (in pixels) \\
Requirement: A customer image is selected as "Background".
\end{tabular} \\
\hline "Label offset": & Distance from the labels to the tick marks (in pixels) \\
\hline "Unit offset": & Distance of the unit text "Label \(\rightarrow\) Unit" from the upper scale edge (in pixels) \\
\hline "Origin offset" & \begin{tabular}{l} 
Offset of the element (in pixels) \\
Requirement: For the elements "Meter \(180^{\circ} "\) and "Meter \(90{ }^{\circ} ", ~ t h i s ~ p r o p e r t y ~ i s ~\) \\
displayed only if a custom image is selected as "Background".
\end{tabular} \\
\hline
\end{tabular}

Element prop-
erty 'Colors'
\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|l|}{"Color areas"} \\
\hline "Durable color areas" & All color areas are visible, regardless of the current value.
Only the color area is visible that includes the current value. \\
\hline "Use colors for scale" & \(\checkmark\) : Colors in the color area are used only for the scale and frame. \\
\hline "Color areas" & \\
\hline "Create new" & A new color area is added to the "Elements" view. \\
\hline " Delete" & The color area is removed from the list and the list is refreshed. \\
\hline "Begin of area" & \begin{tabular}{l}
Start value of the color area \\
Example: 20 \\
*: The property "Variable" is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the start value. \\
Example: PLC_PRG.iColorAreaStart0 \\
Declaration: \\
PROGRAM PLC_PRG \\
VAR \\
iColorAreaStart0 : INT := 80;
\end{tabular} \\
\hline "End of area" & \begin{tabular}{l}
End value of the color area \\
Example: 120 \\
*: The property "Variable" is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the end value. \\
Example: iColorAreaEndo \\
Declaration: \\
PROGRAM PLC_PRG \\
VAR \\
iColorAreaEnd0 : INT := 100;
\end{tabular} \\
\hline "Color" & Color that is used for displaying the area. \\
\hline
\end{tabular}

\section*{Element property 'Absolute movement'}

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|c|c|c|}
\hline \multicolumn{3}{|l|}{"Movement"} \\
\hline "X" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular}} \\
\hline "Y" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the \(Y\) position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular}} \\
\hline "Rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle1. \\
The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol. \\
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
\end{tabular} &  \\
\hline "Interior rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle2. \\
In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. \\
The rotation point is shown as the \(\boldsymbol{\oplus}\) symbol. \\
Note: If a static angle of rotation is specified in the "Position \(\rightarrow\) Angle" property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.
\end{tabular} &  \\
\hline
\end{tabular}

You can link the variables to a unit conversion.


See also

Element prop- The variables control the element behavior dynamically. erty 'State variables'
\begin{tabular}{|l|l}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
\\
TRUE: The element is not visible at runtime.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

Element prop- Requirement: User management is set up for the visualization. erty 'Access
rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages:
\end{tabular} \\
& - "Not set. Full rights.": Access rights for all user groups : "operable" \\
- "Rights are set: Limited rights": Access is restricted for at least one group. \\
\hline
\end{tabular}

See also
- \& Chapter 6.4.5.21.3.1 "Dialog 'Access Rights"' on page 3366

See also
- \({ }^{4}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Meter'}

Symbol:


\section*{Category: "Measurement Controls"}

The element displays the value of a variable. The needle is positioned according to the value of the assigned variable. A meter is used to represent a tachometer, for example.

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & Example: GenElemInst_1 \\
\hline "Type of element" & "Meter" \\
\hline "Value" & \begin{tabular}{l} 
Variable (numeric data type). \\
The variable value determines the pointer direction of the element.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center' rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X "\) & X-coordinate of the point of rotation \\
\hline\(" Y "\) & \(Y\)-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus}\) ) to other positions in the editor.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X\) " & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols ([『) to other positions in the editor.

See also
- * Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop-
erty 'Back-
ground'
\begin{tabular}{|c|c|}
\hline "Image color" & List box containing background colors \\
\hline "Own image" & \begin{tabular}{l}
- "Image": ID of the background image. \\
You select the background image from an image pool by clicking \(\square\) Info: If you specify the value "<default>" or select the image from the "Default" category in the Input Assistant, then the original element background image is used. \\
- "Transparency color": Selection from list box or Input Assistant.
\end{tabular} \\
\hline
\end{tabular}

\section*{Element prop-}
erty 'Arrow'
\begin{tabular}{|l|l|}
\hline "Hand style" & Drop-down list with different arrow types \\
\hline "Color" & - ...: The "Color" dialog box opens. \\
- : Drop-down list with color names
\end{tabular}

\section*{Element prop- \\ erty 'Scale'}
\begin{tabular}{|l|l|}
\hline "Sub scale position" & \begin{tabular}{l} 
- "Outside": The subscale is displayed on the outer scale ring. ("Frame \\
outside") \\
- "Inside": The subscale is displayed on the inner scale ring. ("Frame inside")
\end{tabular} \\
\hline "Scale type" & \begin{tabular}{l} 
Type of scale \\
- "Lines" \\
- "Dots" \\
- "Squares"
\end{tabular} \\
\hline "Scale start" & \begin{tabular}{l} 
Least value of the scale and the lower limit of the value range for the element \\
Example: 0 \\
": The "Variable" property is displayed in the line below this.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the scale start \\
Example: PLC_PRG.iScaleStart \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iScaleStart : INT := 0;
END_VAR

```
\end{tabular} \\
\hline "Scale end" & \begin{tabular}{l}
Greatest value of the scale and the upper limit of the value range for the element Example: 100 \\
*: The "Variable" property is shown below this.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the scale end \\
Example: PLC_PRG.iScaleEnd \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iScaleEnd : INT := 120;
END_VAR

```
\end{tabular} \\
\hline "Main scale" & \begin{tabular}{l}
Distance between two values on the main scale Example: 10 \\
*: The "Variable" property is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type) Contains the distance between two values on the main scale \\
Example: PLC_PRG.iMainScale \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iMainScale : INT := 20;
END_VAR

```
\end{tabular} \\
\hline "Sub scale" & \begin{tabular}{l}
Distance between two values on the fine scale You can hide the fine scale by setting the value to 0 . Example: 2 \\
*: The "Variable" property is shown below this.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type) Contains the distance between two values on the fine scale \\
Example: PLC_PRG.iSubScale \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iSubScale : INT := 5;
END_VAR

```
\end{tabular} \\
\hline "Scale line width" & \begin{tabular}{l}
Specified in pixels \\
Example: 3
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Scale color" & \begin{tabular}{l}
Color of scale lines
: The "Color" dialog opens. \\
- - A list box with style colors opens.
\end{tabular} \\
\hline "Scale in 3D" & Scale lines are displayed with soft 3D shadowing. Note: This property is not displayed in "FlatStyle". \\
\hline "Show scale" & \(\square\) : The scale is displayed. \\
\hline "Frame inside" & \(\square\) : A frame is drawn at the inner end of the scale. \\
\hline "Frame outside" & - A frame is drawn at the outer end of the scale. \\
\hline
\end{tabular}

Element property 'Label'
\begin{tabular}{|l|l|}
\hline "Label" & \begin{tabular}{l} 
Selection list \\
- "Outside": Scale values are placed outside of the scale. \\
- "Inside": Scale values are placed inside of the scale.
\end{tabular} \\
\hline "Unit" & \begin{tabular}{l} 
Text that is displayed in the element. \\
Example: Units displayed in m/s.
\end{tabular} \\
\hline "Font" & \begin{tabular}{l} 
Font for labels (example: scale numbering). \\
Selection from the drop-down list or by clicking the ""button....
\end{tabular} \\
\hline "Scale format (C Syntax)" & \begin{tabular}{l} 
Values scaled in "printf" syntax \\
Examples: od, \%5.2f
\end{tabular} \\
\hline "Max. text width of labels" & \begin{tabular}{l} 
(optional) Value that redefines the maximum width of the scale label. The correct \\
value is normally set automatically. \\
Note: Change this value only if the automatic adjustment does not yield the \\
expected result.
\end{tabular} \\
\hline "Text height of labels" & \begin{tabular}{l} 
(optional) Value that redefines the maximum height of the scale label. The cor- \\
rect value is normally set automatically. \\
Note: Change this value only if the automatic adjustment does not yield the \\
expected result.
\end{tabular} \\
\hline Selection from the drop-down list or by clicking the ■... button. \\
\hline
\end{tabular}

\section*{Element prop- \\ erty 'Positio- \\ ning'}
\begin{tabular}{|l|l|}
\hline "Usage of" & - "Preset style values": Values from the current style \\
- "User-defined settings": The subnode "Positioning" appears.
\end{tabular}\(|\)\begin{tabular}{l|l|}
\hline \begin{tabular}{l} 
Requirement: "User-defined settings" is selected as "Usage of". \\
The displayed positioning settings depend on the type of needle instrument and Potentiometer, and partially on \\
whether a custom background image is selected. The following settings are used for achieving the exact position \\
relative to the background image.
\end{tabular} \\
\hline "Needle movement" & Length of the needle (in pixels) \\
\hline "Scale movement" & \begin{tabular}{l} 
Distance from the tick marks to the center (in pixels) \\
Requirement: A customer image is selected as "Background".
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Scale length" & \begin{tabular}{l} 
Length of the tick marks (in pixels) \\
Requirement: A customer image is selected as "Background".
\end{tabular} \\
\hline "Label offset": & Distance from the labels to the tick marks (in pixels) \\
\hline "Unit offset": & Distance of the unit text "Label \(\rightarrow\) Unit" from the upper scale edge (in pixels) \\
\hline "Origin offset" & \begin{tabular}{l} 
Offset of the element (in pixels) \\
Requirement: For the elements "Meter \(180^{\circ} "\) and "Meter \(90{ }^{\circ} ", ~ t h i s ~ p r o p e r t y ~ i s ~\) \\
displayed only if a custom image is selected as "Background".
\end{tabular} \\
\hline
\end{tabular}

Element prop-
erty 'Colors'
\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|l|}{"Color areas"} \\
\hline "Durable color areas" & All color areas are visible, regardless of the current value.
Only the color area is visible that includes the current value. \\
\hline "Use colors for scale" & \(\checkmark\) : Colors in the color area are used only for the scale and frame. \\
\hline "Color areas" & \\
\hline "Create new" & A new color area is added to the "Elements" view. \\
\hline " Delete" & The color area is removed from the list and the list is refreshed. \\
\hline "Begin of area" & \begin{tabular}{l}
Start value of the color area \\
Example: 20 \\
*: The property "Variable" is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the start value. \\
Example: PLC_PRG.iColorAreaStart0 \\
Declaration: \\
PROGRAM PLC_PRG \\
VAR \\
iColorAreaStart0 : INT := 80;
\end{tabular} \\
\hline "End of area" & \begin{tabular}{l}
End value of the color area \\
Example: 120 \\
*: The property "Variable" is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the end value. \\
Example: iColorAreaEndo \\
Declaration: \\
PROGRAM PLC_PRG \\
VAR \\
iColorAreaEnd0 : INT := 100;
\end{tabular} \\
\hline "Color" & Color that is used for displaying the area. \\
\hline
\end{tabular}

\section*{Element property 'Absolute movement'}

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|c|c|c|}
\hline \multicolumn{3}{|l|}{"Movement"} \\
\hline " \(\times\) " & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular}} \\
\hline " \(Y\) " & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the \(Y\) position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular}} \\
\hline "Rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle1. \\
The midpoint of the element rotates at the "Center" point. This rotation point is shown as the \(\boldsymbol{\oplus}\) symbol. \\
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
\end{tabular} &  \\
\hline "Interior rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle2. \\
In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. \\
The rotation point is shown as the \(\boldsymbol{\oplus}\) symbol. \\
Note: If a static angle of rotation is specified in the "Position \(\rightarrow\) Angle" property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.
\end{tabular} &  \\
\hline
\end{tabular}

You can link the variables to a unit conversion.


See also

Element prop- The variables control the element behavior dynamically. erty 'State variables'
\begin{tabular}{|l|l}
\hline "Invisible" & Variable (BOOL). Toggles the visibility of the element. \\
& TRUE: The element is not visible at runtime.
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

Element prop- Requirement: User management is set up for the visualization. erty 'Access
rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages:
\end{tabular} \\
& \begin{tabular}{l} 
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
\(\bullet\) \\
\\
\hline
\end{tabular} "Rights are set: Limited rights": Access is restricted for at least one group. \\
\hline
\end{tabular}

See also
- \(\boldsymbol{y y}^{\boldsymbol{y}}\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights"" on page 3366

See also
- \(\boldsymbol{y y}^{2}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Potentiometer'}

Symbol:


Category: "Measurement Controls"
The element displays the value of a variable as a setting on the potentiometer. A visualization user can modify the value by dragging the pointer to another position.

See also
- ⓨ Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Element proper- \\ ties}
\begin{tabular}{|l|l|}
\hline "Element name" & Example: GenElemInst_1 \\
\hline "Type of element" & "Potentiometer" \\
\hline "Variable" & \begin{tabular}{l} 
Variable (numeric data type). Contains the position of the pointer for the potenti- \\
ometer. \\
A visualization user can modify the value by dragging the pointer to another \\
position.
\end{tabular} \\
\hline
\end{tabular}

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline\(" Y\) " & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\Psi})\) to other positions in the editor.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The position defines the location and size of the element in the visualization window. These erty 'Position' are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline "X" & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (■-) to other positions in the editor.

See also
- Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

\section*{Element prop- \\ erty 'Back- \\ ground'}
\begin{tabular}{|l|l|}
\hline "Image color" & List box containing background colors \\
\hline "Own image" & \begin{tabular}{l} 
- "Image": ID of the background image. \\
\\
\\
\\
\\
\\
\\
\\
\\
\\
You select the background image from an image pool by clicking \begin{tabular}{l} 
Info. If you specify the value "<default>" or select the image from the \\
"Defautl" category in the Input Assistant, then the original element back- \\
ground image is used. \\
- "Transparency color": Selection from list box or Input Assistant.
\end{tabular} \\
\hline
\end{tabular} \\
\hline
\end{tabular}

\section*{Element prop- \\ erty 'Arrow'}
\begin{tabular}{|c|c|}
\hline "Hand style" & Drop-down list with different arrow types \\
\hline "Color" & \begin{tabular}{l}
- ...: The "Color" dialog box opens. \\
- - Drop-down list with color names
\end{tabular} \\
\hline "Arrow start" & Angle (in degrees) between the left edge of the element and the horizontal axis \\
\hline "Arrow end" & Angle (in degrees) between the right edge of the element and the horizontal axis \\
\hline
\end{tabular}

\section*{Element property 'Scale'}
\begin{tabular}{|l|l|}
\hline "Sub scale position" & \begin{tabular}{l} 
- "Outside": The subscale is displayed on the outer scale ring. ("Frame \\
outside") "Inside": The subscale is displayed on the inner scale ring. ("Frame inside")
\end{tabular} \\
\hline "Scale type" & \begin{tabular}{l} 
Type of scale \\
- "Lines" \\
- "Dots" \\
- "Squares"
\end{tabular} \\
\hline "Scale start" & \begin{tabular}{l} 
Least value of the scale and the lower limit of the value range for the element \\
Example: 0 \\
: The "Variable" property is displayed in the line below this.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (integer data type). Contains the scale start \\
Example: PLC_PRG. iScaleStart \\
Declaration:
\end{tabular} \\
\hline "Scale end" & \begin{tabular}{l} 
PROGRAM PLC_PRG \\
VAR \\
END_VAR iScaleStart : INT : \(\quad 0 ;\)
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Greatest value of the scale and the upper limit of the value range for the element \\
Example: 100 \\
: The "Variable" property is shown below this.
\end{tabular} \\
\hline Variable (integer data type). Contains the scale end \\
Example: PLC_PRG. iScaleEnd \\
Declaration:
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type) Contains the distance between two values on the fine scale \\
Example: PLC_PRG.iSubScale \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iSubScale : INT := 5;
END_VAR

```
\end{tabular} \\
\hline "Scale line width" & \begin{tabular}{l}
Specified in pixels \\
Example: 3
\end{tabular} \\
\hline "Scale color" & \begin{tabular}{l}
Color of scale lines The "Color" dialog opens. \\
- - : A list box with style colors opens
\end{tabular} \\
\hline "Scale in 3D" & Scale lines are displayed with soft 3D shadowing. Note: This property is not displayed in "FlatStyle". \\
\hline "Show scale" & V: The scale is displayed. \\
\hline "Frame inside" & \(\square\) : A frame is drawn at the inner end of the scale. \\
\hline "Frame outside" & V: A frame is drawn at the outer end of the scale. \\
\hline
\end{tabular}

Element prop-
erty 'Label'
\begin{tabular}{|l|l|}
\hline "Label" & \begin{tabular}{l} 
Selection list \\
- "Outside": Scale values are placed outside of the scale. \\
- "Inside": Scale values are placed inside of the scale.
\end{tabular} \\
\hline "Unit" & \begin{tabular}{l} 
Text that is displayed in the element. \\
Example: Units displayed in m/s.
\end{tabular} \\
\hline "Font" & \begin{tabular}{l} 
Font for labels (example: scale numbering). \\
Selection from the drop-down list or by clicking the "" button. ...
\end{tabular} \\
\hline "Scale format (C Syntax)" & \begin{tabular}{l} 
Values scaled in "printf" syntax \\
Examples: \%d, \%5.2f
\end{tabular} \\
\hline "Max. text width of labels" & \begin{tabular}{l} 
(optional) Value that redefines the maximum width of the scale label. The correct \\
value is normally set automatically. \\
Note: Change this value only if the automatic adjustment does not yield the \\
expected result.
\end{tabular} \\
\hline "Text height of labels" & \begin{tabular}{l} 
(optional) Value that redefines the maximum height of the scale label. The cor- \\
rect value is normally set automatically. \\
Note: Change this value only if the automatic adjustment does not yield the \\
expected result.
\end{tabular} \\
\hline "Font color" & Selection from the drop-down list or by clicking the ■... button. \\
\hline
\end{tabular}

\section*{Element property 'Positioning'}
\begin{tabular}{|c|c|}
\hline "Usage of" & \begin{tabular}{l}
- "Preset style values": Values from the current style \\
- "User-defined settings": The subnode "Positioning" appears.
\end{tabular} \\
\hline \begin{tabular}{l}
"Positioning" \\
Requirement: "User- \\
The displayed positio whether a custom ba relative to the backgrour
\end{tabular} & \begin{tabular}{l}
tings" is selected as "Usage of". \\
gs depend on the type of needle instrument and Potentiometer, and partially on mage is selected. The following settings are used for achieving the exact position
\end{tabular} \\
\hline "Needle movement" & Length of the needle (in pixels) \\
\hline "Scale movement" & \begin{tabular}{l}
Distance from the tick marks to the center (in pixels) \\
Requirement: A customer image is selected as "Background".
\end{tabular} \\
\hline "Scale length" & \begin{tabular}{l}
Length of the tick marks (in pixels) \\
Requirement: A customer image is selected as "Background".
\end{tabular} \\
\hline "Label offset": & Distance from the labels to the tick marks (in pixels) \\
\hline "Unit offset": & Distance of the unit text "Label \(\rightarrow\) Unit" from the upper scale edge (in pixels) \\
\hline "Origin offset" & \begin{tabular}{l}
Offset of the element (in pixels) \\
Requirement: For the elements "Meter \(180^{\circ}\) " and "Meter \(90^{\circ}\) ", this property is displayed only if a custom image is selected as "Background".
\end{tabular} \\
\hline
\end{tabular}

\section*{Element prop- \\ erty 'Colors'}
\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|l|}{"Color areas"} \\
\hline "Durable color areas" & All color areas are visible, regardless of the current value.
Only the color area is visible that includes the current value. \\
\hline "Use colors for scale" & 回: Colors in the color area are used only for the scale and frame. \\
\hline "Color areas" & \\
\hline "Create new" & A new color area is added to the "Elements" view. \\
\hline "Delete" & The color area is removed from the list and the list is refreshed. \\
\hline "Begin of area" & \begin{tabular}{l}
Start value of the color area \\
Example: 20 \\
*: The property "Variable" is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the start value. \\
Example: PLC_PRG.iColorAreaStart0 \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iColorAreaStart0 : INT := 80;
END_VAR

```
\end{tabular} \\
\hline "End of area" & \begin{tabular}{l}
End value of the color area \\
Example: 120 \\
: The property "Variable" is shown below.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Variable" & \begin{tabular}{l} 
Variable (integer data type). Contains the end value. \\
Example: iColorAreaEnd0 \\
Declaration: \\
\\
\\
\\
\\
\hline PROGRAM PLC_PRG \\
VAR iColorAreaEndo : INT \(:=100 ;\) \\
END_VAR \(\quad\) Color that is used for displaying the area. \\
\hline
\end{tabular} \\
\hline
\end{tabular}

Element prop- The variables control the element behavior dynamically. erty 'State varia-
bles'
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime. \\
\\
\\
Example: bIsVisible with VAR bIsVisible : BOOL : \(=\) FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" \\
\hline "X" & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\oplus\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
"Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PPRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\boldsymbol{\oplus}\) symbol.
\end{tabular} \\
\begin{tabular}{l} 
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular}
\end{tabular}

You can link the variables to a unit conversion.

The "X", "Y", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

\section*{Element property 'Access rights'}
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages: \\
- "Not set. Full rights.". Access rights for all user groups : "operable" \\
\(\bullet\) \\
- "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\(\qquad\)\begin{tabular}{l} 
See also \\
• « Chapter 6.4.5.21.3.1 "Dialog 'Access Rights"' on page 3366
\end{tabular}
\end{tabular}

\section*{Visualization Element 'Histogram'}

\section*{Symbol:}

\section*{Category: "Measurement Controls"}

The element displays the data of a one-dimensional array as a histogram. You can define specific colors for certain value ranges.
See also
- \({ }^{\leftrightarrows}\) Chapter 6.4.5.2.4 "Displaying Array Data in a Histogram" on page 2862

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & Example: GenElemInst_35 \\
\hline "Type of element" & "Histogram" \\
\hline "Data array" & \begin{tabular}{l} 
One-dimensional array with data displayed in this histogram. \\
Example: PLC_PRG.arr1
\end{tabular} \\
\hline
\end{tabular}

\section*{Element property 'Subrange of array'}
\begin{tabular}{|l|l|}
\hline "Use subrange" & \(\boxed{\square}\) : Only part of the array is displayed in the histogram. \\
\hline "Start index" & \begin{tabular}{l} 
First array index with a displayed value. \\
Requirement: "Use subrange" is activated.
\end{tabular} \\
\hline "End index" & \begin{tabular}{l} 
Last array index with a displayed value. \\
Requirement: "Use subrange" is activated.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Display type" & \begin{tabular}{l} 
- "Bars": Data is displayed as bars. \\
- "Lines": Data is displayed as lines. \\
- "Curve": Interpolation of data into a curve.
\end{tabular} \\
\hline "Line width" & \begin{tabular}{l} 
Specified in pixels \\
Requirement: "Curve" is selected as the "Display type".
\end{tabular} \\
\hline "Show horizontal lines" & \begin{tabular}{l} 
: Horizontal lines are drawn on the main scale. \\
Note: Not all visualization styles have this property. This element property is not \\
available for visualization styles that have striped backgrounds (example: "Flat \\
style").
\end{tabular} \\
\hline "Relative bar width" & \begin{tabular}{l} 
Integer value between 1 and 100 \\
- 1: The bars are drawn as lines. \\
- 100: The entire width of the histogram is filled with the bars.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline "X" & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (■-) to other positions in the editor.

See also
- Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center'
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline\(~ Y "\) & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols \((\boldsymbol{\oplus})\) to other positions in the editor.

Element property 'Scale'
\begin{tabular}{|l|l|}
\hline "Scale start" & \begin{tabular}{l} 
Least value of the scale and the lower limit of the value range for the element. \\
Example: 0 \\
0: The property "Variable" is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (integer data type). Contains the scale start. \\
Example: PLC_PRG. iScaleStart
\end{tabular} \\
\hline "Scale end" & \begin{tabular}{l} 
Greatest value of the scale and the upper limit of the value range for the ele- \\
ment. \\
Example: 100 \\
: The property "Variable" is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (integer data type). Contains the scale end. \\
Example: PLC_PRG.iScaleEnd
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Main scale" & \begin{tabular}{l}
Distance between 2 values on the rough scale. \\
Example: 10 \\
*: The property "Variable" is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the distance. \\
Example: PLC_PRG.iMainScale
\end{tabular} \\
\hline "Subscale" & \begin{tabular}{l}
Distance between 2 values on the fine scale. \\
You can hide the fine scale by setting the value to 0 . \\
Example: 2 \\
p: The property "Variable" is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the spacing. \\
Example: PLC_PRG.iSubScale
\end{tabular} \\
\hline "Scale color" & \begin{tabular}{l}
Color of scale lines \\
- ...: \\
The "Color" dialog box opens. \\
- \(\quad\) : A drop-down list with color names opens.
\end{tabular} \\
\hline "Base line" & \begin{tabular}{l}
Value of the main scale where the horizontal base line of the Histogram is located. \\
The drawing of the bar starts at the base line.
\end{tabular} \\
\hline
\end{tabular}
```

Example A valid declaration is required for the variables used as an example in the table above.
PROGRAM PLC_PRG
VAR
iScaleStart : INT := 0;
iScaleEnd : INT := 120;
iMainScale : INT := 20;
iSubScale : INT := 5;
END_VAR

```

Element prop-
erty 'Label'
\begin{tabular}{|l|l|}
\hline "Unit" & \begin{tabular}{l} 
Text that is displayed in the element. \\
Example: Units displayed in \(\mathrm{m} / \mathrm{s}\).
\end{tabular} \\
\hline "Font" & \begin{tabular}{l} 
Font for labels (example: scale numbering). \\
Selection from the drop-down list or by clicking the "" button....
\end{tabular} \\
\hline "Scale format (C Syntax)" & \begin{tabular}{l} 
Values scaled in "printf" syntax \\
Examples: od, \%5.2f
\end{tabular} \\
\hline "Max. text width of labels" & \begin{tabular}{l} 
Optional value that defines the maximum width of the scale label. \\
Note: Change this value only if the automatic adjustment does not yield the \\
expected result.
\end{tabular} \\
\hline "Text height of labels" & \begin{tabular}{l} 
Optional value that defines the maximum height of the scale label. \\
Note: Change this value only if the automatic adjustment does not yield the \\
expected result.
\end{tabular} \\
\hline "Font color" & Selection from the drop-down list or by clicking the .... button. \\
\hline
\end{tabular}

Element property 'Colors'
\begin{tabular}{|l|l|}
\hline "Graph color" & \begin{tabular}{l} 
Color of the bar in normal state. \\
Note: The normal state is in effect when the current value of the array compo- \\
nent does not fulfill the alarm condition.
\end{tabular} \\
\hline "Alarm value" & Threshold for the alarm \\
\hline "Alarm condition" & \begin{tabular}{l} 
If the current value of the array component fulfills the alarm condition, then the \\
alarm condition is set. \\
- "Less": The current value is less than the "Alarm value" \\
- "More": The current value is greater than the "Alarm value"
\end{tabular} \\
\hline "Alarm color" & Color of the bar in alarm state. \\
\hline "Use color areas" & ": The color areas defined in this element are used. \\
\hline "Color areas" & A new color area is added. \\
\hline "Create new" & The color area is removed from the list. \\
\hline "Delete" & The start value on the "Scale" of the Histogram where the color area begins. \\
\hline "Begin of area" & The end value on the "Scale" of the Histogram where the color area ends. \\
\hline "End of area" & Color that is used for displaying the area. \\
\hline "Color" &
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" & \multicolumn{1}{|l|}{\begin{tabular}{|l|}
\hline Variable (numeric data type). Defines the \(X\) position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular}} \\
\hline "Y" & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(₫\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center"according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
systen of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\boldsymbol{\oplus}\) symbol.
\end{tabular} \\
\begin{tabular}{l} 
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

The " \(X\) ", " \(Y\) ", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also
-

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|l|l|}
\hline "Animation duration" & \begin{tabular}{l} 
Defines the duration (in milliseconds) in which the element runs an animation \\
- \begin{tabular}{l} 
Variable (integer value) \\
Example: Menu. tContent with VAR tContent : INT \(:=500 ;\) \\
END_VAR \\
- Integer literal \\
Example: 500
\end{tabular} \\
Animatable properties \\
- "Absolute movement", "Movement", "X", " \(Y\) " \\
- "Absolute movement", ""otation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable \\
property has changed. The movement then executed is not jerky, but is smooth \\
within the specified animation duration. The visualization element travels to the \\
specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline \begin{tabular}{l} 
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIs InForeground with VAR bIsInForeground : BooL \(:=\) \\
FALSE; END_VAR
\end{tabular} \\
\hline "Move to foreground" \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it \\
was inserted in the visualization editor.
\end{tabular}

Element prop- Requirement: User management is set up for the visualization. erty 'Access
rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages:
\end{tabular} \\
& \begin{tabular}{l} 
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
\(\bullet\) \\
"Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

See also
- ※y Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

\section*{Visualization Element 'Image Switcher'}

Symbol:


\section*{Category: "Lamps/Switches/Bitmaps"}

The element displays one of three referenced images. Mouse actions change the displayed image. The images are defined in the "Image settings" element properties. The effects of mouse clicks are defined in the "Element behavior" property.

\section*{Element proper- \\ ties}
\begin{tabular}{|l|l|}
\hline "Element name"" & \begin{tabular}{l} 
Optional \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list. \\
Example: ImageSwitcher_1
\end{tabular} \\
\hline "Type of element" & "Image Switcher" \\
\hline
\end{tabular}

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X\) " & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (파) to other positions in the editor.

See also
- \& Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874
\begin{tabular}{|l|l|}
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL). \\
The value of the variable changes according to user input and it is independent \\
of the "Element behavior" element property.
\end{tabular} \\
\hline
\end{tabular}

Image settings
\begin{tabular}{|l|l|}
\hline "Image "on"" & \begin{tabular}{l} 
Image ID from an image pool. The image can be selected using the input assis- \\
tant. \\
The image is used if the variable of the "Variable" property has the value TRUE.
\end{tabular} \\
\hline "Image "off"" & \begin{tabular}{l} 
Image ID from an image pool. The image can be selected using the input assis- \\
tant. \\
The image is used if the variable of the "Variable" property has the value FALSE.
\end{tabular} \\
\hline "Image "clicked""" & \begin{tabular}{l} 
Image ID from an image pool. The image is selected using the input assistant. \\
In runtime mode, the visualization displays the referenced image when the ele- \\
ment is clicked (and the mouse button is held down). \\
Requirement: The "Element behavior" is "Image toggler".
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Transparency" & 冋: The "Transparent color" is selected. \\
\hline "Transparent color" & \begin{tabular}{l}
The image pixels that have the transparent color are displayed as transparent. \\
Requirement: "Transparency" is activated. \\
- ...The "Color" dialog box opens. \\
- \(\quad\) : A drop-down list with color names opens.
\end{tabular} \\
\hline "Scaling type" & \begin{tabular}{l}
Defines how an image fits in the element frame. \\
- "Fixed": The original size of the image is retained, regardless of the dimensions of the element. \\
- "Isotropic": The entire image is shown in the element frame, either larger or smaller. As a result, the proportion of height and width are retained. \\
- "Anisotropic": The image resizes automatically to the dimensions of the element frame, filling the entire element frame. As a result, the proportions are not retained.
\end{tabular} \\
\hline "Horizontal alignment" & \begin{tabular}{l}
Horizontal alignment of the image within the element frame or element \\
- Left \\
- Centered \\
- Right \\
Requirement: "Scaling type" is "lsotropic".
\end{tabular} \\
\hline "Vertical alignment" & \begin{tabular}{l}
Vertical alignment of the image within the element frame or element \\
- Top \\
- Centered \\
- Bottom \\
Requirement: "Scaling type" is "lsotropic".
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Element behavior" & \begin{tabular}{l} 
- "Image toggler": Every mouse click switches the image. \\
- "Image tapper"" While a visualization user holds down the mouse button, the \\
image of the "Image on" property is displayed. At the same time, the value \\
TRUE is assigned to the "Variable" property.
\end{tabular} \\
\hline "Tap FALSE" & \begin{tabular}{l} 
V: While the mouse button is pressed, the image of the "Image" property is \\
displayed and the "Variable" property gets the value FALSE instead of the value \\
TRUE, and back. \\
Requirement: "Image tapper" is selected in the "Element behavior" property.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the \(\oplus\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline\(Y "\) & \(Y\)-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus})\) to other positions in the editor.

\section*{Element property 'Texts'}
\begin{tabular}{|l|l|}
\hline "Tooltip" & \begin{tabular}{l} 
String display as tooltip for the element \\
Example: Valid access.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG. iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG. iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\oplus\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
\hline "Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol. \\
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

The " \(X\) ", " \(Y\) ", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

\section*{See also}

Element prop- The variables control the element behavior dynamically. erty 'State variables'
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
\\
\\
\\
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \(=\) EALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- Requirement: User management is set up for the visualization. erty 'Access
rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages:
\end{tabular} \\
& \begin{tabular}{l} 
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
\(\bullet\) \\
"Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{\wedge} \Rightarrow\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

See also
- \(\boldsymbol{y}^{2}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Lamp'}

Symbol:


\section*{Category: "Lamps/Switches/Bitmaps"}

The element shows the value of a variable, and the element is displayed as illuminated or not.

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Optional \\
\\
\\
Hint: Assign individual names for elements so that they are found faster in the \\
element list. \\
Example: Lamp_green
\end{tabular} \\
\hline "Type of element" & "Lamp" \\
\hline
\end{tabular}

Element prop- The position defines the location and size of the element in the visualization window. These erty 'Position' are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X\) " & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (무-) to other positions in the editor.

See also
- \({ }^{*}>\) Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874
\begin{tabular}{|l|l|}
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL). \\
The variable value is displayed as a lamp that goes on (TRUE) or off (FALSE).
\end{tabular}.
\end{tabular}

Image settings
\begin{tabular}{|c|c|}
\hline "Transparency" & \(\checkmark\) : The "Transparent color" property is selected. \\
\hline "Transparent color" & \begin{tabular}{l}
Pixels in this color are displayed as transparent. \\
Requirement: "Transparency" is activated. \\
- ... The "Color" dialog box opens. \\
- - A drop-down list with style colors opens.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Scaling type" & \begin{tabular}{l}
Reaction of the element when the dimension of the "Frame" element is changed: \\
- "/sotropic": The height and width of the image are resized proportionally to the "Frame". \\
Please note: To retain the alignment of elements also within a scaled "Frame" element, define the "Horizontal alignment" or "Vertical alignment" explicitly with "Centered". \\
- "Anisotropic": The image fills the entire "Frame" regardless of its proportions.
\end{tabular} \\
\hline "Horizontal alignment" & \begin{tabular}{l}
Horizontal alignment of the image within the element frame or element \\
- Left \\
- Centered \\
- Right \\
Requirement: "Scaling type" is "lsotropic".
\end{tabular} \\
\hline "Vertical alignment" & \begin{tabular}{l}
Vertical alignment of the image within the element frame or element \\
- Top \\
- Centered \\
- Bottom \\
Requirement: "Scaling type" is "Isotropic".
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center' rotation is shown as the \(\oplus\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline " \(Y\) " & \(Y\)-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols (ఉ) to other positions in the editor.

Element prop-
erty 'Texts'
\begin{tabular}{|l|l|}
\hline "Tooltip" & \begin{tabular}{l} 
String display as tooltip for the element \\
Example: Valid access.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline \multicolumn{1}{|l|}{ "Movement" } \\
\hline " \(X "\) & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline\(" Y "\) & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\oplus\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
"Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PPRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\boldsymbol{\oplus}\) symbol.
\end{tabular} \\
\begin{tabular}{l} 
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular}
\end{tabular}

You can link the variables to a unit conversion.

The "X", "Y", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

\section*{See also}

Element prop- The variables control the element behavior dynamically.
erty 'State varia-
bles'
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
\\
\\
\end{tabular} \(\mathrm{TRUE}:\) The element is not visible at runtime.
\end{tabular}


The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop-
erty 'Back-
ground'
\begin{tabular}{|l|l|}
\hline "Image" & \begin{tabular}{l} 
Drop-down list with background colors \\
Depends on the visualization style
\end{tabular} \\
\hline
\end{tabular}

Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages: \\
- "Not set. Full rights.". Access rights for all user groups : "operable" \\
\\
- "Rights are set: Limited rights": Access is restricted for at least one group. \\
\hline
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\xi^{3}\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

See also
- 乡y Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Dip Switch', 'Power Switch', 'Push Switch', 'Push Switch LED', 'Rocker Switch' \\ Symbols: \\  \\ Category: "Lamps/Switches/Bitmaps" \\ The element assigns a value to a Boolean variable. The switch position "on" the value TRUE to the variable, and the switch position "off" assigns the value FALSE. Use the mouse to change the switch position.}

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Example: Operating_Switch \\
Optional
\end{tabular} \\
\begin{tabular}{l} 
Hint: Assign individual names for elements so that they are found faster in the \\
element list.
\end{tabular} \\
\hline "Type of element" & \begin{tabular}{l} 
Depending on the element: "Dip Switch", "Power Switch", "Push Switch", "Push \\
Switch LED", or "Rocker Switch"
\end{tabular} \\
\hline
\end{tabular}

Element prop- The position defines the location and size of the element in the visualization window. These erty 'Position' are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline "X" & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}


You can also change the values by dragging the box symbols ( \({ }^{(-)}\)) to other positions in the editor.

See also
- Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874
\begin{tabular}{|l|l|}
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL) \\
The value of the variables TRUE and FALSE indicates the switch position on/off.
\end{tabular} \\
\hline
\end{tabular}

\section*{Image settings}
\begin{tabular}{|c|c|}
\hline "Transparency" & V: The "Transparent color" property is selected. \\
\hline "Transparent color" & \begin{tabular}{l}
Pixels in this color are displayed as transparent. \\
Requirement: "Transparency" is activated. \\
- ...The "Color" dialog box opens. \\
- \(\quad\) : A drop-down list with style colors opens.
\end{tabular} \\
\hline "Scaling type" & \begin{tabular}{l}
Reaction of the element when the dimension of the "Frame" element is changed: \\
- "Isotropic": The height and width of the image are resized proportionally to the "Frame". \\
Please note: To retain the alignment of elements also within a scaled "Frame" element, define the "Horizontal alignment" or "Vertical alignment" explicitly with "Centered". \\
- "Anisotropic": The image fills the entire "Frame" regardless of its proportions.
\end{tabular} \\
\hline "Horizontal alignment" & \begin{tabular}{l}
Horizontal alignment of the image within the element frame or element \\
- Left \\
- Centered \\
- Right \\
Requirement: "Scaling type" is "Isotropic".
\end{tabular} \\
\hline "Vertical alignment" & \begin{tabular}{l}
Vertical alignment of the image within the element frame or element \\
- Top \\
- Centered \\
- Bottom \\
Requirement: "Scaling type" is "lsotropic".
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Element behavior" & \begin{tabular}{l} 
- "Image toggler": Every mouse click changes the switch and the "Variable" \\
- value. \\
"Image tapper": The switch is "on" and the "Variable" value is TRUE while the \\
mouse button is pressed.
\end{tabular} \\
\hline "Tap FALSE" & \begin{tabular}{l} 
V: The value TRUE is assigned to the "Variable" property instead of the value \\
FALSE, and back. \\
Requirement: "Image tapper" is selected in the "Element behavior" property.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline " \(Y\) " & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus}\) ) to other positions in the editor.

\section*{Element prop- \\ erty 'Texts'}
\begin{tabular}{|l|l|}
\hline "Tooltip" & \begin{tabular}{l} 
String display as tooltip for the element \\
Example: Valid access.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG. iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG. iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\oplus\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
\hline "Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol. \\
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
\hline
\end{tabular}

You can link the variables to a unit conversion.


\section*{See also}

Element prop- The variables control the element behavior dynamically. erty 'State variables'
\begin{tabular}{|l|l|}
\hline "Invisible" & Variable (BOOL). Toggles the visibility of the element. \\
& \begin{tabular}{l} 
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|l|l|}
\hline "Animation duration" & \begin{tabular}{l} 
Defines the duration (in milliseconds) in which the element runs an animation \\
- \begin{tabular}{l} 
Variable (integer value) \\
Example: Menu. tContent with VAR tContent : INT \(:=500 ;\) \\
END_VAR \\
- Integer literal \\
Example: 500
\end{tabular} \\
Animatable properties \\
- "Absolute movement", "Movement", "X", " \(Y\) " \\
- "Absolute movement", ""otation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable \\
property has changed. The movement then executed is not jerky, but is smooth \\
within the specified animation duration. The visualization element travels to the \\
specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline \begin{tabular}{l} 
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIs InForeground with VAR bIsInForeground : BooL \(:=\) \\
FALSE; END_VAR
\end{tabular} \\
\hline "Move to foreground" \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it \\
was inserted in the visualization editor.
\end{tabular}

Element prop-
erty 'Back-
ground'
\begin{tabular}{|l|l|}
\hline "Image" & \begin{tabular}{l} 
Drop-down list with background colors \\
Depends on the visualization style
\end{tabular} \\
\hline
\end{tabular}

Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages:
\end{tabular} \\
& \begin{tabular}{l}
\(\bullet\) "Not set. Full rights.": Access rights for all user groups : "operable" \\
\(\bullet\) \\
\hline "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

See also
- ② Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

See also
- \(\boldsymbol{y y}^{4}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Rotary Switch'}

Symbol:

\section*{Category: "Lamps/Switches/Bitmaps"}

The element assigns a value to a Boolean variable. The switch position "on" the value TRUE to the variable, and the switch position "off" assigns the value FALSE. Use the mouse to change the switch position.

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Example: Operating_Switch \\
Optional \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list.
\end{tabular} \\
\hline "Type of element" & "Rotary Switch" \\
\hline
\end{tabular}

\section*{Element property 'Position'}

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X\) " & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (匝) to other positions in the editor.

See also
- \(\quad \stackrel{y}{ }\) Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874
\begin{tabular}{|l|l|}
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL). \\
The value of the variables TRUE and FALSE indicates the switch position on/off.
\end{tabular} \\
\hline
\end{tabular}

\section*{Image settings}
\begin{tabular}{|c|c|}
\hline "Transparency" & \(\checkmark\) : The "Transparent color" property is selected. \\
\hline "Transparent color" & \begin{tabular}{l}
Pixels in this color are displayed as transparent. \\
Requirement: "Transparency" is activated. \\
- ...The "Color" dialog box opens. \\
- \(\mathbf{~ - ~ : ~ A ~ d r o p - d o w n ~ l i s t ~ w i t h ~ s t y l e ~ c o l o r s ~ o p e n s . ~}\)
\end{tabular} \\
\hline "Scaling type" & \begin{tabular}{l}
Reaction of the element when the dimension of the "Frame" element is changed: \\
- "Isotropic": The height and width of the image are resized proportionally to the "Frame". \\
Please note: To retain the alignment of elements also within a scaled "Frame" element, define the "Horizontal alignment" or "Vertical alignment" explicitly with "Centered". \\
- "Anisotropic": The image fills the entire "Frame" regardless of its proportions.
\end{tabular} \\
\hline "Horizontal alignment" & \begin{tabular}{l}
Horizontal alignment of the image within the element frame or element \\
- Left \\
- Centered \\
- Right \\
Requirement: "Scaling type" is "lsotropic".
\end{tabular} \\
\hline "Vertical alignment" & \begin{tabular}{l}
Vertical alignment of the image within the element frame or element \\
- Top \\
- Centered \\
- Bottom \\
Requirement: "Scaling type" is "/sotropic".
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Element behavior" & \begin{tabular}{l} 
- "Image toggler": Every mouse click changes the switch and the "Variable" \\
- value. \\
"Image tapper": The switch is "on" and the "Variable" value is TRUE while the \\
mouse button is pressed.
\end{tabular} \\
\hline "Orientation" & \begin{tabular}{l} 
- "At top": The rotary switch turns from the top right to the top left. \\
- "At side": The rotary switch turns from the top right to the bottom right.
\end{tabular} \\
\hline "Color change" & ص: The element changes in color when "Variable" is TRUE. \\
\hline
\end{tabular}

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center' rotation is shown as the \(\oplus\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline " \(Y\) " & Y-coordinate of the point of rotation \\
\hline
\end{tabular}


\section*{Element property 'Texts'}
\begin{tabular}{|l|l|}
\hline "Tooltip" & \begin{tabular}{l} 
String display as tooltip for the element \\
Example: Valid access.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG. iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG. iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\oplus\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
\hline "Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol. \\
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

The " \(X\) ", " \(Y\) ", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

\section*{See also}

Element prop- The variables control the element behavior dynamically. erty 'State variables'
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
\\
\\
\\
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \(=\) EALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

\section*{Element prop-}
erty 'Back-
ground'
\begin{tabular}{|l|l|}
\hline "Image" & \begin{tabular}{l} 
Drop-down list with background colors \\
Depends on the visualization style
\end{tabular} \\
\hline
\end{tabular}

Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages: \\
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
\\
- "Rights are set: Limited rights": Access is restricted for at least one group. \\
\hline
\end{tabular} \\
\hline
\end{tabular}

See also
- ② Chapter 6.4.5.21.3.1 "Dialog 'Access Rights"" on page 3366

See also
- \({ }^{4}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Trace'}

Symbol:


\section*{Category: "Special Controls"}

The element displays the graphical curve of variable values. In addition, variables can be configured to control the view.

See also
- \({ }^{4}\) Chapter 6.4.5.12 "Displaying data curve with trace" on page 2925
- そy "Dialog box 'Trace Configuration"' on page 3355

\section*{Element proper- \\ ties}
\begin{tabular}{|c|c|}
\hline "Element name" & Example: Velocity \\
\hline "Data Source" & \begin{tabular}{l}
Location where the trace data is buffered. \\
- "<local application>": The trace record is listed below the local application. The visualization that contains the trace is located below this application. When the application is downloaded, the trace configuration is downloaded to the local device. During execution, the data is stored locally in the trace buffer. \\
- " <data source name>": Data source that identifies the remote device where the trace record is created. \\
When the local application is downloaded with the visualization, the trace configuration is downloaded to the remote device. During execution, the trace buffer is filled, and the trace data is transferred and then displayed in the local visualization as HMI. \\
Example: DataSoure_PLC_A \\
Note: The trace buffer is filled only if the remote application is being executed. The data recording is started when the local visualization is started.
\end{tabular} \\
\hline "Application" & \begin{tabular}{l}
Application where data was recorded. \\
Lists all applications that are present below the data source. \\
Requirement: A remote data source (not "<local application>") is referenced in the "Data source" property.
\end{tabular} \\
\hline "Type of element" & "Trace" \\
\hline "Trace" & Q" <name of trace configuration>": Opens the "Trace Configuration" dialog where you can modify the trace configuration. \\
\hline
\end{tabular}

See also
- ② "Dialog box 'Trace Configuration'" on page 3355
- Data Source Manager

Element prop-
erty 'Position'

The position defines the location and size of the element in the visualization window. This is based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|c|c|}
\hline " \(X\) " & \begin{tabular}{l}
The x-coordinate of the upper left corner of the element Specified in pixels \\
Example: 10
\end{tabular} \\
\hline "Y" & \begin{tabular}{l}
The y-coordinate of the upper left corner of the element Specified in pixels \\
Example: 10
\end{tabular} \\
\hline "Width" & \begin{tabular}{l}
Specified in pixels \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l}
Specified in pixels \\
Example: 30
\end{tabular} \\
\hline & Tip: You can change the values in " \(X\) ", " \(Y\) ", "Width", and "Height" by dragging the corresponding symbols 龺- to another position in the editor. \\
\hline "Angle" & \begin{tabular}{l}
Static angle of rotation (in degrees) \\
Example: 35 \\
The element is displayed rotated in the editor. The point of rotation is the center of the element. A positive value rotates clockwise. \\
Tip: You can change the value in the editor by focusing the element to the handle. When the cursor is displayed as a rotating arrow \(\mathcal{A}\), you can rotate the element about its center as a handle. \\
(1): Handle \\
Note: If a dynamic angle of rotation is also configured in the property "Absolute movement \(\rightarrow\) Internal rotation", then the static and dynamic angles of rotation are added in runtime mode. The static angle of rotation acts as an offset.
\end{tabular} \\
\hline
\end{tabular}

See also
- ② Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874
\begin{tabular}{|l|l|}
\hline "Show cursor" & \begin{tabular}{l} 
V: A cursor (vertical line) is displayed at the mouse position. The trigger and \\
variable values where the cursor points are displayed as a tooltip.
\end{tabular} \\
\hline "Overwrite existing trace on & \begin{tabular}{l} 
V: If a trace with the same name is on the PLC, then it is overwritten at down- \\
PLC"
\end{tabular} \\
\hline "Number format" with the configuration that is defined here.
\end{tabular}

Element prop- \(\quad\) The control variables are assigned automatically when you click "Insert elements for controlling erty 'Control Trace". variables'
\begin{tabular}{|l|l|}
\hline "Reset Trigger" & \begin{tabular}{l} 
Variable (BOOL). \\
Standard control variable: bResetTrigger \\
TRUE: Resets the triggering. After the action is executed, the variable is set \\
automatically to FALSE.
\end{tabular} \\
\hline "Start Trace" & \begin{tabular}{l} 
Variable (BOOL). \\
Standard control variable: bStart \\
TRUE: Starts the Trace. After the action is executed, the variable is set automati- \\
cally to FALSE.
\end{tabular} \\
\hline "Stop Trace" & \begin{tabular}{l} 
Variable (BOOL). \\
Standard control variable: bStop
\end{tabular} \\
\hline "Save Trace to a file" & \begin{tabular}{l} 
TRUE: Stops the Trace. After the action is executed, the variable is set automati- \\
cally to FALSE.
\end{tabular} \\
\hline "Save Trace" & \begin{tabular}{l} 
Variable (BOOL). \\
Standard control variable: bStore
\end{tabular} \\
\hline "File name" & \begin{tabular}{l} 
TRUE: Saves the current trace configuration and the data that is stored in the \\
development system to a file. When the action is ended, the variable is set \\
automatically to FALSE.
\end{tabular} \\
\hline "File name" & \begin{tabular}{l} 
Variable (STRING) that contains the file name of the file to be saved. \\
Standard control variable: sStoreFilename
\end{tabular} \\
\hline Standard variable: sRestoreFilename
\end{tabular}

See also
- \({ }^{*}\) Chapter 6.4.5.21.2.15 "Command 'Insert Elements for Controlling Trace"" on page 3358

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline\(" X "\) & X-coordinate of the point of rotation \\
\hline\(" Y "\) & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus}\) ) to other positions in the editor.

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG. iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG.iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\oplus\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
"Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol. \\
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

The "X", " \(Y\) ", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also
-

Element prop- The variables control the element behavior dynamically. erty 'State variables'
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- Requirement: User management is set up for the visualization. erty 'Access
rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages:
\end{tabular} \\
& \begin{tabular}{l} 
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
\(\bullet\) \\
"Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\quad \stackrel{y}{c}\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

\section*{Visualization Element 'Trend'}

Symbol:

\section*{Category: "Special Controls"}

The element displays the curve of variable values as a trend diagram. The trend diagram is suitable for representing a long-term data curve because the data is read from a trend recording and hence from a database. Moreover, you can run the "Trend" element together with the "Date Range Picker", "Legend", and "Time Range Picker" operating elements so that the user can navigate conveniently in the diagram.

You can programmatically delete the recorded trend curve at runtime. The recording starts again from the time of deletion. See the help page for "Programming a Trend Visualization".

\section*{Element proper-}
ties
\begin{tabular}{|c|c|}
\hline "Element name" & Example: Velocity \\
\hline "Data source" & \begin{tabular}{l}
Data source for the connection via the device and the application to the "Trend Recording" object where the trend data that you want to show was saved. \\
If the "Trend Recording" object is on the local device, then it is sufficient when you specify the respective application. If the trend recording is on a remote device, then you need to specify the data source connection to this device. \\
- " \lllocal application>" \\
The "Trend Recording" object is located on the local device in the local application. \\
- <device name> . <application name> \\
Example: Device_A.App_A \\
The "Trend Recording" object is located on the local device Device_A below the application App_A. \\
- \(\mathrm{F}^{\mathrm{m}}\) <data source name> \\
Example: DataSource_B \\
The "Trend Recording" object is located on a remote device that is connected via the data source DataSource_B. Below the (now visible) "Application" property, the remote application is displayed as configured in the data source. \\
Example: App_B \\
Note: If the data source is accessed symbolically by means of a symbol file (CODESYS symbolic), then the required symbol file and the corresponding project have to be saved in the same folder.
\end{tabular} \\
\hline "Type of element" & "Trend" \\
\hline "Trend recording" & \begin{tabular}{l}
: Trend recording whose data is displayed as a diagram \\
The trend recording is located on the device specified in the "Data source" property.
\end{tabular} \\
\hline "Display Mode" & L: Opens the "Display Settings" dialog. \\
\hline
\end{tabular}

See also
- \(\#\) Chapter 6.4.5.13 "Displaying data curve with trend" on page 2928
- \(\triangleq\) Chapter 6.4.5.21.2.12 "Command 'Configure Display Settings of Trend"' on page 3353
- Object 'Data Source'

Element property 'Position'

The position defines the location and size of the element in the visualization window. This is based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|c|c|}
\hline " \(X\) " & \begin{tabular}{l}
The x-coordinate of the upper left corner of the element Specified in pixels \\
Example: 10
\end{tabular} \\
\hline "Y" & \begin{tabular}{l}
The y-coordinate of the upper left corner of the element Specified in pixels \\
Example: 10
\end{tabular} \\
\hline "Width" & \begin{tabular}{l}
Specified in pixels \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l}
Specified in pixels \\
Example: 30
\end{tabular} \\
\hline & Tip: You can change the values in " \(X\) ", " \(Y\) ", "Width", and "Height" by dragging the corresponding symbols \\
\hline "Angle" & \begin{tabular}{l}
Static angle of rotation (in degrees) \\
Example: 35 \\
The element is displayed rotated in the editor. The point of rotation is the center of the element. A positive value rotates clockwise. \\
Tip: You can change the value in the editor by focusing the element to the handle. When the cursor is displayed as a rotating arrow \(\mathcal{A}\), you can rotate the element about its center as a handle. \\
(1): Handle \\
Note: If a dynamic angle of rotation is also configured in the property "Absolute movement \(\rightarrow\) Internal rotation", then the static and dynamic angles of rotation are added in runtime mode. The static angle of rotation acts as an offset.
\end{tabular} \\
\hline
\end{tabular}

See also
- ※ Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874
\begin{tabular}{|c|c|}
\hline \multirow[t]{3}{*}{"Show cursor"} & : A cursor (black triangle with vertical line) is shown in the trend diagram. Behavior at runtime: As soon as the graph is drawn, the user can move the cursor along the time axis in order to mark a specific time. Then the variable value belonging to the cursor position is displayed in the legend above the graph. \\
\hline &  \\
\hline &  \\
\hline "Show tooltip" & \begin{tabular}{l}
Requirement: "Show cursor" is activated. \\
: A tooltip opens at the cursor. \\
Behavior at runtime: The variable value belonging to the cursor position is displayed as a tooltip.
\end{tabular} \\
\hline "Show frame" & - The trend diagram is drawn with a frame. \\
\hline "Number format" & \begin{tabular}{l}
Format specification in printf syntax, which determines how the values are displayed in the tooltip and in the legend \\
Example: \%d (integer variable) or \(\% 5.2 \mathrm{f}\) (floating-point number)
\end{tabular} \\
\hline
\end{tabular}

\section*{Element property 'Tick mark labels'}

The time stored in the trend recording are in the UTC time zone. If the time is displayed in the trend of the visualization element, then the time stamps are converted to the local time zone of the operating system of the PLC.

Change the time zone in the operating system if the times in the trend diagram are not in the zone that you need.
\begin{tabular}{|c|c|}
\hline "Time stamps" & \begin{tabular}{l}
X-value of the trend diagram \\
- "Absolute time stamps" \\
The absolute time with date and time is displayed at each tick mark on the time axis. \\
Example: 03/18/2016 12h30m50s \\
- "Relative time stamps" \\
The time period from the start of the recording \((=0)\) is displayed at each tick mark. \\
Example: 5m30s
\end{tabular} \\
\hline "Draw labels on two lines" & The time stamps are displayed on two lines (for example, the date is displayed on the first line and the time on the second line).
The time stamp is displayed on one line. Example: 2019-11-01-12:30:50. \\
\hline " Omit irrelevant information in timestamps" & The time stamps are displayed in a truncated form (without insignificant information). For example, the date is displayed at the first tick mark, and only the time is displayed at the following tick marks. The "Internationalization (format strings)" property is not visible and is ignored.
\(\square\) The time stamps are displayed with all information. This takes into consideration the "Internationalization (format strings)" property which contains the format specification for the date and time display. \\
\hline "Internationalization (format strings)" & \begin{tabular}{l}
Format specification for the date and time display of the time stamp (when it is displayed in full) \\
Note: The property is visible only if the "Omit irrelevant information in timestamps" option is not selected.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Date" & \begin{tabular}{l}
Format string that returns the date display according to the defined format. The operating system locale is used as the default setting. \\
Defined format strings for the date: \\
- Year: yyyy, yy, y \\
- Month: MM, M \\
- Day: dd, d \\
- Recommended separator: - . / \\
Example: \\
yyyy-MM-d displays 2019-10-25 \\
Yyyy-MM-dd displays 2019-10-25 \\
dd.MM. yyyy displays 25.10. 2019 \\
dd/MM/yyyy displays 25/10/2019
\end{tabular} \\
\hline "Time" & \begin{tabular}{l}
Format string that returns the time (or time of day) display according to the defined format. The operating system locale is used as the default setting. \\
Defined format strings for the time: \\
- 24-hour time definition: HH, H \\
- 12-hour time definition: hh, h \\
- AM/PM for 12-hour time definition: \(t t\) \\
- Minutes: mm, m \\
- Seconds: ss, s \\
- Milliseconds: ms \\
- Microseconds: us \\
- Recommended separator: : or space character \\
Example: \\
HH:mm:ss:ms displays \(15: 30: 59: 123\) \\
\(\mathrm{h}: \mathrm{mm}\) :ss tt displays 3:30:59 PM
\end{tabular} \\
\hline
\end{tabular}

\section*{See also}
- \({ }^{4}\) Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329

Element property 'Assigned control elements'

These elements are created automatically when the control elements are added with the command "Insert elements for controlling Trend".
\begin{tabular}{|l|l|}
\hline "Date Range Picker" & \begin{tabular}{l} 
Control element for changing the date and time of the displayed data sets. \\
With [..., all elements are provided that have implemented the interface \\
IDateRangeSelector. By default, instances of the "Date Range Picker" visu- \\
alization element are available.
\end{tabular} \\
\hline "Time Range Picker" & \begin{tabular}{l} 
Control element for changing the time of the displayed data sets. With \(\ldots .\). , all \\
elements are provided that have implemented the interface ITimeSelector. \\
By default, instances of the "Time Range Picker" visualization element are avail- \\
able.
\end{tabular} \\
\hline "Legend" & \begin{tabular}{l} 
Control element for displaying a legend for the graphs. With \\
provided that have implemented the interface ILegendDisplayer.
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{\text {n }} \gg\) Chapter 6.4.5.21.2.18 "Command 'Insert Elements for Controlling the Trend'" on page 3360

Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline\(" Y "\) & \(Y\)-coordinate of the point of rotation \\
\hline
\end{tabular}

```

You can also change the values by dragging the symbols ( $\boldsymbol{\oplus}$ ) to other positions in the editor.

```

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|c|c|c|}
\hline \multicolumn{3}{|l|}{"Movement"} \\
\hline "X" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the \(X\) position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular}} \\
\hline "Y" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular}} \\
\hline "Rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle1. \\
The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol. \\
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
\end{tabular} &  \\
\hline "Interior rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle2. \\
In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol. \\
Note: If a static angle of rotation is specified in the "Position \(\rightarrow\) Angle" property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.
\end{tabular} &  \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

The "X", " \(Y\) ", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also
-

Element prop- The variables control the element behavior dynamically. erty 'State variables'
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- Requirement: User management is set up for the visualization. erty 'Access
rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages:
\end{tabular} \\
& \begin{tabular}{l} 
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
• "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

See also
- ⓨ Chapter 6.4.5.21.3.1 "Dialog 'Access Rights"" on page 3366

See also
- « Chapter 6.4.5.13 "Displaying data curve with trend" on page 2928
- \({ }^{4}\) Chapter 6.4.5.13.2 "Getting Started with Trend Visualization" on page 2928
- \& Chapter 6.4.5.13.3 "Programming a Trend Visualization" on page 2931
- Object 'Trend Recording'

\section*{Visualization Element 'Legend'}

Symbol:

Category: "Special Controls"

The element is used as a legend for another element (for example, a trend). The legend is assigned in the properties of the other element.
See also
- \({ }^{4}\) Chapter 6.4.5.13 "Displaying data curve with trend" on page 2928

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Example: LegendOfTrendA \\
Optional \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list.
\end{tabular} \\
\hline "Type of element" & "Legend" \\
\hline
\end{tabular}

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline "X" & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (吊) to other positions in the editor.

\section*{See also}
- \({ }^{4}\) Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center' rotation is shown as the \(\oplus\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline "Y" & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\mathbf{\oplus})\) to other positions in the editor.
\begin{tabular}{|l|l|}
\hline "Orientation" & \begin{tabular}{l} 
Orientation of the element. The value is configured in the assigned element. \\
- "Horizontal" \\
- "Vertical"
\end{tabular} \\
\hline "Attached element instance" & Example: Element_A \\
\hline "Show frame" & : The element is displayed with frames. \\
\hline "Number format" & The format of the value in printf syntax (example: \%d, \%5.2f) \\
\hline
\end{tabular}

\section*{Element Prop- \\ erty 'Layout'}

Defines how many variables can be displayed at a maximum and is calculated from the row and column number.
\begin{tabular}{|l|l|}
\hline "Max. number of rows" & Example: 3 \\
\hline "Max. number of columns" & Example: 2 \\
\hline
\end{tabular}

Element Prop- The property affects the text configured in the associated element. erty 'Text properties'
\begin{tabular}{|l|l|}
\hline "Text format" & \begin{tabular}{l} 
"Default": The text will be cut and displayed in only the part that fits into the \\
visualization element. \\
"Linebreak": The text will be wrapped in rows. \\
"Ellipsis": The text is cut and ellipsis . . . are added to indicate that something is \\
missing.
\end{tabular} \\
\hline "Font" & \begin{tabular}{l} 
Font of the text. The entries of the selection list are defined in the visualization \\
style.
\end{tabular} \\
\hline "Font color" & \begin{tabular}{l} 
Text color, for example Grey. The entries of the selection list are defined in the \\
visualization style.
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l} 
Transparency value (255 to 0), which defines the transparency of the corre- \\
sponding color. \\
Example: 255 : The color is opaque. 0: The color is fully transparent.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" \\
\hline "X" & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline " \(Y\) " & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG. iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\oplus\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
\hline "Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol. \\
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

The " \(X\) ", " \(Y\) ", "Rotation", and "Interior rotation" properties are supported by the
"Client Animation" functionality.

See also

\section*{-}

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- Requirement: User management is set up for the visualization. erty 'Access
rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages:
\end{tabular} \\
& \begin{tabular}{l} 
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
\(\bullet\) \\
"Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{\wedge} \Rightarrow\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

\section*{Visualization Element 'ActiveX'}

Symbol:


\section*{Category: "Special Controls"}

The element is used to link an existing ActiveX control in the visualization. You can configure the method calls and their parameters in the element properties of the "ActiveX" element.

\section*{Element proper- \\ ties}
\begin{tabular}{|l|l|}
\hline "Element name" & Example: GenElemInst_1 \\
\hline "Type of element" & "ActiveX" \\
\hline "Element" & \begin{tabular}{l} 
Installed ActiveX component that is linked to the visualization. \\
Hint: To avoid typing errors, select the required ActiveX component by means of \\
the Input Assistant.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X "\) & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (匝-) to other positions in the editor.

\section*{See also}
- \#y Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline "X" & X-coordinate of the point of rotation \\
\hline "Y" & Y-coordinate of the point of rotation \\
\hline
\end{tabular}
\(\square\)

\footnotetext{
Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
}
\begin{tabular}{|c|c|c|}
\hline \multicolumn{3}{|l|}{"Movement"} \\
\hline "X" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular}} \\
\hline "Y" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the \(Y\) position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular}} \\
\hline "Rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle1. \\
The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol. \\
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
\end{tabular} &  \\
\hline "Scaling" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (integer data type). Causes centric stretching. \\
Example: PLC_PRG.iScaling. \\
The reference point is the "Center" property. \\
The value 1 shrinks the element by a factor of 0.001 . The value 1000 returns the element to its original size.
\end{tabular}} \\
\hline "Interior rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle2. \\
In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol. \\
Note: If a static angle of rotation is specified in the property "Position \(\rightarrow\) Angle", then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.
\end{tabular} &  \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

See also
erty 'State varia-
bles'
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime.
\end{tabular} \\
\hline
\end{tabular}


Element prop- These method calls are executed during initialization. They are executed in the first cycle only. erty 'Initial calls'
\begin{tabular}{|l|l|}
\hline "Method calls" " & \begin{tabular}{l} 
Button '"Create new" \\
Creates a subnode below "Methods" with parameters for the method call.
\end{tabular} \\
\hline "Methods" & \begin{tabular}{l} 
" \(<\) number \(>\) "" \\
\\
- "Method": Name of the method \\
- "Parameter": Parameter passed at the method call \\
\\
\\
- "Result parameter": Optional variable for the return value of the method
\end{tabular} \\
\hline
\end{tabular}

Element prop- These method calls are executed in every cycle. They are executed in the refresh rate of the erty 'Cyclic visualization. calls'
\begin{tabular}{|l|l|}
\hline "Method calls" & \begin{tabular}{l} 
Button '"Create new" \\
\\
Creates a subnode below "Methods" for a method call and its parameters.
\end{tabular} \\
\hline "Methods" & " \(\ll\) number \(>\) ]" \\
& - "Method": Name of the method \\
& - "Parameter": Parameter passed at the method call \\
& - "Result parameter": Optional variable for the return value of the method \\
\hline
\end{tabular}

Element prop- These method calls are executed in the refresh rate of the visualization. You define the call erty 'Conditional condition in the property "Methods \(\rightarrow\) [<number \(>] \rightarrow\) Call condition". calls'
\begin{tabular}{|c|c|}
\hline "Method calls" & \begin{tabular}{l}
Button '"Create new" \\
Creates a subnode below "Methods" with a call condition and parameters for the method call.
\end{tabular} \\
\hline "Methods" & \begin{tabular}{l}
"[<number>]" \\
- "Method": Name of the method \\
- "Call condition": Variable (BOOL). A rising edge of this variable triggers the call of this method. \\
- "Parameter": Parameter passed at the method call \\
- "Result parameter": Optional variable for the return value of the method
\end{tabular} \\
\hline
\end{tabular}

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages:
\end{tabular} \\
& \begin{tabular}{l} 
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
\(\bullet \quad\) "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

See also
- ※ Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

\section*{Visualization Element 'Web Browser'}

Symbol:


\section*{Category: "Special Controls"}

The element shows a website, PDF file, or video that has a URL.

\section*{NOTICE!}

The display options of the "Web Browser" element depend on the operating system and the display variant of the visualization.

Requirement: The software components of the web browser are available in the runtime and configured accordingly (example: videos to be shown on Linux).

See also
- Chapter 6.4.5.2.6 "Displaying Web Contents" on page 2865

\section*{Element proper- \\ ties}
\begin{tabular}{|l|l|}
\hline "Element name" & Example: GenElemInst_59 \\
\hline "Type of element" & "Web Browser" \\
\hline
\end{tabular}

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X "\) & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline " \(Y\) " & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols ( ( (T) \(^{-}\)) to other positions in the editor.

See also
- * Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the \(\oplus\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X "\) & X-coordinate of the point of rotation \\
\hline\(Y "\) & \(Y\)-coordinate of the point of rotation \\
\hline
\end{tabular}


Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|c|c|c|}
\hline \multicolumn{3}{|l|}{"Movement"} \\
\hline " \(X\) " & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular}} \\
\hline " \(Y\) " & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular}} \\
\hline "Rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle1. \\
The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol. \\
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
\end{tabular} &  \\
\hline "Scaling" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (integer data type). Causes centric stretching. \\
Example: PLC_PRG.iScaling. \\
The reference point is the "Center" property. \\
The value 1 shrinks the element by a factor of 0.001 . The value 1000 returns the element to its original size.
\end{tabular}} \\
\hline "Interior rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle2. \\
In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. \\
The rotation point is shown as the \(\boldsymbol{\oplus}\) symbol. \\
Note: If a static angle of rotation is specified in the property "Position \(\rightarrow\) Angle", then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.
\end{tabular} &  \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

Element prop- The variables control the element behavior dynamically. erty 'State variables'
\begin{tabular}{|l|l}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

Element prop-
erty 'Control
variables'

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages:
\end{tabular} \\
& \begin{tabular}{l} 
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
\(\bullet\) \\
"Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{3}\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

\section*{Visualization Element 'Busy Symbol, Cube'}

Symbol:


\section*{Category: "Special Controls"}

At runtime, this element indicates automatically that the runtime is busy or waiting for data.

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Example: Data_Transfer \\
Optional \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list.
\end{tabular} \\
\hline "Type of element" & "Busy Symbol, Cube" " \\
\hline
\end{tabular}

Element prop- The position defines the location and size of the element in the visualization window. These erty 'Position' are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X\) " & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (區) to other positions in the editor.

See also
- «y Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center' rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline " \(Y\) " & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus})\) to other positions in the editor.

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|c|c|c|}
\hline \multicolumn{3}{|l|}{"Movement"} \\
\hline "X" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular}} \\
\hline "Y" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the \(Y\) position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular}} \\
\hline "Rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle1. \\
The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol. \\
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
\end{tabular} &  \\
\hline "Interior rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle2. \\
In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. \\
The rotation point is shown as the \(\boldsymbol{\oplus}\) symbol. \\
Note: If a static angle of rotation is specified in the "Position \(\rightarrow\) Angle" property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.
\end{tabular} &  \\
\hline
\end{tabular}

You can link the variables to a unit conversion.


See also

Element prop- The variables control the element behavior dynamically. erty 'State variables'
\begin{tabular}{|l|l}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
\\
TRUE: The element is not visible at runtime.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", "X", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- Requirement: User management is set up for the visualization.
erty 'Access
rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages: \\
\\
\\
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
- "Rights are set: Limited rights": Access is restricted for at least one group. \\
\hline
\end{tabular} \\
\hline
\end{tabular}

See also
- \& Chapter 6.4.5.21.3.1 "Dialog 'Access Rights"' on page 3366

See also
- \({ }^{*} \boldsymbol{y}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Busy Symbol, Flower'}

\section*{Symbol:}

\section*{Category: "Special Controls"}

The element indicates that the system is busy or waiting for data.

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Example: Data_Transfer \\
Optional \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list.
\end{tabular} \\
\hline "Type of element" "Busy Symbol, Flower" \\
\hline
\end{tabular}

Element prop- The position defines the location and size of the element in the visualization window. These erty 'Position' are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X "\) & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (ए-) to other positions in the editor.

\section*{See also}
- \(y^{y}\) Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center'
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline " \(Y\) " & \(Y\)-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols (ఉ) to other positions in the editor.

Element prop- The properties contain fixed values for setting colors. erty 'Colors'
\begin{tabular}{|l|l|}
\hline "Frame color" & \\
\hline "Fill color" & \\
\hline "Transparency" & \begin{tabular}{l} 
Value (0 to 255) for defining the transparency of the selected color. \\
Example 255: The color is opaque. 0: The color is completely transparent.
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\boldsymbol{y y}^{\boldsymbol{y}}\) Chapter 6.4.5.21.3.5 "Dialog 'Gradient Editor"' on page 3369
- \({ }^{*}>\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

Element prop- The properties contain fixed values for setting the look of the element. erty 'Appearance'
\begin{tabular}{|c|c|}
\hline "Line width" & \begin{tabular}{l}
Value in pixels \\
Example: 2 \\
Note: The values 0 and 1 both result in a line weight of 1 pixel. If no line should be displayed, then the "Line style" property must be set to the option "Invisible".
\end{tabular} \\
\hline "Fill attributes" & \begin{tabular}{l}
The way in which the element is filled. \\
- "Filled":The element is filled with the color from property "Colors \(\rightarrow\) Fill color". \\
- "Invisible": The fill color is invisible.
\end{tabular} \\
\hline "Line style" & \begin{tabular}{l}
Type of line representation \\
- "Solid" \\
- "Dashes" \\
- "Dots" \\
- "Dash Dot" \\
- "Dash Dot Dot" \\
- "not visible"
\end{tabular} \\
\hline
\end{tabular}

You can assign variables in the "Appearance variables" property for controlling the appearance dynamically. The fixed values here are overwritten.

See also
- «2" Element property 'Appearance variables'" on page 3292
\begin{tabular}{|l|l|}
\hline "Symbol color" & Selection of a color for the flower symbol. \\
\hline "Line" & Stroke width of the lines (in pixels). \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\boldsymbol{\oplus}\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
\hline "Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol. \\
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

The " \(X\) ", " \(Y\) ", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also

Element prop- The variables control the element behavior dynamically. erty 'State variables'
\begin{tabular}{|l|l}
\hline "Invisible" & Variable (BOOL). Toggles the visibility of the element.
\end{tabular}

TRUE: The element is not visible at runtime.

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", "X", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

\footnotetext{
Element property 'Access
rights'
}
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages: \\
\(-\quad\) "Not set. Full rights.": Access rights for all user groups : "operable" \\
\\
\\
- "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

\section*{See also}
- \({ }^{\#}\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights"' on page 3366

See also
- \(\Leftrightarrow\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Text Editor'}

Symbol:


\section*{Category: "Special Controls"}

The element shows the contents of text files that are saved on the controller. Files can be encoded in ASCII or Unicode formats.
A visualization user can also edit the text.

\section*{Element proper- \\ ties}
\begin{tabular}{|l|l|}
\hline "Element name" & Example: GenElemInst_1 \\
\hline "Type of element" & "Text Editor" \\
\hline
\end{tabular}

Element prop- The position defines the location and size of the element in the visualization window. These erty 'Position' are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline "X" & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (ए) to other positions in the editor.

See also
- « Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element property 'Font'
\begin{tabular}{|l|l|}
\hline "Font name" & \begin{tabular}{l} 
Non-proportional font used by the visualization to display the contents of the file \\
Example: "Courier New"
\end{tabular} \\
\hline "Size" & \begin{tabular}{l} 
Font size \\
Example: 12
\end{tabular} \\
\hline
\end{tabular}

Element prop-
erty 'Control
variables'
Table 631: Element property "Control variables --> File"
\begin{tabular}{|l|l|}
\hline "Variable" & \begin{tabular}{l} 
Variable (STRING). Contains the file names and optionally the location of the file. \\
It is located in the file system of the controller. \\
Example: PLC_PRG.strFile: STRING : = / Documentation/ \\
Info.txt';
\end{tabular} \\
\hline "Open" & \begin{tabular}{l} 
Variable (BOOL). Controls opening the file which is defined in the "Variable" \\
property \\
Example: bopen: BOOL; \\
TRUE: The file is opened.
\end{tabular} \\
\hline "Close" & \begin{tabular}{l} 
Variable (BOOL). Controls closing the file which is defined in the "Variable" prop- \\
erty \\
Example: bClose: BOOL; \\
TRUE: The file is closed.
\end{tabular} \\
\hline "Save" & \begin{tabular}{l} 
Variable (BOOL). Controls saving the file which is defined in the "Variable" prop- \\
erty \\
Example: bStore: BOOL; \\
TRUE: The file is saved.
\end{tabular} \\
\hline "New" & \begin{tabular}{l} 
Variable (BOOL). Controls creating a new file. The name is defined in the \\
"Variable" property. \\
Example: bCreate: BOOL; \\
TRUE: A file is created and opened.
\end{tabular} \\
\hline
\end{tabular}

Table 632: Element property "Control variables --> Edit "
\begin{tabular}{|l|l|}
\hline "Variable" & \begin{tabular}{l} 
Variable (STRING). Contains the string to search for in the file \\
Example: strFind: STRING : \(=~ ' a b c ' ; ~\)
\end{tabular} \\
\hline "Find" & \begin{tabular}{l} 
Variable (BOOL). Controls executing the search for the string in the "Variable" \\
property \\
Example: bFind: BOOL; \\
TRUE: The search is performed. The variable is automatically reset to FALSE.
\end{tabular} \\
\hline "Find next" & \begin{tabular}{l} 
Variable (BOOL). Controls the location to begin the search in the file \\
Example: bFindNext: BOOL; \\
TRUE: The search begins at the last search result location.
\end{tabular} \\
& FALSE: The search begins at the beginning of the file.
\end{tabular}

Table 633: Element property "Control variables --> Cursor position"
\begin{tabular}{|l|l|}
\hline "Line" & \begin{tabular}{l} 
Variable (integer data type). Contains the line of the cursor \\
Example: iRowCursor: INT;
\end{tabular} \\
\hline "Column" & \begin{tabular}{l} 
Variable (integer data type). Contains the column of the cursor \\
Example: iColumnCursor: INT;
\end{tabular} \\
\hline "Position" & \begin{tabular}{l} 
Output variable (integer data type). Shows the absolute cursor position in the \\
text. \\
Example: iPosCursor: INT;
\end{tabular} \\
\hline "Set cursor" & \begin{tabular}{l} 
Variable (BOOL). Controls the setting of the cursor at a specific location \\
Example: iSetCursor: INT; \\
TRUE: The cursor is moved. The new position is defined in the "Line" and \\
"Column" properties. \\
FALSE: The "Line", "Column", and "Position" properties contain the actual \\
values. \\
Note: The variable is used as the control variable for an input event triggered \\
by a visualization user.
\end{tabular} \\
\hline
\end{tabular}

Table 634: Element property "Control variables --> Selection"
\begin{tabular}{|l|l|}
\hline "Start position" & \begin{tabular}{l} 
Output variable (integer data type). Shows the absolute position for starting the \\
text selection \\
Example: iPosSelection: INT;
\end{tabular} \\
\hline "End position" & \begin{tabular}{l} 
Output variable (integer data type). Shows the absolute position for ending the \\
text selection. \\
Example: iPosEndSelection: INT;
\end{tabular} \\
\hline "Start line number" & \begin{tabular}{l} 
Output variable (integer data type). Shows the line where the text selection \\
begins \\
Example: iRowSelection: INT;
\end{tabular} \\
\hline "Start column index" & \begin{tabular}{l} 
Output variable (integer data type). Shows the column where the text selection \\
begins \\
Example: iColumnSelection: INT;
\end{tabular} \\
\hline "End line number" & \begin{tabular}{l} 
Output variable (integer data type). Shows the line where the text selection ends \\
Example: iRowEndSelection: INT;
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "End column index" & \begin{tabular}{l} 
Output variable (integer data type). Shows the column where the text selection \\
ends \\
Example: iCol umnEndSelection: INT;
\end{tabular} \\
\hline "Line to select" & \begin{tabular}{l} 
Variable (integer data type). Contains the line number that is selected \\
Note: The selection is controlled by the variables in the "Trigger selection" prop- \\
erty.
\end{tabular} \\
\hline "Set selection" & \begin{tabular}{l} 
Variable (BOOL). Controls the selection of a line. \\
Example: bSetSelection: BooL;
\end{tabular} \\
\begin{tabular}{l} 
TRUE: The line from the "Line to select" property is selected and highlighted in \\
the Text Editor. \\
if the line is not in the current text segment of the Text Editor, then the text \\
segment is moved to this line. \\
Note: The variable is used as the control variable for an input event triggered \\
by a visualization user. The control variable is not reset automatically. You are \\
responsible for this to occur in the visualization.
\end{tabular} \\
\hline
\end{tabular}

Table 635: Element property "Control variables --> Error handling"
\begin{tabular}{|l|l|}
\hline "Variable for error code" & \begin{tabular}{l} 
Variable (integer data type). Contains the error code when an error occurs \\
\\
\\
\\
\\
\\
\\
\\
\\
\\
\end{tabular} \begin{tabular}{l} 
The error codes are declared in GVL Errorcodes in the \\
VisuElemTextEditor library. To display the error text, the \\
VisuFctTextEditorGetErrorText () function of the library must be called. \\
\hline
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Variable for content changed" & \begin{tabular}{l} 
Variable (BOOL). Shows whether the contents have changed \\
Example: bIsContentEdited: BOOL; \\
TRUE: The contents of the Text Editor have changed.
\end{tabular} \\
\hline "Variable for access mode" & \begin{tabular}{l} 
Variable (BOOL). Controls the access privileges to the file \\
Example: bIsReadOnly: BOOL;
\end{tabular} \\
& \begin{tabular}{l} 
TRUE: A visualization user has read-only permission. At runtime, the file contents \\
are highlighted in gray in the Text Editor. \\
FALSE: A visualization user has read/write permission. \\
\\
Note: The variable overwrites the setting in the "Editor mode" property.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Maximum line length" & Maximum number of characters per line \\
\hline "Editor mode" & \begin{tabular}{l} 
- "Read-only": A visualization user has read-only permissions to the file. At \\
- runtime, the file contents are highlighted in gray in the text editor.
\end{tabular} \\
\hline
\end{tabular}

\section*{Element prop- \\ erty 'New files'}
\begin{tabular}{|c|c|}
\hline "Encoding" & \begin{tabular}{l}
Character encoding of the new file: \\
- "ASCII" \\
- "Unicode (Little endian)" \\
- "Unicode (Big endian)"
\end{tabular} \\
\hline "New line character sequence" & \begin{tabular}{l}
End of line character of the new file: \\
- "CR/LF": Normal for Windows systems \\
- "LF": Normal for UNIX systems \\
Please note: When a visualization user opens an existing file, the end-of-line character of the file is detected and used automatically.
\end{tabular} \\
\hline
\end{tabular}

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- Requirement: User management is set up for the visualization.
erty 'Access
rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
\\
Status messages: \\
\\
\\
\(\bullet\) • "Not set. Full rights.": Access rights for all user groups : "operable"" \\
\(\bullet \quad\) "Rights are set: Limited rights": Access is restricted for at least one group. \\
\hline
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\xi^{\wedge}\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

See also
- \(\Leftrightarrow\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Path3D'}

Symbol:


Path3D

\section*{Category: "Special Controls"}

The "Path3D" visualization element graphically displays the curves of two independent records as a 3D path. It is specially designed for use with Motion Solution CNC in order to display the trajectory of a machine tool or a robot. The programmed path (path) and the path actually traveled (track) is displayed.

Although the visualization element is designed for use with Motion Solution CNC, it can also be used to display any other record. In this case the application has to provide the path data. The sample application 3D Path Generator, which is available in CODESYS Forge, shows how these data can be generated.
If the element is used together with SoftMotion CNC, then function blocks from the library SM3_CNC Visu help to generate the data from the path and track. These function blocks are used by the sample project CNC_File_3DPath, which is stored in the installation directory of CODESYS.
- SMC_PathCopier
- SMC_PathCopierCompleteQueue
- SMC_PathCopierFile
- SMC_PositionTracker

A description of the function blocks can be found in the Library Manager in the library SM3_CNC_Visu.

The element does not work with the CODESYS HMI display variant.

Element proper-
ties
\begin{tabular}{|l|l|}
\hline "Element name" & Example: GenElemInst_1 \\
\hline "Type of element" & "Path3D" \\
\hline
\end{tabular}

\section*{Element property 'Position'}

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline "X" & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}


See also
- Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Path description
\begin{tabular}{|l|l|}
\hline "Path data \\
(VisuStruct3DTrack)" & \begin{tabular}{l} 
Variable of the type VisuSt ruct 3DTrack, which is declared in the IEC code. \\
Example: PLC_PRG.pc. vs 3dt. A description of the structure can be found in \\
the library manager in the library VisuElem3DPath. Iibrary. \\
The data structure describes a path or track through a certain number of points. \\
The points are determined and buffered by the application. The track typically \\
displays the last n positions, so that only a certain part of them is ever displayed \\
at any one time. VisuStruct 3DTrack. pProjection is a variable that is \\
set by the visualization element and contains information about the path/track \\
projection. It can be read (only) by the application. In addition, the methods \\
Projection. Apply or .Appl yV can be used in order to see whether the \\
transformed position lies inside or outside the visualization display area, which is \\
defined by Project ion. ElementRect.
\end{tabular} \\
\hline Color of the path drawn \\
\hline "Path color" & Path line width in pixels, e.g.: "2" \\
\hline "Style of boundary points" & \begin{tabular}{l} 
Display of the points between two successive objects in the path \\
- End points are not displayed
\end{tabular} \\
\hline
\end{tabular}

Track descrip- The track data are structured in exactly the same way as the path data: VisuStruct3DTrack tion
\begin{tabular}{|l|l|}
\hline "Track data \\
(VisuStruct3DTrack)" & \begin{tabular}{l} 
Variable of the type VisuStruct 3DTrack, which is declared in the IEC code. \\
Example: PLC_PRG.pc.vs3dt. A description of the structure can be found in \\
the library manager in the library VisuElem3DPath.library.
\end{tabular} \\
\hline "Track color" & Color of the track drawn \\
\hline "Track line width" & Track line width in pixels, e.g.: "2" \\
\hline
\end{tabular}

Camera control The camera position for the 3D mode is controlled with a reference to the external data structure. This structure allows the following operations:
- Shifting to the left/to the right/upwards/downwards
- Rotation around the \(X / Y / Z\) axis
- Resetting of the view at \(X / Y, Y / Z\) or \(Z / X\) level, so that the path and the track are completely visible.
\begin{tabular}{|l|l|}
\hline "Control data \\
(VisuStruct3DControl)" & \begin{tabular}{l} 
Variable of the type VisuStruct3DControl, which is declared in the IEC \\
code. Example: PLC_PRG.pc.vs3dc. \\
A description of the structure can be found in the library manager in the library \\
VisuElem3DPath.
\end{tabular} \\
& \begin{tabular}{l} 
The values can be set via the application itself or via the visualization element \\
"ControlPanel". The library VisuElem3DPath contains ready-to-use visualiza- \\
tion frames that provide a possible user interface for these data.
\end{tabular} \\
\hline
\end{tabular}

\section*{Additional}
aspects
\begin{tabular}{|l|l|}
\hline "Coordinate system" & \(\boxed{ }:\) : The coordinate system is displayed \\
\hline "Grid" & \(\boxed{\square}:\) Grid lines are displayed \\
\hline "Grid color" & Color of the grid lines \\
\hline
\end{tabular}

Highlighting Individual parts of the path can be visually highlighted. This is typically used to mark the already processed part of a track with a different color. Each point in the path is given a unique ID, which in the case of a CNC editor is linked with the object ID on which the point lies. This ID ("highlight ID") can be specified via the application so that dynamic elements/parts of the track can be highlighted.
\begin{tabular}{|l|l|}
\hline Highlight mode & \begin{tabular}{l} 
Select one of the following highlight modes: \\
\(\bullet \quad\) Only the element whose ID corresponds to the value of the variable is high- \\
lighted. \\
All elements whose ID (linked with the object ID in the case of a CNC editor) \\
is smaller than or equal to the value in Variable are highlighted.
\end{tabular} \\
\hline Variable & \begin{tabular}{l} 
Project variable that specifies the ID of an element. Example: \\
PLC_PRG. iVarElement ID. This "highlight ID" is taken into account for the \\
setting of the highlight mode. The variable must be set in the IEC application.
\end{tabular} \\
\hline Highlight color & \\
\hline
\end{tabular}

\section*{Element look}
\begin{tabular}{|c|c|}
\hline "Frame line width" & Width of the frame around the element, in pixels, for example: "1" \\
\hline "Frame line style" & \begin{tabular}{l}
Select one of these style types for the frame line: \\
- Solid \\
- Dashes \\
- Dots \\
- Dash Dot \\
- Dash Dot Dot \\
- Hollow
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Transparent background" & \begin{tabular}{l}
\(\square:\) The background of the element is displayed transparently. \\
\(\square\)
\end{tabular} \\
\hline : The background of the element is displayed in the defined background color.
\end{tabular} \begin{tabular}{l} 
"Background color" \\
\hline
\end{tabular}

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", "X", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example:bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- Requirement: User management is set up for the visualization.
erty 'Access
rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages:
\end{tabular} \\
& - "Not set. Full rights." Access rights for all user groups : "operable"" \\
- "Rights are set: Limited rights": Access is restricted for at least one group. \\
\hline
\end{tabular}

See also
- \& Chapter 6.4.5.21.3.1 "Dialog 'Access Rights"' on page 3366

\section*{Visualization Element 'Control Panel'}

Symbol:


\section*{Category: "Special Controls"}

This visualization element is used in connection with the "Path3D" visualization element. It is used for changing the position and orientation to the CNC path shown with "Path3D".
See also
- * \({ }^{2}\) Chapter 6.4.5.20.1.42 "Visualization Element 'Path3D'" on page 3279

\section*{Element proper- \\ ties}
\begin{tabular}{|c|c|}
\hline "Element name" & \begin{tabular}{l}
Optional. \\
Hint: Assign individual names for elements so that they are found faster in the element list. \\
Example: Camera_Path_1
\end{tabular} \\
\hline "Type of element" & "Frame" \\
\hline "Clipping" & If you have set the "Scaling type" to "Fixed", then only that part of the visualization is displayed that fits in the frame. \\
\hline "Show frame" & \begin{tabular}{l}
Displays the frame \\
- "No frame": The displayed area of the frame does not have borders. \\
- "Frame": The displayed area of the frame has borders. \\
- "No frame with offset": The displayed area of the frame does not have a border and the displayed area of the referenced visualization is reduced inwards by one pixel as compared to the frame area. \\
The resulting gap prevents the referenced visualization from touching any adjacent elements.
\end{tabular} \\
\hline "Scaling type" & \begin{tabular}{l}
Describes how the frame reacts when the visualization is resized: \\
- "/sotropic": The frame retains its proportions. This allows the ratio of height to width to be preserved, even if the height and width of the visualization have been changed separately. \\
- "Anisotropic": The frame depends on the size of the visualization, so that height and width of the referenced visualization can be changed separately. \\
- "Fixed": The original size of the frame is retained, regardless of the visualization size. If you have also selected the "Clipping" option, then only the fitting part is displayed. \\
- "Fixed and scrollable": The referenced visualization is displayed without scaling. If the value is greater than the window area of the frame, then scrollbars are added to the frame. To set the position of the scroll bar with a variable, use the "Scroll position variable horizontal" or "Scroll position variable vertical" property.
\end{tabular} \\
\hline "Deactivation of the background drawing" & \begin{tabular}{l}
To optimize the performance of the visualization, the non-animated elements of the frame element are drawn as a background bitmap. This could result in the elements not being displayed in the expected order. \\
Deactivation of the background drawing. This can prevent the behavior described above.
\end{tabular} \\
\hline
\end{tabular}

\footnotetext{
Element prop- Contains the currently configured visualization references as a subnode erty 'References'
}
\begin{tabular}{|l|l|}
\hline "References" & \begin{tabular}{l} 
Clicking "Configure" opens the "Frame Configuration" dialog. This is used to \\
manage the referenced visualizations. \\
Caution: Visualizations can be nested at any depth by means of Frame ele- \\
ments. In order to use the "Switch to any visualization" Frame selection type \\
without any problems, a Frame must not contain more than 21 referenced \\
visualizations. For more information, see also the description for the "Input \\
configuration" of an element: Action "Switch Frame visualization".
\end{tabular} \\
\hline \begin{tabular}{ll} 
List of the currently referenced \\
visualizations
\end{tabular} & \begin{tabular}{l} 
Visualizations that have a button also have this displayed as a subnode. Each \\
interface variable is listed with the currently assigned transfer parameters. \\
Example: \\
vis_FormA
\end{tabular} \\
& \begin{tabular}{l} 
- iDataToDisplay_1 : PLC_PRG. iVar1 \\
iDataToDisplay_2 : PLC_PRG. iVar2 \\
Hint: You can change the assignment of the variables to an interface variable \\
here and edit the value field. Or click the "Configure" button instead.
\end{tabular} \\
\hline
\end{tabular}

See also
- « Chapter 6.4.5.21.2.1 "Command 'Interface Editor"" on page 3340
- \(\Leftrightarrow\) Chapter 6.4.5.17 "Creating a structured user interface" on page 2940
- 乡 "Input action 'Switch Frame Visualization"' on page 3377

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X\) " & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (ए़-) to other positions in the editor.

\section*{See also}
- \({ }^{\text {• }}\), Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline "Y" & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus}\) ) to other positions in the editor.

Element prop- The properties contain fixed values for setting colors. erty 'Colors'
\begin{tabular}{|l|l|}
\hline "Color" & \begin{tabular}{l} 
Color for the element in its normal state. \\
Please note that the normal state is in effect if the expression in the \\
"Color variables \(\rightarrow\) Toggle color" property is not defined or it has the value \\
FALSE.
\end{tabular} \\
\hline "Alarm color" & \begin{tabular}{l} 
Color for the element in alarm state. \\
Please note that the alarm state is in effect if the expression in the \\
"Color variables \(\rightarrow\) Toggle color" property has the value TRUE.
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l} 
Value (0 to 255) for defining the transparency of the selected color. \\
Example 255: The color is opaque. 0: The color is completely transparent.
\end{tabular} \\
\hline "Use gradient color" & V: The element is displayed with a color gradient. \\
\hline "Gradient setting" & The "Color gradient editor" dialog box opens. \\
\hline "Frame color" & Example: "Black" \\
\hline "Fill color" & Example: "Light gray" \\
\hline
\end{tabular}

See also
- \({ }^{\text {® }}\) Chapter 6.4.5.21.3.5 "Dialog 'Gradient Editor'" on page 3369

Element prop- The properties contain fixed values for setting the look of the element.
erty 'Appear-
ance'

Value in pixels
Example: 2
Note: The values 0 and 1 both result in a line weight of one pixel. If no line should be displayed, then the "Line style" property must be set to the option "Invisible".

Type of line representation
- "Solid"
- "Dashes"
- "Dots"
- "Dash Dot"
- "Dash Dot Dot"
- "not visible"

You can assign variables in the "Appearance variables" property for controlling the appearance dynamically. The fixed values are defined here.

See also
- « "Element property 'Appearance variables'" on page 3292

Element prop- The properties contains character strings for labeling the element. The character string can also erty 'Texts' contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.
CODESYS accepts the specified texts automatically into the "GlobalTextList" text list. Therefore, these texts can be localized.
\begin{tabular}{|l|l|}
\hline "Text" & \begin{tabular}{l} 
Character string (without single straight quotation marks) for the labeling the \\
element. Add a line break by pressing the keyboard shortcut [CtrI] + [Enter]. \\
Example: Accesses: \%i \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Text".
\end{tabular} \\
\hline "Tooltip" & \begin{tabular}{l} 
Character string (without single straight quotation marks) that is displayed as the \\
tooltip of an element. \\
Example: Number of valid accesses. \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Tooltip".
\end{tabular} \\
\hline
\end{tabular}

See also
- \#2 "Element property 'Text variables"' on page 3289
- © Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872
- Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329

Element prop- The properties contain fixed values for the text properties. erty 'Text properties'
\begin{tabular}{|c|c|}
\hline "Horizontal alignment" & Horizontal alignment of the text within the element. \\
\hline "Vertical alignment" & Vertical alignment of the text within the element. \\
\hline "Text format" & \begin{tabular}{l}
Definition for displaying texts that are too long \\
- "Default": The long text is truncated. \\
- "Line break": The text is split into parts. \\
- "Ellipsis": The visible text ends with "..." indicating that it is not complete.
\end{tabular} \\
\hline "Font" & \begin{tabular}{l}
Example: "Default"
: The "Font" dialog box opens. \\
- : Drop-down list with style fonts.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Font color" & \begin{tabular}{l} 
Example: "Black" \\
\\
\hline... : The "Color" dialog box opens. \\
-: Drop-down list with style colors.
\end{tabular} \\
\hline & \begin{tabular}{l} 
Whole number (value range from 0 to 255 ). This determines the transparency of \\
the respective color. \\
Example: \(255:\) The color is opaque.
\end{tabular} \\
& \begin{tabular}{l} 
0: The color is completely transparent. \\
Please note: If the color is a style color and already has a transparency value, \\
then this property is write-protected.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|l|}{"Movement"} \\
\hline "X" & \begin{tabular}{l}
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline " \(Y\) " & \begin{tabular}{l}
Variable (numeric data type). Defines the \(Y\) position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline "Rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle1. \\
The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol. \\
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
\end{tabular} \\
\hline "Scaling" & \begin{tabular}{l}
Variable (integer data type). Causes centric stretching. \\
Example: PLC_PRG.iScaling. \\
The reference point is the "Center" property. \\
The value 1 shrinks the element by a factor of 0.001 . The value 1000 returns the element to its original size.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol. \\
Note: If a static angle of rotation is specified in the \\
property "Position \(\rightarrow\) Angle", then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
\hline
\end{tabular}


See also

Element prop- The properties contains variables for moving the element. The reference point is the position of erty 'Relative the element ("Position" property). The shape of the element can change. movement'
\begin{tabular}{|c|c|}
\hline "Movement top-left" & \\
\hline " \(X\) " & \begin{tabular}{l}
Variable (integer data type). It contains the number (in pixels) that the left edge is moved horizontally. Incrementing the value moves the element to the right. \\
Example: PLC PRG.iDeltaX
\end{tabular} \\
\hline "Y" & \begin{tabular}{l}
Variable (integer data type). It contains the number (in pixels) that the top edge is moved vertically. Incrementing the value moves the element to the down. \\
Example: \(\qquad\) PRG.iDeltaY
\end{tabular} \\
\hline "Movement bottom-right" & \\
\hline " \(X\) " & \begin{tabular}{l}
Variable (integer data type). It contains the number (in pixels) that the right edge is moved horizontally. Incrementing the value moves the element to the right. \\
Example: PLC_PRG.iDeltaWidth
\end{tabular} \\
\hline "Y" & \begin{tabular}{l}
Variable (integer data type). It contains the number (in pixels) that the bottom edge is moved vertically. Incrementing the value moves the element to the down. \\
Example: PLC_PRG.iDeltaHeight
\end{tabular} \\
\hline
\end{tabular}

See also
- \& "Element property 'Absolute movement"' on page 3319

\section*{Element prop- These properties are variables with contents that replace a format definition. erty 'Text variables'}
\begin{tabular}{|l|l|}
\hline "Text variable" & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses
\end{tabular} \\
& \begin{tabular}{l} 
Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Text". \\
Note: If you specify a variable of type enumeration with text list support, then \\
the name of the enumeration data type is added automatically in angle brackets \\
after the variable name. Example: PLC_PRG.enVar <enumeration name>. \\
Then the symbolic value of the enumeration component is printed instead of the \\
numeric value when text is printed. Refer to the help page for the enumerations.
\end{tabular} \\
\hline "Tooltip variable" & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses InTooltip \\
Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Tooltip".
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{4}\) Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329
- そy "Element property 'Texts'" on page 3286
- \(\xi^{4}\) Chapter 6.4.1.20.5.18 "Enumerations" on page 2263

Element prop- Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is erty 'Dynamic texts' currently indexed in the variable.
\begin{tabular}{|c|c|}
\hline "Text list" & \begin{tabular}{l}
Variable (string) or name of the text list as a fixed string in single straight quotation marks. \\
Example: 'Errorlist' \\
: Drop-down list with the dialogs available in the text lists.
\end{tabular} \\
\hline "Text index" & \begin{tabular}{l}
Text list ID. This refers to the desired output text. \\
- As fixed string with the ID in single straight quotation marks. Example: '1' \\
- As a variable (STRING) for dynamically controlling the text output. \\
Example: strTextID \\
Sample assignment: PLC_PRG.strTextID := '1';
\end{tabular} \\
\hline "Tooltip index" & \begin{tabular}{l}
Text list ID. This refers to the desired output text. \\
- As fixed string with the ID in single straight quotation marks. \\
Example: '2' \\
- As a variable (STRING) for dynamically controlling the text output. \\
Example: strToolTipID \\
Sample assignment: PLC_PRG.strToolTipID := '2';
\end{tabular} \\
\hline \multicolumn{2}{|r|}{\begin{tabular}{l}
See also \\
- \(\boldsymbol{y y}^{4}\) Chapter 6.4.1.21.2.28 "Object 'Text List"" on page 2532
\end{tabular}} \\
\hline Element property 'Font variables' & allow for dynamic control of the text display. \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Font name" & \begin{tabular}{l}
Variable (STRING). Includes the font of the text. \\
Example: PLC_PRG.stFontVar := 'Arial'; \\
The selection of fonts corresponds to the default "Font" dialog.
\end{tabular} \\
\hline "Size" & \begin{tabular}{l}
Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name. \\
- <pt>: Points (default) \\
Example: PLC_PRG.iFontHeight <pt> \\
Code: iFontHeight : INT := 12; \\
- <px> : Pixels \\
Example: PLC_PRG.iFontHeight <px> \\
Code: iFontHeight : INT := 19; \\
If you click in the value field, a drop-down list opens on the right for setting the unit. \\
Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property "Text property \(\rightarrow\) Font".
\end{tabular} \\
\hline "Flags" & \begin{tabular}{l}
Variable (DWORD). Contains the flags for displaying fonts. \\
Flags: \\
- 1: Italics \\
- 2: Bold \\
- 4: Underline \\
- 8: Strikethrough \\
Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: PLC_PRG.dwFontType :=6;
\end{tabular} \\
\hline "Character set" & \begin{tabular}{l}
Variable (DWORD). Contains a character set number for the font. \\
The selection of character set numbers corresponds to the "Script" setting of the standard "Font" dialog.
\end{tabular} \\
\hline "Color" & \begin{tabular}{l}
Variable (DWORD). Includes the color of the text. \\
Example: PLC_PRG.dwColorFont:= 16\#FF000000;
\end{tabular} \\
\hline "Flags for text alignment" & \begin{tabular}{l}
Variable (integer data type). Contains the coding for text alignment. \\
Example: PLC_PRG.dwTextAlignment. \\
Coding: \\
- 0 : Top left \\
- 1: Horizontal center \\
- 2: Right \\
- 4: Vertical center \\
- 8: Bottom \\
Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: PLC_PRG.dwFontType := 5;
\end{tabular} \\
\hline
\end{tabular}

Fixed values for displaying texts are set in "Text properties".

\section*{See also}
- » "Element property 'Text properties'" on page 3286

Element prop- The Element property is used as an interface for project variables to dynamically control colors erty 'Color varia- at runtime.

\section*{bles'}
\begin{tabular}{|c|c|}
\hline "Toggle color" & \begin{tabular}{l}
The property controls the toggled color at runtime. \\
Value assignment: \\
- FALSE: The element is displayed with the color specified in the "Color" property. \\
- TRUE: The element is displayed with the color specified in the "Alarm color" property. \\
Assignment options: \\
- Placeholder for the user input variable \\
- "<toggle/tap variable>" \\
- "<NOT toggle/tap variable>" \\
The color change is not controlled by its own variable, but by a user input variable. \\
Note: Specify a variable for the mouse events "Tap" or "Toggle" in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both "Toggle" and "Tap", then the variable specified in "Tap" is used. \\
Hint: Click the symbol to insert the placeholder "<toggle/tap variable>". When you activate the "Inputconfiguration", "Tap FALSE" property, then the "<NOT toggle/tap variable>" placeholder is displayed. \\
- Instance path of a project variable (BOOL) \\
Example: PLC_PRG.xColorIsToggeled \\
Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.
\end{tabular} \\
\hline \begin{tabular}{l}
"Normal state" \\
"Alarm state"
\end{tabular} & The properties listed below control the color depending on the state. The normal state is in effect if the variable in "Color variables", "Toggle colo"" is not defined or it has the value FALSE. The alarm state is in effect if the variable in "Colorvariables", "Toggle color" has the value TRUE. \\
\hline "Frame color" & \begin{tabular}{l}
Assignment options: \\
- Variable (DWORD) for the frame color Example: PLC_PRG.dwBorderColor \\
- Color literal Example of green and opaque: 16\#FF00FF00
\end{tabular} \\
\hline "Filling color" & \begin{tabular}{l}
Assignment options: \\
- Variable (DWORD) for the fill color Example: PLC_PRG.dwFillColor \\
- Color literal Example of gray and opaque: 16\#FF888888
\end{tabular} \\
\hline
\end{tabular}


See also
- 乡 Chapter 6.4.5.10.4 "Animating a color display" on page 2914

Element prop- The properties contain variables for controlling the appearance of the element dynamically. erty 'Appearance variables'
\begin{tabular}{|c|c|}
\hline \multirow[t]{2}{*}{"Line width"} & Variable (integer data type). Contains the line weight (in pixels). \\
\hline & Note: The values 0 and 1 both result in a line weight of one pixel. If no line should be displayed, then the "Line style" property must be set to the option "Invisible". \\
\hline "Line style" & \begin{tabular}{l}
Variable (DWORD). Controls the line style. \\
Coding: \\
- 0 : Solid line \\
- 1: Dashed line \\
- 2: Dotted line \\
- 3: Line type "Dash Dot" \\
- 3: Line type "Dash Dot Dot" \\
- 8: Invisible: The line is not drawn.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Variable" & \begin{tabular}{l}
- Variable (integer data type) that contains the index of the active visualization Example: PLC_PRG.uiIndexVisu \\
Hint: The "Frame Configuration" dialog includes a list of referenced visualizations. The visualizations are automatically numerically indexed via the order in the list. \\
Note: This variant of switching usually affects all connected display variants. \\
- Array element (integer data type) for index access via CURRENTCLIENTID Example: PLC_PRG.aIndexVisu[CURRENTCLIENTID] \\
Note: This variant of switching applies to the current client only, and therefore only on one display variant. That is the display variant where the value change was triggered (for example, by means of user input).
\end{tabular} \\
\hline
\end{tabular}

\section*{See also}
- Chapter 6.4.5.21.2.9 "Command 'Frame Selection"' on page 3348

Element prop- The variables control the element behavior dynamically. erty 'State varia-
bles'
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
\\
\\
\\
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \(=\) FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain the configurations for the user input when using the mouse or keyboard. erty 'Input con- A user input defines an event and one or more actions that are executed when an event occurs. figuration'
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
The "Configure" button opens the "Input Configuration" dialog. There you can create or edit user inputs. \\
Configured user inputs are listed below the events. They each include the action that is triggered and the setting \\
in short form. \\
Example: "Execute ST Code": उPLC_PRG.i_x \(:=0 ;\)
\end{tabular} \\
\hline "OnDialogClosed" & Input event: The user closes the dialog. \\
\hline "OnMouseClick" & \begin{tabular}{l} 
Input event: The user clicks the mouse button completely in the element area. \\
The mouse button is clicked and released.
\end{tabular} \\
\hline "OnMouseDown" & Input event: The user clicks down on the mouse button. \\
\hline "OnMouseEnter" & Input event: The user drags the mouse pointer to the element. \\
\hline "OnMouseLeave" & Input event: The user drags the mouse pointer away from the element. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "OnMouseMove" & Input event: The user moves the mouse pointer over the element area. \\
\hline "OnMouseUp" & \begin{tabular}{l} 
Input events: \\
- \begin{tabular}{l} 
The user releases the mouse button within the element area. It is irrelevant \\
whether the user has previously pressed the mouse button inside or outside \\
the element area.
\end{tabular} \\
- The user presses the mouse button within the element area, leaves the \\
element area, and then releases the mouse button. \\
Note: This CODESYS-specific triggering behavior guarantees that actions for \\
key elements are completed. A key element starts an action for "OnMouseDown"" \\
and ends the action for "OnMouseUp". \\
Example: A visualization user presses the mouse button within the element area \\
of the key element and then moves the cursor position so that it lies outside the \\
element area. The action is ended anyway because "OnMouseUp" is triggered.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Tap" & \begin{tabular}{l} 
When a mouse click event occurs, the variable defined in "Variable" is described \\
in the application. The coding depends on the "Tap FALSE" and "Tap on enter if \\
captured" options.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL) that is set on mouse click event. \\
Example: PLC_PRG. bIsTapped \\
TRUE: A mouse click event exists. It lasts as long as the user presses the mouse \\
button over the element. It ends when the button is released. \\
FALSE: A mouse click event does not exist. \\
Requirement: The "Tap FALSE" option is not activated.
\end{tabular} \\
\hline "Tap FALSE" & \begin{tabular}{l} 
V: The mouse click event leads to a complementary value in "Variable". \\
TRUE: A mouse click event does not exist.
\end{tabular} \\
\hline "Tap on enter if captured" & \begin{tabular}{l} 
FALSE: While the mouse click event exists.
\end{tabular} \\
\hline \begin{tabular}{l} 
pointer is dragged within the element area or not while the mouse button is \\
pressed. \\
TRUE: While the mouse click event exists and the mouse pointer is moved over \\
the element area. \\
FALSE: A mouse click event does not exist. Or the user moves the mouse \\
pointer outside of the element area while the mouse button is pressed. \\
The value is TRUE again as soon as the user moves the pointer back to the \\
element area. The mouse is then captured.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Toggle" & \begin{tabular}{l} 
With the onset of a mouse click event, the variable is set; when the mouse click \\
event is completed, the variable is reset.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL). Its value toggled when the mouse click event is ended. This is \\
when the user releases the mouse button while the mouse pointer is over the \\
element area. \\
If the user releases the mouse button while the mouse pointer is outside of the \\
element area, then the mouse click event is not ended and the value is not \\
toggled. \\
Hint: The user can cancel a started toggle input by dragging the mouse pointer \\
out of the element area.
\end{tabular} \\
\hline "Toggle on up if captured" & \begin{tabular}{l} 
V: The value toggles regardless of where the mouse pointer is when the mouse \\
button is released. The mouse is then captured.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Hotkey" & \begin{tabular}{l}
Keyboard shortcut on the element for triggering specific input actions. \\
When the keyboard shortcut event occurs, the input actions in the "Events" property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.
\end{tabular} \\
\hline "Key" & \begin{tabular}{l}
Key pressed for input action. \\
Example: [T] \\
Note: The following properties appear when a key is selected.
\end{tabular} \\
\hline "Events" & \begin{tabular}{l}
- "None" \\
- "Mouse down": Pressing the key triggers the input actions that are configured in the "OnMouseDown" property. \\
- "Mouse up": Releasing the key triggers the input actions that are configured in the "OnMouseUp" property. \\
- "Mouse down/up": Pressing and releasing the key triggers the input actions that are configured in the "OnMouseDown" property and the "OnMouseUp" property.
\end{tabular} \\
\hline "Shift" & Combination with the Shift key Example: [Shift]+[T]. \\
\hline "Control" & Combination with the Ctrl key Example: [CtrI]+[T]. \\
\hline "Alt" & Combination with the Alt key Example: [AIt]+[T]. \\
\hline
\end{tabular}

All keyboard shortcuts and their actions that are configured in the visualization are listed on the "Keyboard Configuration" tab.

\section*{See also}
- \({ }^{4}\) Chapter 6.4.5.21.2.2 "Command 'Keyboard Configuration'" on page 3341
- \(\quad{ }^{*}\) Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration'" on page 3370

See also
- ② Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Cartesian XY Chart'}

Symbol:


\section*{Category: "Special Controls"}

The element displays the curve of array values graphically as a line or bar chart in the Cartesian coordinate system. The chart can display multiple curves at one time.

\section*{NOTICE!}

\section*{Constraint}

The element can be used with controller with V3.5 SP11 and higher.

\section*{in CODESYS Forge, you will find a sample project for using "Cartesian XY} Chart" elements in visualizations.

See also
- Sample project in CODESYS Forge

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & Example: velocity chart \\
\hline "Type of element" & "Cartesian XY Chart" \\
\hline "Cartesian XY Chart" & \begin{tabular}{l} 
@ XYChart: Opens the "XY Chart Configuration" dialog. This is where the chart \\
is configured.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X "\) & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline " \(Y\) " & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols ( ( - ) to other positions in the editor.

See also
- Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The properties contain fixed values for defining the look of the element.
erty 'Element
look'
\begin{tabular}{|l|l|}
\hline "Border line width" & \begin{tabular}{l} 
Value (in pixels) \\
Example: 2 \\
\\
\\
\\
\\
\\
Note: The values 0 and 1 both result in a line weight of one pixel. If no line \\
should be displayed, then the "Border line style" property must be set to the \\
option "Invisible".
\end{tabular} \\
\hline & \begin{tabular}{l} 
- "Solid" \\
- "Dash" \\
- "Dots" \\
- "Dash Dot"
\end{tabular} \\
& \begin{tabular}{l} 
- "Dash Dot Dot" \\
\\
\\
\\
\\
\\
- "Invisible"
\end{tabular} \\
\hline - Style color from the list box. Example: Black \\
& - Fixed value that is selected in the color dialog. Example: \(0 ; 0 ; 0\) \\
\hline
\end{tabular}

Element prop-
erty 'Axis font'
\begin{tabular}{|l|l|}
\hline "Font" & Example: "Default" \\
& \begin{tabular}{l}
...: Opens the "Font" dialog. \\
\(\mathbf{v :}\) : List box with style fonts
\end{tabular} \\
\hline
\end{tabular}

\section*{Element prop- \\ erty 'Control variables'}

Table 636: "Zoom"
\begin{tabular}{|l|l|}
\hline & \begin{tabular}{l} 
Zooming the displayed curve is done by means of the mouse, or the pinch \\
gesture on a multitouch device. It also applies to all axes. \\
At runtime when "Enable" is TRUE, you can draw a box with the mouse by \\
holding down the left mouse button. When you release the mouse button, the \\
display zooms in on the box and the curve is magnified. To zoom in and out \\
on a multitouch device, move two fingers together or away from each other, \\
respectively. \\
Zooming and panning can work together.
\end{tabular} \\
\hline "Enable" & \begin{tabular}{l} 
Variable (BOOL) that enables or disables zooming. \\
TRUE: Enables zooming \\
Example: PLC_PRG. xZoomEnable
\end{tabular} \\
\hline "Home" & \begin{tabular}{l} 
Variable (BOOL) \\
Rising edge: Reset the displayed curve to the initial state after the display has \\
changed due to zooming. \\
Example: PLC_PRG.xZoomHome
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Undo" & \begin{tabular}{l} 
Variable (BOOL) \\
Rising edge: Reset the displayed curve to the previous position after the display \\
has changed due to zooming. \\
Example: PLC_PRG. xZoomUndo
\end{tabular} \\
\hline "Is zoomed" & \begin{tabular}{l} 
Variable (BOOL) that indicates whether or not the displayed curve was modified \\
due to zooming. \\
TRUE: Curve setting was zoomed. \\
Example: PLC_PRG.xIsZoomed
\end{tabular} \\
\hline
\end{tabular}

Table 637: "Pan"
\begin{tabular}{|l|l|}
\hline & \begin{tabular}{l} 
Panning the displayed curve is done by means of the mouse or the pinch ges- \\
ture on a multitouch device. It also applies to all axes. \\
At runtime if "Enable" is TRUE, then you can drag the displayed curve to another \\
position by holding down the left mouse button. To pan the displayed curve on a \\
multitouch device, drag it with one finger to another position.
\end{tabular} \\
\hline "Enable" & \begin{tabular}{l} 
Variable (BOOL) to enable or disable panning. \\
TRUE: Enables panning \\
Example: PLC_PRG. xPanEnable
\end{tabular} \\
\hline "Home" & \begin{tabular}{l} 
Variable (BOOL) \\
Rising edge: Reset the displayed curve to the initial position after the display has \\
changed due to panning. \\
Example: PLC_PRG.xPanHome
\end{tabular} \\
\hline "Is panned" & \begin{tabular}{l} 
Variable (BOOL) whose state indicates whether or not the displayed curve was \\
modified due to zooming. \\
TRUE: Curve setting was panned. \\
Example: PLC_PRG.xIsPanned
\end{tabular} \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" & \multicolumn{1}{|l|}{\begin{tabular}{|l|l|}
\hline Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular}} \\
\hline "Y" & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|}
\hline "Rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle1. \\
The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol. \\
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
\end{tabular} &  \\
\hline "Interior rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle2. \\
In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. \\
The rotation point is shown as the \(\boldsymbol{\oplus}\) symbol. \\
Note: If a static angle of rotation is specified in the "Position \(\rightarrow\) Angle" property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.
\end{tabular} &  \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

The " \(X\) ", " "", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

\section*{Element prop- The variables control the element behavior dynamically. erty 'State variables'}
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \(=\) FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages: \\
\\
\\
\(\bullet\) - "Not set. Full rights.": Access rights for all user groups : "operable"" \\
• "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\quad\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

\section*{Visualization Element 'Date Range Picker'}

Symbol:

\section*{Category: "Date/Time Controls"}

The element provides the capability of selecting the date and time range of a saved data set. The element is used with the "Trend" visualization element.

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & Example: DateTrend1 \\
& \begin{tabular}{l} 
Optional \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list.
\end{tabular} \\
\hline "Type of element" & "Date Range Picker" \\
\hline
\end{tabular}

Element prop- The position defines the location and size of the element in the visualization window. These erty 'Position' are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X "\) & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (무) to other positions in the editor.

See also
- \#y Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline\(" X "\) & X-coordinate of the point of rotation \\
\hline\(" Y "\) & \(Y\)-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus})\) to other positions in the editor.
\begin{tabular}{|l|l|}
\hline "Show frame" & \(\boxed{\square}:\) The visualization element is drawn with a frame. \\
\hline "Resolution" & Resolution saved for the time stamp: "Millisecond" or "Microsecond" \\
\hline "Attached element instance" & \begin{tabular}{l} 
The element can be assigned to a "Trend" visualization element. As a result, the \\
time range of the trend element can be changed. The available visual elements \\
are selected with the help of the Input Assistant ( \(\ldots)\) ).
\end{tabular} \\
\hline
\end{tabular}

\section*{Element prop- \\ erty 'Tick mark \\ labels'}
\begin{tabular}{|l|l|}
\hline "Two-line labelling" & \begin{tabular}{l} 
V: The time stamps are displayed in two lines. The date is displayed in the first \\
line and the time is displayed in the second line. \\
\(\square\) \\
in : Time stamp is displayed in one line. The date and time can also be displayed
\end{tabular} \\
\hline " one lending on the formatting.
\end{tabular}.

Element prop- The properties contain fixed values for the text properties.
erty 'Text properties'
\begin{tabular}{|c|c|}
\hline "Horizontal alignment" & Horizontal alignment of the text within the element. \\
\hline "Vertical alignment" & Vertical alignment of the text within the element. \\
\hline "Font" & Example: "Default"
The "Font" dialog box opens.
Drop-down list with style fonts. \\
\hline "Font color" & \begin{tabular}{l}
Example: "Black"
The "Color" dialog box opens. \\
*: Drop-down list with style colors.
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l}
Whole number (value range from 0 to 255). This determines the transparency of the respective color. \\
Example: 255: The color is opaque. \\
0 : The color is completely transparent. \\
Please note: If the color is a style color and already has a transparency value, then this property is write-protected.
\end{tabular} \\
\hline
\end{tabular}

\section*{Element property 'Additional buttons'}
\begin{tabular}{|c|c|}
\hline "Jump to the largest possible time stamp" & \(\checkmark\) : An additional button ( \(\quad\) ) is displayed for jumping to the last time stamp. \\
\hline "Jump to the smallest possible time stamp" & \(\checkmark\) : An additional button (14) is displayed for jumping to the first time stamp. \\
\hline "Zoom out" & An additional button ( \(\sigma\) ) is displayed for setting the current min./max. range to the maximum range. The selected range is left. \\
\hline
\end{tabular} erty 'Absolute movement'

Element prop- The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|c|c|c|}
\hline \multicolumn{3}{|l|}{"Movement"} \\
\hline "X" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the \(X\) position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular}} \\
\hline "Y" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the \(Y\) position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular}} \\
\hline "Rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle1. \\
The midpoint of the element rotates at the "Center" point. This rotation point is shown as the \(\boldsymbol{\oplus}\) symbol. \\
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
\end{tabular} &  \\
\hline "Interior rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle2. \\
In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. \\
The rotation point is shown as the \(\boldsymbol{\oplus}\) symbol. \\
Note: If a static angle of rotation is specified in the "Position \(\rightarrow\) Angle" property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.
\end{tabular} &  \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

The "X", " \(Y\) ", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also
-

Element prop- The variables control the element behavior dynamically. erty 'State variables'
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- Requirement: User management is set up for the visualization. erty 'Access
rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages:
\end{tabular} \\
& \begin{tabular}{l} 
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
\(\bullet \quad\) "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

See also
- ※y Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

\section*{Visualization Element 'Time Range Picker'}

Symbol:


\section*{Category: "Date/Time Controls"}

The element provides configurable buttons for setting the time range of a trend display to a defined time. In the process the end time of the previous display is left unchanged and the start time is adapted.

\section*{Element proper- \\ ties}
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Optional \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list. \\
Example: TimeRangeTemperature
\end{tabular} \\
\hline "Type of element" & "Time range picker" \\
\hline
\end{tabular}

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X\) " & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (ए़-) to other positions in the editor.

See also
- © Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874
\begin{tabular}{|l|l|}
\hline "Orientation" & \begin{tabular}{l} 
Specifies whether the time picker element is aligned horizontally or vertically in \\
the editor. \\
Hint: Change the width to height ratio of the element in the editor.
\end{tabular} \\
\hline "Show frame" & \(\boxed{\square}\) : The visualization element is drawn with a frame. \\
\hline "Resolution" & Resolution saved for the time stamp: "Millisecond" or "Microsecond" \\
\hline "Attached element instance" & \begin{tabular}{l} 
Assignment to the element that processes the time picker \\
The element can be assigned for example to a "Trend" visualization element. \\
Then the time range of the trend element can be changed. The available visual \\
elements are selected with the help of the input assistance (...). \\
Example: GenElemInst_1
\end{tabular} \\
\hline
\end{tabular}

Element prop-
erty 'Texts'
\begin{tabular}{|l|l|}
\hline "Text" & \begin{tabular}{l} 
String label for the element. \\
Example: Zoom
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain fixed values for the text properties.
erty 'Text prop-
erties'
\begin{tabular}{|l|l|}
\hline "Font" & \begin{tabular}{l} 
Example: "Default" \\
\(\ldots . .:\) The "Font" dialog box opens. \\
v: Drop-down list with style fonts.
\end{tabular} \\
\hline "Font color" & \begin{tabular}{l} 
Example: "Black" \\
\(\ldots . .:\) The "Color" dialog box opens. \\
v: Drop-down list with style colors.
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l} 
Whole number (value range from 0 to 255). This determines the transparency of \\
the respective color. \\
Example: \(255:\) The color is opaque. \\
\(0:\) The color is completely transparent. \\
Please note: If the color is a style color and already has a transparency value, \\
then this property is write-protected.
\end{tabular} \\
\hline
\end{tabular}

Property 'Times' In "Times", the buttons that the element provides at runtime are defined and configured in an array.
\begin{tabular}{|c|c|}
\hline "Provide "All" selection" & :Time Range Picker bar extended by "All" button. The diagram represents a time interval that covers all time stamps. \\
\hline "Times" & Adds another button to the Time Range Picker bar and increases the array by one entry. An additional index is present in the property "Times \(\rightarrow\) Times \(\rightarrow\) Times \(\rightarrow\) [<new>]". "Time" is located under this index. The configuration of the button is to be entered there. \\
\hline \begin{tabular}{l}
"Times" \\
- " [Index]" \\
with index \(\in\{0,1,2, \ldots\}\)
\end{tabular} & \begin{tabular}{l}
Array of all buttons in the time selection bar. Index corresponds to the number of buttons. \\
\(\chi\) : The associated button is removed from the Time Range Picker bar. The configuration entry is deleted from the "Times" property list.
\end{tabular} \\
\hline \begin{tabular}{l}
" [Index]" \\
- "Time"
\end{tabular} & Time interval in standardized notation. Example: 3M for 3 months; 30 m for 30 minutes. If a time interval is indicated in the field, then the button is labelled with it. If a user clicks on the button at runtime, the command is executed to switch the diagram to this time interval. The default is empty. \\
\hline
\end{tabular}

Element prop-
erty 'Control
variables'
\begin{tabular}{|l|l|}
\hline "Time" & \begin{tabular}{l} 
Displays which time is currently selected. \\
Variable (STRING) \\
Example: PLC_PRG.strSelcetedTime
\end{tabular} \\
\hline "All" selected" & \begin{tabular}{l} 
Displays the state of the "All" button \\
Variable (BOOL) \\
Example: PLC_PRG.AllTimesAreSelected
\end{tabular} \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG. iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG. iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\boldsymbol{\oplus}\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
\hline "Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol. \\
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

The " \(X\) ", " \(Y\) ", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

\section*{See also}

\section*{Element prop- The variables control the element behavior dynamically. erty 'State variables'}
\begin{tabular}{|l|l}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- Requirement: User management is set up for the visualization.
erty 'Access
rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages:
\end{tabular} \\
& \begin{tabular}{l} 
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
• "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\xi^{\prime}\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

\section*{Visualization Element 'Date Picker'}

\section*{Symbol:}


\section*{Category: "Date/Time Controls"}

The element is a calendar that displays the current date. A user can click a day to select a date, which is saved to a variable. In addition, it can customize the time interval that the calendar displays. Clicking the calendar header changes the year. Clicking the arrows in the calendar header changes the month.

Languagedependent texts of the element

The element contains language-dependent texts that are managed in the system text list. This deals with the names of the month and the days of the week written out completely or abbreviated. When the date picker is added to a visualization, CODESYS generates the text list automatically below the POU view. The IDs correspond to the standard text and therefore English terms. The text list makes it possible to translate these texts.

Example System text list
\begin{tabular}{|l|l|}
\hline ID & Default \\
\hline Apr & Apr \\
\hline April & April \\
\hline
\end{tabular}

\section*{See also}
- \({ }^{*}\) Chapter 6.4.5.8 "Setting Up Multiple Languages" on page 2906

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & Example: DueDateCalendar \\
\hline "Type of element" & "Date Picker" \\
\hline
\end{tabular}

Element prop- The position defines the location and size of the element in the visualization window. These erty 'Position' are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline "X" & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (ए-) to other positions in the editor.

See also
- « Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center' rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X "\) & X-coordinate of the point of rotation \\
\hline " \(Y\) " & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\mathbf{\oplus})\) to other positions in the editor.
\begin{tabular}{|l|l|}
\hline "Variable" & \begin{tabular}{l} 
Input variable (DATE). Contains the date that a user selects in the calendar. \\
Example: PLC_PRG. dtDueDate
\end{tabular} \\
\hline "Design" & \begin{tabular}{l} 
- "From style": All settings are preconfigured according to the style. \\
- "Explicit": The "Design settings" property is available. You can customize the \\
calendar here.
\end{tabular} \\
\hline
\end{tabular}

Design settings Requirement: This property is visible only if the "Design" property is set to "Explicit". The values of the property can be predefined in the style. Then they are available in the drop-down list.

Table 638: "Header of Date Picker"
\begin{tabular}{|l|l|}
\hline Design of the header & Style font or user-defined font \\
\hline "Font" & Style color or user-defined color \\
\hline "Font color" & \\
\hline "Arrows" & Style color or user-defined color \\
\hline "Arrow color" & \\
\hline "Color of printed arrow" & \\
\hline "Background" & \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Draw background" & "From style": The style defines whether and how a background is drawn. \\
"Yes": The background is filled with the color in the "Background color" property. \\
& "No": The background is not filled with a color.
\end{tabular}

Table 639: Design of the main display area
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
Design of the main display \\
area
\end{tabular} & \\
\hline "Today" & Design of today \\
\hline "Font" & Style font or user-defined font \\
\hline "Font color" & Style color or user-defined color \\
\hline "Draw background" & \begin{tabular}{ll} 
"From style": The style defines whether and which background is drawn. \\
"Yes": The background is filled with the color in the "Background color" property. \\
"No": The background is not filled with a color.
\end{tabular} \\
\hline "Background color" & Style color or user-defined color. Used if "Yes" is selected in "Draw background". \\
\hline "Show frame" & \begin{tabular}{l} 
"From style": The style defines whether and how a frame is drawn. \\
"Yes": The frame is displayed with the following properties. \\
"No": A frame is not displayed.
\end{tabular} \\
\hline "Frame color" & Used if "Yes" is selected in "Show frame". \\
\hline "Rectangle type" & \begin{tabular}{l} 
"Line width"
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Selected day" & Design of the selected day \\
\hline "Font" & Style font or user-defined font \\
\hline "Font color" & Style color or user-defined color \\
\hline "Draw background" & \begin{tabular}{l} 
"From style": The style defines whether and how a background is drawn. \\
"Yes": The background is filled with the color in the "Background color" property. \\
"No": The background is not filled with a color.
\end{tabular} \\
\hline "Background color" & Style color or user-defined color \\
\hline "Show frame" & \begin{tabular}{l} 
"From style": The style defines whether and how a background is drawn. \\
"Yes": The frame is displayed with the following properties. \\
"No": A frame is not displayed.
\end{tabular} \\
\hline "Frame color" & Used if "Yes" is selected in "Show frame". \\
\hline "Rectangle type" & \\
\hline "Line width" & \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Current month" & Design of the current month \\
\hline "Font" & Style font or user-defined font \\
\hline "Font color" & Style color or user-defined color \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Draw background" & \begin{tabular}{l} 
"From style": The style defines whether and how a background is drawn. \\
"Yes": The background is filled with the color in the "Background color" property. \\
"No": The background is not filled with a color.
\end{tabular} \\
\hline "Background color" & \multicolumn{4}{|l|}{\begin{tabular}{ll} 
"Show frame" & \begin{tabular}{l} 
"From style": The style defines whether and how a frame is drawn. \\
"Yes": The frame is displayed with the following properties. \\
"No": A frame is not displayed.
\end{tabular} \\
\hline "Frame color" & Used if "Yes" is selected in "Show frame". \\
\hline "Rectangle type" & \\
\hline
\end{tabular}} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Other months" & Design of the previous and subsequent months \\
\hline "Font" & Style font or user-defined font \\
\hline "Font color" & Style color or user-defined color \\
\hline "Display other month" & Design of the previous and subsequent months \\
\hline "Draw background" & \begin{tabular}{l} 
"From style": The style defines whether and how a background is drawn. \\
"Yes": The background is filled with the color in the "Background color" property. \\
"No": The background is not filled with a color.
\end{tabular} \\
\hline "Background color" & \begin{tabular}{l} 
"From style": The style defines whether and how a frame is drawn. \\
"Yes": The frame is displayed with the following properties. \\
"No": A frame is not displayed.
\end{tabular} \\
\hline "Frame color" & Used if "Yes"is selected in "Show frame". \\
\hline "Rectangle type" & \\
\hline "Line width" &
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Day of week heading" & Design of the heading with the days of the week \\
\hline "Font" & Style font or user-defined font \\
\hline "Font color" & Style color or user-defined color \\
\hline "Draw background" & \begin{tabular}{l} 
"From style": The background is filled with the style color "From style". The style \\
defines whether and how a background is drawn. \\
"Yes": The background is filled with the color in the "Background color" property. \\
"No": The background is not filled with a color.
\end{tabular} \\
\hline "Background color" & \multicolumn{1}{|l|}{\begin{tabular}{l} 
"From style": The style defines whether and how a frame is drawn. \\
"Yes": The frame is displayed with the following properties. \\
"No": A frame is not displayed.
\end{tabular}} \\
\hline "Frame color" & Used if "Yes" is selected in "Show frame". \\
\hline "Rectangle type" & \multicolumn{1}{|l|}{\begin{tabular}{l} 
"Line width"
\end{tabular}} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Display separator line" & \begin{tabular}{l} 
"From style": The style defines whether and how a separator line is drawn. \\
"Yes": Display with the following properties. \\
"No": A separator line is not displayed.
\end{tabular} \\
\hline "Color of the separator line" & Used if "Yes" is selected in "Display separator line". \\
\hline "Width of separator line" & \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Background" & Design of the calendar days \\
\hline "Draw background" & \begin{tabular}{l} 
"From style": The style defines whether and how a background is drawn. \\
"Yes": The background is filled with the color in the "Fill color" property and \\
framed in the "Frame color". \\
"No": The background is not filled with a color.
\end{tabular} \\
\hline "Fill color" & Style color or user-defined color \\
\hline "Frame color" & \\
\hline
\end{tabular}

\section*{Element prop- \\ erty 'Display \\ type'}
\begin{tabular}{|l|l|}
\hline "Rows" & Number of month calendars per row (preset: 1) \\
\hline "Columns" & Number of month calendars per column (preset: 1) \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|c|c|c|}
\hline \multicolumn{3}{|l|}{"Movement"} \\
\hline "X" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the \(X\) position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular}} \\
\hline "Y" & \begin{tabular}{l}
Variable (numeric data type). Defines the \(Y\) position \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the ele
\end{tabular} & \begin{tabular}{l}
(in pixels). \\
ment downwards.
\end{tabular} \\
\hline "Rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle1. \\
The midpoint of the element rotates at the "Center" point. This rotation point is shown as the \(\boldsymbol{\oplus}\) symbol. \\
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
\end{tabular} &  \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\boldsymbol{\oplus}\) symbol.
\end{tabular} \\
\begin{tabular}{l} 
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

The "X", " \(Y\) ", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also
-

Element prop- The variables control the element behavior dynamically.
erty 'State varia-
bles'
\begin{tabular}{|l|l|}
\hline "Invisible" & Variable (BOOL). Toggles the visibility of the element. \\
& \begin{tabular}{l} 
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \(=\) FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|l|l|}
\hline "Animation duration" & \begin{tabular}{l} 
Defines the duration (in milliseconds) in which the element runs an animation \\
- \begin{tabular}{l} 
Variable (integer value) \\
Example: Menu. tContent with VAR tContent : INT \(:=500 ;\) \\
END_VAR \\
- Integer literal \\
Example: 500
\end{tabular} \\
Animatable properties \\
- "Absolute movement", "Movement", "X", " \(Y\) " \\
- "Absolute movement", ""otation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable \\
property has changed. The movement then executed is not jerky, but is smooth \\
within the specified animation duration. The visualization element travels to the \\
specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline \begin{tabular}{l} 
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIs InForeground with VAR bIsInForeground : BooL \(:=\) \\
FALSE; END_VAR
\end{tabular} \\
\hline "Move to foreground" \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it \\
was inserted in the visualization editor.
\end{tabular}

Element prop- Requirement: User management is set up for the visualization. erty 'Access
rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages:
\end{tabular} \\
& \begin{tabular}{l} 
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
\(\bullet\) \\
"Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

See also
- ※y Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

See also
- \(\Leftrightarrow_{2}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Analog Clock'}

Symbol:


\section*{Category: "Date/Time Controls"}

The element is a clock that displays the current time of day. The clock can also display a random time.

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & Example: GenElemInst_1 \\
\hline "Type of element" & "Analog Clock" \\
\hline
\end{tabular}

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X "\) & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (區) to other positions in the editor.

See also
- 出 Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center' rotation is shown as the \(\oplus\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline\(" Y\) " & \(Y\)-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus}\) ) to other positions in the editor.

\section*{Element property 'Time Display'}
\begin{tabular}{|l|l|}
\hline "Use system time" & 目: The system time of the PLC is displayed. \\
\hline "Variable" & \begin{tabular}{l} 
Variable (time data type TOD, TIME_OF_DAY). This receives the time of day that \\
is not the system time. \\
Example: PLC_PRG. todTimeTokio \\
Requirement: The "Use system time" property is not activated.
\end{tabular} \\
\hline
\end{tabular}

See also
- 㘯 Chapter 6.4.1.20.5.6 "Data Type 'TIME"' on page 2237
\begin{tabular}{|l|l|}
\hline "Design" & \begin{tabular}{l}
\(\bullet\) \\
\(\bullet\) "From style": All settings are preconfigured according to the style. \\
"Explicit": The "Settings" property is available. Here you can customize the \\
analog clock.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline " \(Y\) " & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG.iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\boldsymbol{\oplus}\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
\hline "Scaling" & \begin{tabular}{l} 
Variable (integer data type). Causes centric stretching. \\
Example: PLC_PRG. iScaling. \\
The reference point is the "Center" property. \\
The value 1 shrinks the element by a factor of \(0.001 . ~ T h e ~ v a l u e ~\) \\
to its original size.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol. \\
Note: If a static angle of rotation is specified in the \\
property "Position \(\rightarrow\) Angle", then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
\hline "Use REAL values"" & \begin{tabular}{l} 
Note: Only available if the device supports the use \\
of REAL coordinates. \\
V: The properties of the absolute movement are \\
interpreted as REAL values. The values are not \\
rounded. \\
The option allows for the individual fine-tuning of \\
drawing the element, for example for the visualiza- \\
tion of a smoother rotation. \\
Hint: If a horizontal or vertical line is drawn blurry \\
on a specific visualization platform, then this can \\
be corrected by an offset of 0.5px in the direction \\
of the line thickness.
\end{tabular} \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

The properties " \(X\) ", " \(Y\) ", "Rotation", and "Interior rotation" are supported by the "Client Animation" functionality.

Element prop- Requirement: The "Property" is "Explicit". Only then is the "Clock Settings" category visible. erty 'Settings'

Table 640: "Background"
\begin{tabular}{|l|l|}
\hline "Background color"" & Color variants of the default background image \\
& \(\bullet\) "Yellow" \\
& \(\bullet\) "Red" \\
& \(\bullet\) "Blue" \\
& \(\bullet\) "Green" \\
& "Black" \\
\hline "Own background" & \begin{tabular}{l} 
Background display with the specific "Image". Replaces the default background \\
image.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Image" & \begin{tabular}{l} 
Image from an image pool or library \\
Example: my Imagepool .my Image
\end{tabular} \\
\hline "Transparency color" & \begin{tabular}{l} 
The transparent color in the image representation. \\
Example: "White". The white parts of the image are transparent.
\end{tabular} \\
\hline "Use background color" & \begin{tabular}{l} 
V: The image background is displayed using the color defined in the \\
"Background color" property. \\
Requirement: No image reference is given in the "Image" property.
\end{tabular} \\
\hline "Background color" & \begin{tabular}{l} 
Style color or color \\
Requirement: "Use background color" is activated.
\end{tabular} \\
\hline
\end{tabular}

Table 641: "Hands"
\begin{tabular}{|l|l|}
\hline "Hand style" & Example: "Thin arrow" \\
\hline "Color hour hand" & Style color or color for the hands \\
\cline { 1 - 1 } "Color minute hand" & \\
\hline "Color second hand" & \\
\hline
\end{tabular}

Table 642: "Lines"
\begin{tabular}{|l|l|}
\hline "Lines style" & Clock face graduation \\
& \(\bullet \quad\) "None" \\
& \(\bullet \quad\) "Line": Graduation lines by hour \\
& \(\bullet \quad\) "Hours and minutes": Graduation lines by hours and minutes \\
\hline "Color" & Color of the clock face graduation \\
\hline "Line width" & Line weight of the clock face graduation \\
\hline "Scale in 3D" & \(\boxed{\square}:\) Representation of the clock face with 3D effect \\
\hline
\end{tabular}

Table 643: "Numerics"
\begin{tabular}{|l|l|}
\hline "Style of numerics" & \begin{tabular}{l} 
Digits on the clock face \\
\\
\\
\\
\(\bullet\) \\
\(\bullet \quad\) "None" \\
\(\bullet \quad\) "All""
\end{tabular} \\
\hline "Font" & Font for displaying the digits \\
\hline "Font color" & Font for displaying the digits \\
\hline
\end{tabular}

Table 644: "Center point"
\begin{tabular}{|l|l|}
\hline "Color" & Color of the center of the clock \\
\hline
\end{tabular}

Table 645: "Positioning"
\begin{tabular}{|l|l|}
\hline "Usage of" & \begin{tabular}{l}
\(\bullet\) \\
\(\bullet\) "Default style values": Presetting of the style values \\
"Individual settings": User-defined settings in the subordinate "Positioning" \\
property.
\end{tabular} \\
\hline "Positioning" & Requirement: Visible when the "Usage or" property is set to "Individual settings". \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Numerics movement" & \begin{tabular}{l} 
Value (in pixels) for shifting the digits. \\
Example: 80
\end{tabular} \\
\hline "Line movement" & \begin{tabular}{l} 
Value (in pixels) for shifting the hour lines. \\
Example: 100
\end{tabular} \\
\hline "Hands scaling" & \begin{tabular}{l} 
Factor for scaling the length of the hour hand. You can customize the exact \\
position of the hour hand relative to the background image. \\
Example: 100
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Scaling type" & \begin{tabular}{l} 
Defines the scaling of the height and width of the element. \\
- "Anisotropic": The background image is scaled to the size of the element The \\
height and width are scaled independently of each other. \\
"Isotropic": The background image is scaled to the size of the element, \\
retaining its proportion. The proportion of height and width is fixed.
\end{tabular} \\
\hline
\end{tabular}
"Optimized drawing"
\(\checkmark\) : The background image is drawn one time. When the hour hand moves, only the affected part of the image is redrawn.
\(\square\) : The background image is redrawn in cycles.
Hint: Disable this option only for extreme exceptions.

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" & \multicolumn{1}{|l|}{\begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular}} \\
\hline "Y" & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\ddagger\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol.
\end{tabular} \\
\begin{tabular}{l} 
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

The "X", " \(Y\) ", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also

Element prop- The variables control the element behavior dynamically.
erty 'State varia-
bles'
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

See also
- « Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Date/Time Picker'}

Symbol:


\section*{Category: "Date/Time Controls"}

The element provides the capability of selecting the date and time. The value can be changed by means of the arrow keys on the keyboard. The date can be selected from a calendar.

\section*{Element proper- \\ ties}
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Optional \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list. \\
Example: StartDateAndTime
\end{tabular} \\
\hline "Type of element" & "Date/Time Picker" \\
\hline
\end{tabular}

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X "\) & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (무-) to other positions in the editor.

See also
- *a Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline\(" X "\) & X-coordinate of the point of rotation \\
\hline\(" Y "\) & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus})\) to other positions in the editor.
\begin{tabular}{|c|c|}
\hline "Variable" & \begin{tabular}{l}
Variable (DATE, DT, TIME, LTIME, TOD) \\
The value of the value of the variable is displayed and modified by means of the element. \\
The data type automatically determines the displayed value units: \\
- TIME: Day, hour, minute, and second (by default, milliseconds are not displayed) \\
- DATE: Year, month, and day \\
- DT: Year, month, day, hour, minute, and second \\
- TOD: Hour, minute, and second (by default, milliseconds are not displayed) \\
- LTIME: Day, hour, minute, and second (by default, milliseconds, microseconds, and nanoseconds are not displayed)
\end{tabular} \\
\hline "Format string" & \begin{tabular}{l}
The format can restrict the output to individual values. \\
Example for LTIME: Format: HH:mm:ss.ms.us.ns --> displayed: 08:15:12.780.150.360 LTIME restricted: format: \(\mathrm{HH}: \mathrm{mm}\)--> displayed: 08:15 \\
Example for DATE: Format: yyyy/MM/dd --> displayed: 2015/12/17 . \\
Basically, all usual formats available for \%t are also supported.
\end{tabular} \\
\hline "Design date time picker" & \begin{tabular}{l}
- "From style": All settings are preconfigured according to the style. \\
- "Explicit": The "Design settings" property is available. You can customize the calendar here.
\end{tabular} \\
\hline "Design date picker" & \begin{tabular}{l}
- "From style": All settings are preconfigured according to the style. \\
- "Explicit": The "Design settings" property is available. You can customize the calendar here.
\end{tabular} \\
\hline "Positioning date picker" & \begin{tabular}{l}
- "Dynamic": The calendar is adapted and positioned automatically. \\
- "Manual": The "Position settings" property is available. You can customize the calendar here.
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{\sharp}\) Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329

\section*{Element property 'Absolute movement'}

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" & \multicolumn{1}{|l|}{\begin{tabular}{|l|l|}
\hline\(X "\) & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline " \(Y\) " & \begin{tabular}{l} 
Variable (numeric data type). Defines the \(Y\) position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline
\end{tabular}} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\oplus\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
"Scaling" & \begin{tabular}{l} 
Variable (integer data type). Causes centric stretching. \\
Example: PLC_PRG. iScal ing. \\
The reference point is the "Center" property. \\
The value 1 shrinks the element by a factor of 0.001. \\
to its original size. The value 1000 returns the element
\end{tabular} \\
"Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol.
\end{tabular} \\
\begin{tabular}{l} 
Note: If a static angle of rotation is specified in the \\
property "Position \(\rightarrow\) Angle", then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
\hline
\end{tabular}


The properties " \(X\) ", " \(Y\) ", "Rotation", and "Interior rotation" are supported by the "Client Animation" functionality.

\footnotetext{
Element prop- The variables control the element behavior dynamically. erty 'State variables'
}
\begin{tabular}{|l|l|}
\hline "Invisible" & Variable (BOOL). Toggles the visibility of the element. \\
& \begin{tabular}{l} 
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \(=\) FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
\\
\\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
\\
Status messages: \\
\\
\\
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
\(\bullet \quad\) "Rights are set: Limited rights": Access is restricted for at least one group. \\
\hline
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\xi^{\prime}\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

\subsection*{6.4.5.20.2 Placeholders with Format Definition in the Output Text}

A character string which is output in the visualization can include the placeholder \% for a variable. At runtime, the placeholder is replaced by the actual value of the variable in the defined format. The data type in the format definition and of the variable have to be identical. A character string can contain a maximum of one placeholder.
Character strings for output are listed in the "Text" property. The assigned variable is listed in the "Text variable" property.
See also
- Integer Data Types
- REAL/LREAL Data Type
- Time Data Types

For the output of integers
\begin{tabular}{|c|c|c|}
\hline \begin{tabular}{l}
\(\% d\) \\
\%i
\end{tabular} & Output of variable (integer data type) as decimal number & \begin{tabular}{l}
Code: iCounter : INT := 12; \\
Property "Text": Value: \%i \\
Property "Text variable": PLC_PRG.iCounter \\
Output: Value: 12
\end{tabular} \\
\hline \%.b & Output of variable (integer data type) as binary number & \begin{tabular}{l}
Code: byCode : BYTE := 255; \\
Property "Text": Coding: \%b \\
Property "Text variable": PLC_PRG.byCode \\
Output: Coding: 11111111
\end{tabular} \\
\hline \% 0 & Output of variable (integer data type) as unsigned octal number without a preceding zero & \begin{tabular}{l}
Code: byCode : BYTE := 8\#377; \\
Property "Text": Coding: \%o \\
Property "Text variable": PLC_PRG .byCode \\
Output: Coding: 377
\end{tabular} \\
\hline \% \(x\) & Output of variable (integer data type with max. 32 bits) as unsigned hexadecimal number without a preceding "0x" & \begin{tabular}{l}
Code: dwCode : INT := 16\#FFFFFFFF; \\
Property "Text": Coding: \%x \\
Property "Text variable": PLC_PRG.dwCode \\
Output: Coding: ffffffff
\end{tabular} \\
\hline \[
\begin{aligned}
& \hline \% 11 X \\
& \% 01211 X
\end{aligned}
\] & \begin{tabular}{l}
Output of 64-bit variable (LWORD, LINT, ULINT) as hexadecimal number. \\
Note: llx means "long long hexadecimal"
\end{tabular} & \begin{tabular}{l}
Code: lwCode : LWORD := 16\#4FFF_3FFF_2FFF_1FFF; \\
Property "Text": Coding: \%1lx \\
Property "Text variable": PLC_PRG. lwCode \\
Output: Coding: 4fffeffff2fffifff
\end{tabular} \\
\hline \%u & Output of variable (integer data type) as unsigned decimal number & \begin{tabular}{l}
Code: uiNumber : UINT := 1234; \\
Property "Text": Number: \%u \\
Property "Text variable": PLC_PRG.uiNumber \\
Output: Number: 1234
\end{tabular} \\
\hline
\end{tabular}

For the output Floating-point numbers have the data type REAL or LREAL. of floating-point numbers
\begin{tabular}{|c|c|c|}
\hline \%f & In decimal form with decimal point in format 1.6 & \begin{tabular}{l}
Code: rWeight : REAL := 1.123456789; \\
Property "Text": Weight: \%f \\
Property "Text variable": PLC_PRG.rWeight \\
Output: Weight: 1.123456
\end{tabular} \\
\hline ```
%<alignment><
minimum
width>.<accur
acy>f
``` & \begin{tabular}{l}
As decimal number in user-defined format \\
- <alignment>: - or +, optional \\
-: Left-aligned \\
+ : Right-aligned \\
- <minimum width>: Number of places to the left of the decimal point \\
- <accuracy>: Number of places to the right of the decimal point
\end{tabular} & \begin{tabular}{l}
Code: rWeight : REAL := 12.1 \\
Property "Text": Weight: \%2.3f \\
Property "Text variable": PLC_PRG.rWeight \\
Output: Weight: 12.100
\end{tabular} \\
\hline \%e & Output of floating-point number (REAL or LREAL) in exponential notation of base 10 & \begin{tabular}{l}
Code: rValue : REAL := 1.234567\%e-003; \\
Property "Text": Value: \%E \\
Property "Text variable": PLC_PRG.rValue \\
Output: Value: 1.23E-6
\end{tabular} \\
\hline \% E & & \begin{tabular}{l}
Code: rValue : REAL := 1.234567\%e-003; \\
Property "Text": Value: \%e \\
Property "Text variable": PLC_PRG.rValue \\
Output: Value: 1.23e-6
\end{tabular} \\
\hline
\end{tabular}

For the output of text
\begin{tabular}{|l|l|l|}
\hline\(\% \mathrm{c}\) & \begin{tabular}{l} 
Output of single character \\
in ASCII character set
\end{tabular} & \begin{tabular}{l} 
Code: bChar := 16\#41; \\
Property "Text": Key: \%c \\
Property "Text variable": PLC_PRG.bChar \\
Output: Key: A
\end{tabular} \\
\hline\(\% \mathrm{~s}\) & Output of character string & \begin{tabular}{l} 
Code: strName : = 'Paul Smith'; \\
Property "Text": Name : \%s \\
Property "Text variable": PLC_PRG.strName \\
Output: Name: Paul Smith
\end{tabular} \\
\hline
\end{tabular}

For the output of the percent sign
\begin{tabular}{|l|l|l|}
\hline\(\% \%\) & \begin{tabular}{l} 
Output of percent sign in \\
character string
\end{tabular} & \begin{tabular}{l} 
Property "Text": Valid until 90\%\% \\
Output: Valid until 90\%
\end{tabular} \\
\cline { 3 - 6 } & \begin{tabular}{l} 
Code: iPercentage : INT : \(=80 ;\) \\
Property "Text": Valid until \%d\%\%. \\
Property "Text variable": \\
PLC_PRG.iPercentage \(:=80 ;\) \\
Output: Valid until 80\%
\end{tabular} \\
\hline
\end{tabular}

For the output of the date and time

If the output text in the element "Text" property contains the placeholder "\%t", then a date and/or time is output. If a variable is not specified in the "Text variable" property, then the system time is output; otherwise it is the value of the variable.

By default, the names of the days and months are displayed in English. If localized texts are used, then the text list System has to be supplemented. This text list is created automatically in the "POUs" view when the placeholder \%t is used. The English terms have to be used as the ID here. The localization can be done for both the abbreviated names and full names.
Time data types include LTIME, TIME, TIME_OF_DAY, TOD, DATE, DATE_AND_TIME, and DT.

\section*{Compatibility Notice}

In order to get the usual display, in V3.5 SP17 and higher, as a rule three digits are used for the output of fractions of a second ( \(\mathrm{ms} / \mu \mathrm{s} / \mathrm{ns}\) ). Example: In \%t [dd-HH:mm:ss:ms], ms is specified with three digits for the milliseconds. For this purpose, the two-digit ms number is prepended with a zero. If a two-digit output is desired (like before V3.5 SP17), then a special compiler define has to be set in the compiler properties of the application:
VISU_MILLISEC_NOLEADING_ZERO.

\begin{tabular}{|c|c|c|}
\hline \%t [MMMM] & Month as full name & \begin{tabular}{l}
Code: dateMonth : DATE := DATE\#2016-1-1; \\
Property "Text": Month: \%t [MMMM] \\
Property "Text variable": \\
PLC_PRG.dateMonth \\
Output: Month: January
\end{tabular} \\
\hline \% t [MMM] & Month as abbreviated name & \begin{tabular}{l}
Code: dateMonth : DATE := DATE\#2016-1-1; \\
Property "Text":: Month : \%t [MMM] \\
Property "Text variable": \\
PLC_PRG.dateMonth \\
Output: Month: Jan
\end{tabular} \\
\hline \%t [MM] & Month as number (01-12) & \begin{tabular}{l}
Code: dateMonth : DATE := DATE\#2016-1-1; \\
Property "Text":: Month: \%t [MM] \\
Property "Text variable": \\
PLC_PRG.dateMonth \\
Output: Month: 01
\end{tabular} \\
\hline \%t [M] & Month as number (1-12) & \begin{tabular}{l}
Code: dateMonth : DATE := DATE\#2016-1-1; \\
Property "Text":: Month: \%t [M] \\
Property "Text variable": \\
PLC_PRG.dateMonth \\
Output: Month: 1
\end{tabular} \\
\hline \%t [ddddd] & Day of week as number (1=Monday - 7=Sunday) & \begin{tabular}{l}
Code: iDay : INT := 7; \\
Property "Text": Day: \%t [ddddd] \\
Property "Text variable": PLC_PRG.iDay \\
Output: Day: 7
\end{tabular} \\
\hline \%t [dddd] & Day of week as full name & \begin{tabular}{l}
Code: iDay : INT := 7; \\
Property "Text":: Day: \%t [dddd] \\
Property "Text variable": PLC_PRG.iDay \\
Output: Day: Sunday
\end{tabular} \\
\hline \%t [ddd] & Day of week as abbreviated name & \begin{tabular}{l}
Code: iDay : INT := 7; \\
Property "Text":: Day: \%t [ddd] \\
Property "Text variable": PLC_PRG.iDay \\
Output: Day: Sun
\end{tabular} \\
\hline \%t [dd] & Day of month as number (01-31) & \begin{tabular}{l}
Code: iDay : INT := 1; \\
Property "Text": Day: \%t [dd] \\
Property "Text variable": PLC_PRG.iDay \\
Output: Day: 01
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|}
\hline \%t [d] & Day of month as number (1-31) & \begin{tabular}{l}
Code: iDay : INT := 1; \\
Property "Text":: Day: \%t [d] \\
Property "Text variable": PLC_PRG.iDay \\
Output: Day: 1
\end{tabular} \\
\hline \%t[jjj] & Day of year as number (001-366) & \begin{tabular}{l}
Code: dateOfNoReturn : DATE := DATE\#2016-09-01; \\
Property "Text":: Day of no return: \%t[jjj] \\
Property "Text variable": PLC_PRG.dateOfNoReturn \\
Output: Day of no return: 245
\end{tabular} \\
\hline \%t [ HH ] & Hour in 24-hour format (00-23) & \begin{tabular}{l}
Code: todEnd : TOD := \\
TIME_OF_DAY\#17:0:0; \\
Property "Text": Ends at: \%t [HH]:00 \\
Property "Text variable": PLC_PRG.todEnd \\
Output: Ends at 17:00
\end{tabular} \\
\hline \%t [hh] & Hour in 12-hour format (01-12) & \begin{tabular}{l}
Code: todEnd : TOD := \\
TIME_OF_DAY\#17:0:0; \\
Property "Text": Ends at: \%t [hh] -'clock \\
Property "Text variable": PLC_PRG.todEnd \\
Output: Ends at: 05 o'clock
\end{tabular} \\
\hline \%t [mm] & Minutes with leading zero (00-59) & \begin{tabular}{l}
Code: tPeriod : TIME := T\#5M; \\
Property "Text": Period: \%t [mm]m \\
Property "Text variable": PLC_PRG.tPeriod \\
Output: Period: 05m
\end{tabular} \\
\hline \%t [m] & Minutes without leading zero (0-59) & \begin{tabular}{l}
Code: tPeriod : TIME := T\#5m; \\
Property "Text": Period: ot [m 'm'] \\
Property "Text variable": PLC_PRG.tPeriod \\
Output: Period: 5 m
\end{tabular} \\
\hline \%t [ss] & \[
\begin{aligned}
& \text { Seconds with leading zero } \\
& (00-59)
\end{aligned}
\] & \begin{tabular}{l}
Code: tPeriod : TIME := T\#5m3s; \\
Property "Text": Period: \%t[mm'm'ss's'] \\
Property "Text variable": PLC_PRG.tPeriod \\
Output: Period: 05m03s
\end{tabular} \\
\hline \%t [s] & Seconds without leading zero (0-59) & \begin{tabular}{l}
Code: tPeriod : TIME := T\#5m3s; \\
Property "Text": Period: ot [m'm' s's'] \\
Property "Text variable": PLC_PRG.tPeriod \\
Output: Period: 5m 3s
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|}
\hline \%t [ms] & Milliseconds without leading zero (0-999) & \begin{tabular}{l}
Code: tPeriod : TIME := T\#500ms; \\
Property "Text": Period: \%t[ms 'ms'] \\
Property "Text variable": PLC_PRG.tPeriod \\
Output: Period: 500 ms
\end{tabular} \\
\hline \%t [us] & Only for LTIME variables: microsecond definition (0999) & \multirow[t]{2}{*}{\begin{tabular}{l}
Code: ltPeriod :LTIME := LTIME\#1000D23H4 4M12S34MS2US44NS; \\
Property "Text": 'Period': \\
\%t [dd.HH.m.s.ms.us.ns] \\
Property "Text variable": PLC_PRG.ItPeriod \\
Output: Period: \\
1000.23.44.12.34.2.44 \\
Hint: Overflow is permitted in the greatest time unit of a definition.
\end{tabular}} \\
\hline \%t [ns] & Only for LTIME variables: nanosecond definition (0999) & \\
\hline \%t [t] & If the value is a time \(<12 \mathrm{~h}\), then \(A\) is output; otherwise \(P\) is output. & \begin{tabular}{l}
Code: tClosed : TOD := TOD\#17:17:17.17; \\
Property "Text": Closed at \%t[hh:mm t] \\
Property "Text variable": PLC_PRG.tClosed \\
Output: Closed at 05:17 P
\end{tabular} \\
\hline \% \(t\) [ \(t\) ] & If the value is a time \(<12 \mathrm{~h}\), then AM is output; otherwise PM is output. & \begin{tabular}{l}
Code: tClosed : TOD := TOD\#17:17:17.17; \\
Property "Text": Closed at \%t[hh:mm tt] \\
Property "Text variable": PLC_PRG.tClosed \\
Output: Closed at 05:17 PM
\end{tabular} \\
\hline \%t[' '] & If character strings should be output which correspond to a format definition, then these have to be represented in single straight quotation marks. & \\
\hline
\end{tabular}
\begin{tabular}{|l|l|l|}
\hline \%t [<f><n>] & \begin{tabular}{l} 
A number (<n>) which \\
defines the number of dec- \\
imal places of the time \\
value follows the letters \\
which define the time unit \\
(<f>). \\
As a result, the hours, \\
minutes, and seconds (for \\
TIME values) and also \\
the microseconds and \\
nanoseconds (for LTIME \\
values) can be specified or \\
displayed as values with \\
decimal places.
\end{tabular} & \begin{tabular}{l} 
Examples of the formating \\
ot [hh4] or \%t [ HH 4\(]:\) The time can be speci- \\
fied/displayed with a hour definition of four \\
decimal places. \\
ot [mm2] or \%t [m2]: The time can be \\
specified/displayed with a minute definition \\
of four decimal places. Then for a value \\
of t\#1h20m15s, this leads to the following \\
output: 80.25. \\
ot [ss0]: The time can be specified/dis- \\
played with a second definition without dec- \\
imal places.
\end{tabular} \\
\hline
\end{tabular}

See also
- Time Data Types

\subsection*{6.4.5.20.3 Methods of the Dialog Manager}

Visualizations that are a "Dialog" visualization type and are used to prompt an input are instantiated automatically and managed by the internal dialog manager.
In the application, the dialog manager can be accessed via the also internal Visualization Manager by calling the method GetDialogManager.
The dialog manager is provided with the following methods for handling a dialog.

\section*{NOTICE!}

You can program the method calls in function blocks or functions which are themselves called from the visualization by the action Execute ST Code.
Moreover, you can program the method calls in the application code. Make sure that the call runs in VISU_TASK. If this is not the case, then the behavior is undefined.

Method 'GetDia- Returns the instance (IVisualisationDialog) of the dialog whose name is passed. log' \(^{\prime}\)

Table 646: Inputs (VAR_INPUT)
\begin{tabular}{|l|l|l|}
\hline Name & Data Type & Description \\
\hline stName & STRING & Name of the dialog \\
\hline
\end{tabular}

Table 647: Outputs (VAR_OUTPUT)
\begin{tabular}{|l|l|l|}
\hline Name & Data Type & Description \\
\hline GetDialog & \begin{tabular}{l} 
VisuElems.IVisual \\
isationDialog
\end{tabular} & Instance (IVisualisationDialog) of the dialog \\
\hline
\end{tabular}

Method 'Get- Returns a pointer to the dialog structure. ClientInterface'

Respective dialog data held for each display variant.

Table 648: Inputs (VAR_INPUT)
\begin{tabular}{|l|l|l|}
\hline Name & Data Type & Description \\
\hline dialog & \begin{tabular}{l} 
VisuElems. IVisual \\
isationDialog
\end{tabular} & Name of the visualization \\
\hline pClient & \begin{tabular}{l} 
POINTER TO \\
VisuElems. IVisual \\
isationDialogVisu \\
StructClientData
\end{tabular} & Pointer to the display variant \\
\hline
\end{tabular}

Table 649: Outputs (VAR_OUTPUT)
\begin{tabular}{|l|l|l|}
\hline Name & Data Type & Description \\
\hline \begin{tabular}{l} 
GetClientInterfac \\
e
\end{tabular} & \begin{tabular}{l} 
Example: POINTER TO \\
Login_VISU_STRUCT
\end{tabular} & Pointer to the dialog structure \\
\hline
\end{tabular}

Method 'Open- Opens the dialog of the client. Dialog'

Next to it, there is the extended method 'OpenDialog(number)'.

Table 650: Inputs (VAR_INPUT)
\begin{tabular}{|l|l|l|}
\hline Name & Data Type & Description \\
\hline dialog & \begin{tabular}{l} 
VisuElems. IVisual \\
isationDialog
\end{tabular} & Name of the visualization \\
\hline pClient & \begin{tabular}{l} 
POINTER TO \\
VisuElems.VisuStr \\
uctClientData
\end{tabular} & \\
\hline
\end{tabular}
\begin{tabular}{|l|l|l|}
\hline Name & Data Type & Description \\
\hline bModal & BOOL & \\
\hline pRect & POINTER TO & \\
\hline
\end{tabular}

Table 651: Outputs (VAR_OUTPUT)
\begin{tabular}{|l|l|l|}
\hline Name & Data Type & Description \\
\hline OpenDialog & & \\
\hline
\end{tabular}

Method 'Close- Closes the dialog of the client. Dialog'

Table 652: Inputs (VAR_INPUT)
\begin{tabular}{|l|l|l|}
\hline Name & Data Type & Description \\
\hline dialog & \begin{tabular}{l} 
VisuElems.IVisual \\
isationDialog
\end{tabular} & Dialog object as received by GetDialog \\
\hline pClient & \begin{tabular}{l} 
POINTER TO \\
VisuElems.VisuStr \\
uctClientData
\end{tabular} & \\
\hline
\end{tabular}

Table 653: Outputs (VAR_OUTPUT)
\begin{tabular}{|l|l|l|}
\hline Name & Data Type & Description \\
\hline CloseDialog & & \\
\hline
\end{tabular}

Method 'Close- Closes the dialog of the client. Extension of the method CloseDialog. Dialog2'

Table 654: Inputs (VAR_INPUT)
\begin{tabular}{|l|l|l|}
\hline Name & Data Type & Description \\
\hline dialog & \begin{tabular}{l} 
VisuElems. IVisual \\
isationDialog
\end{tabular} & Dialog object as received by GetDialog \\
\hline pClient & \begin{tabular}{l} 
POINTER TO \\
VisuElems.VisuStr \\
uctClientData
\end{tabular} & DWORD
\end{tabular} \begin{tabular}{l} 
Specification of possible options for closing the dialogs. Only \\
the values 0 (behavior as for CloseDialog) and 16\#40 are \\
relevant in the case that a dialog should be closed on all \\
connected clients.
\end{tabular}

Table 655: Outputs (VAR_OUTPUT)
\begin{tabular}{|l|l|l|}
\hline Name & Data Type & Description \\
\hline CloseDialog2 & & \\
\hline
\end{tabular}

\subsection*{6.4.5.20.4 Attribute 'VAR_IN_OUT_AS_POINTER'}

Function: The pragma \{attribute 'VAR IN OUT AS POINTER'\} allows for the passing of a reference to a data object to the interface variable of a visualization.

Requirement: The referenced visualization must be used as a dialog.

\section*{Syntax:}
```

{attribute 'VAR IN OUT AS POINTER'}

```

\section*{NOTICE!}

Uppercase and lowercase characters must be maintained.

Example: Declaration of an interface
```

VAR_IN_OUT
_{a\overline{trribute 'VAR_IN_OUT AS_POINTER'}}
itfController : - Con}tro\overline{lFB;
END_VAR

```

See also
- 出 Chapter 6.4.5.17.5 "Calling a Dialog with an Interface" on page 2962
- « Chapter 6.4.5.21.2.1 "Command 'Interface Editor'" on page 3340

\subsection*{6.4.5.20.5 Attribute 'parameterstringof'}

The pragma \{attribute 'parameterstringof'\} allows that the instance name of the specified parameter is made accessible for the referenced visualization. An interface variable (STRING) will contain the instance name of the specified parameter. The interface variable is visible within the referenced visualization and can for example be used in a text output.

\section*{Syntax:}
\{attribute 'parameterstringof' := '<variable>'\}
```

Example: decla-
ration of a inter- VAR_INPUT
face {attribute 'parameterstringof' := 'iftDut_A'}
sItfNameDut_A: STRING;
END VAR
VAR_IN_OUT
i\overline{f}tDut_A : DUT_A;
END_VAR

```

See also
- \#y Chapter 6.4.5.17.3 "Calling a Visualization with an Interface" on page 2951
- \#y Chapter 6.4.5.21.2.1 "Command 'Interface Editor'" on page 3340
6.4.5.21 Reference, user interface
6.4.5.21.1 Keyboard Shortcuts for Default Keyboard Action.......................... 3338
6.4.5.21.2 Commands..................................................................................... 3339
6.4.5.21.3 Dialog Boxes................................................................................. 3366
6.4.5.21.4 Objects....................................................................................... 3393
6.4.5.21.5 Visualization Elements.................................................................. 3412

\subsection*{6.4.5.21.1 Keyboard Shortcuts for Default Keyboard Action}

Requirement: The "Activate default keyboard handling" option is activated in the "Visualization Manager" object.

The keyboard shortcuts for default keyboard action make it possible for users to operate the visualization with the keyboard only. Elements that respond to user input can process a keyboard event instead of a mouse event. You do not have to change their input configuration for this purpose. The universal keyboard shortcuts are supported by all devices and are available on all display variants when needed.
\begin{tabular}{|l|l|}
\hline Keyboard shortcuts & \\
\hline [Tab] & \begin{tabular}{l} 
Focus jumps to the next element. \\
The next element that responds to a configured or preconfigured user input receives the \\
focus. The order of elements corresponds to the order that the elements were added to the \\
editor. \\
If the focused element is a table, then the upper left cell in the table is the next focus. After \\
that, each next cell until all cells have been focused. It also applies here that only cells that \\
require input are focused. \\
If the focused element is a frame, then an element of the referenced visualization is set \\
next in focus in the frame. After that, each next element until all elements have been \\
focused. It also applies here that only elements that require input are focused.
\end{tabular} \\
\hline [Shift]+[Tab] & \begin{tabular}{l} 
Focus jumps to the previous element. \\
The element is focused that is before the currently focused element in the added order. \\
Therefore, the order is the opposite as for "Tab".
\end{tabular} \\
\hline [Arrow] & The focus jumps to the element that is in the direction as indicated by the arrow. \\
\hline [Input] & The visualization detects the input at the focused element and triggers the input action. \\
\hline
\end{tabular}

\subsection*{6.4.5.21.2 Commands}
6.4.5.21.2.1 Command 'Interface Editor'. ..... 3340
6.4.5.21.2.2 Command 'Keyboard Configuration'. ..... 3341
6.4.5.21.2.3 Command 'Visualization Element List' ..... 3342
6.4.5.21.2.4 Command 'Activate Keyboard Usage' ..... 3343
6.4.5.21.2.5 Command 'Order'. ..... 3344
6.4.5.21.2.6 Command 'Alignment'. ..... 3344
6.4.5.21.2.7 Command 'Group' ..... 3347
6.4.5.21.2.8 Command 'Ungroup'. ..... 3348
6.4.5.21.2.9 Command 'Frame Selection'. ..... 3348
6.4.5.21.2.10 Command 'Background'. ..... 3349
6.4.5.21.2.11 Command 'Multiply Visu Element'. ..... 3350
6.4.5.21.2.12 Command 'Configure Display Settings of Trend'. ..... 3353
6.4.5.21.2.13 Command 'Configure Trace'. ..... 3355
6.4.5.21.2.14 Command 'Export Trace Configuration'. ..... 3357
6.4.5.21.2.15 Command 'Insert Elements for Controlling Trace'. ..... 3358
6.4.5.21.2.16 Command 'Configure Display Settings of Trend'. ..... 3359
6.4.5.21.2.17 Command 'Edit Trend Recording' ..... 3360
6.4.5.21.2.18 Command 'Insert Elements for Controlling the Trend'. ..... 3360
6.4.5.21.2.19 Command 'Visualization Element Repository'. ..... 3361
6.4.5.21.2.20 Command 'Visualization Style Repository'. ..... 3363
6.4.5.21.2.21 Command 'Add Visual Element'. ..... 3364
6.4.5.21.2.22 Command 'Select None'. ..... 3365
6.4.5.21.2.23 Command 'Add Elements for Alarm Acknowledgement'. ..... 3365

\section*{Command 'Interface Editor'}

Symbol: ه; keyboard shortcut: [Alt]+[F6].
Function: The command opens and closes the "Interface Editor" tab above the visualization editor.

Call: Menu bar: "Visualization \(\rightarrow\) Interface Editor" Also by clicking on the small down arrow at the top of the visualization editor

\section*{Tab 'Interface}

Symbol:
Editor'
The tab contains an editor for the declaration of interface variables. The editor behaves in a similar way to the declaration editor of a function block, however interface variables are not initialized.

\section*{Syntax}
```

<scope>
( {attribute '<attribute name>' ( := '<expression>' )? } )?
<identifier> : <data type>;
END_VAR
<scope> : VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT
// (...)? : Optional

```

\section*{Example}

Declaration in
the interface
```

VAR_INPUT

```
editor - \{attribute 'parameterstringof'\}
    sIdentifier : STRING; // String for instance name
            iCounter : INT;
END_VAR
VAR_IN_OUT
            \{ātribute 'VAR_IN_OUT_AS_POINTER'\}
            fbController: \(F \bar{B}\) _Controllér;
END_VAR

Scopes Possible scopes for interfaces of visualizations or dialogs
\begin{tabular}{|l|l|}
\hline VAR_IN_OUT & \begin{tabular}{l}
\(\bullet\)\begin{tabular}{l} 
When transferring a structure \\
When the visualization is instanced, it gets a reference to the current applica- \\
tion data. \\
When transferring a control variable, if the variable is written to when a user \\
input is made. Only then can the visualization write to it. \\
Note: In the case of dialogs, the data is written back only when the dialog is \\
closed. \\
Hint: We strongly recommend that you use this scope so that the return of values \\
is possible. Moreover, no data needs to be copied.
\end{tabular} \\
\hline \begin{tabular}{l} 
PVAR_IN_OUT \\
Pragma \{attribute \\
'VAR_IN_OUT_AS_POINTER
\end{tabular} \\
\hline When transferring a pointer to a data object \\
In contrast to the VAR_IN_OUT scope (without an attribute), the variable \\
changes are effective immediately and not just when the dialog is closed. \\
Note: Use this scope only if the visualization implements a Dialog.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline VAR_INPUT & \begin{tabular}{l} 
When transferring data that will only be read \\
Note:
\end{tabular} \\
\hline \begin{tabular}{l} 
•If the visualization is executed as an integrated visualization, then only input \\
variables of a basic data type (scalar type) are permitted to be transferred. \\
If the visualization is executed as a CODESYS TargetVisu or a CODESYS \\
Wragma \{attribute \\
WebVisu, then input variables of any data type (including POUs) can also be \\
transferred.
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{4}\) Chapter 6.4.1.9.4 "Declaration of Variables" on page 1847
- * Chapter 6.4.1.20.1.2 "Declaration Editor" on page 2047
- \& Chapter 6.4.5.17.3 "Calling a Visualization with an Interface" on page 2951
- « Chapter 6.4.5.17.5 "Calling a Dialog with an Interface" on page 2962
- を Chapter 6.4.5.20.5 "Attribute 'parameterstringof"' on page 3338
- \({ }^{2}\) Chapter 6.4.5.20.4 "Attribute 'VAR_IN_OUT_AS_POINTER"' on page 3337

\section*{Command 'Keyboard Configuration'}

Symbol: \&; keyboard shortcut: [AIt]+[F6].
Function: This command opens and closes the "Keyboard Configuration" tab above the visualization editor.

Call: Menu bar: "Visualization".
Requirement: A visualization is open and active in the visualization editor.
See also
- \({ }^{\Perp}\) Chapter 6.4.5.21.4.2 "Object 'Visualization manager'" on page 3398

\section*{Tab 'Keyboard configuration'}

Symbol:
This tab contains a list of keyboard shortcuts with an editing option.
A keyboard shortcut can refer specifically to an element. Then the configuration appears here and in the "Input configuration" property of the associated element.
A keyboard shortcut can also have several configurations. If a keyboard shortcut has multiple keyboard configurations, then its input actions are executed in the order listed here.

Keyboard shortcuts of the default keyboard action are not listed here.
\begin{tabular}{|c|c|}
\hline "Key" & \begin{tabular}{l}
Key that a keyboard configuration is defined. Example: [M] \\
Note: You can combine the key with [Ctrl], [Alt], and/or [Shift].
\end{tabular} \\
\hline "Key down" & \begin{tabular}{l}
The input action is executed when the user presses the key.
: The input action is executed when the user releases the key. \\
Double-click: Drop-down list of all keys. \\
Note: If the input action should be executed for both pressing the key (KeyDown) and releasing the key (KeyUp), then you must define a keyboard configuration for both input actions.
\end{tabular} \\
\hline "Shift" & \(\checkmark\) : The input event is triggered for [Shift]+[key]. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Ctrl" & \(\boxed{\square}\) : The input event is triggered for [Ctrl]+[key]. \\
\hline "Alt" & \(\boxed{\square}:\) The input event is triggered for \([\) Alt \(]+[k e y]\). \\
\hline "Action type" & \begin{tabular}{l} 
input action \\
Double-click: Drop-down list of input actions. \\
Tip: For a description of input actions, refer to the "Input configuration" dialog \\
box.
\end{tabular} \\
\hline "Action" & \begin{tabular}{l} 
Configuration of the input action that was selected next. \\
Double-click: A dialog box opens that varies according to the input action. It \\
allows the user-prompted customization of the settings. \\
Tip: For a description of dialog boxes, refer to the "Input configuration" dialog \\
box. The input action is configured in the same way here.
\end{tabular} \\
\hline "Element ID" & \begin{tabular}{l} 
ID of the visualization element where the user can execute the key event. The ID \\
is relevant only if the event is also assigned to an element. \\
Tip: The assignment of ID to element name is listed in the "Element list".
\end{tabular} \\
\hline "Access rights" & \begin{tabular}{l} 
Access privileges of the action per user group \\
Requirement: The visualization has a user management.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline\(\downarrow\) & Clicking the symbol on the right of the list moves the selected row one line down. \\
\hline \(\boldsymbol{\uparrow}\) & Clicking the symbol on the right of the list moves the selected row one line up. \\
\hline Blank line & Allows adding a new keyboard configuration. \\
\hline
\end{tabular}

See also
- \& Chapter 6.4.5.21.2.3 "Command 'Visualization Element List"' on page 3342
- © Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration"' on page 3370

\section*{Command 'Visualization Element List'}

Symbol: 四
Function: The command opens the "Visualization Element List" tab for the current visualization. It is displayed in the upper part of the visualization editor.

Call: Menu bar: "Visualization"
Requirement: A visualization is open in the editor.

Tab 'Visualiza- This view contains a list of the visualization elements in the open visualization. Grouped tion Element List' elements are displayed in a tree structure and have their own order within the group (other hierarchy level).

The current selection in the list is always synchronized with the selection in the main window of the editor.

The order in the element list from top to bottom describes the order of the elements on the display layers of the visualization from back to front. When you insert elements consecutively, they are arranged starting from the back (position 0 ) on one layer forward. When you use the commands in the menu "Visualization \(\rightarrow\) Order" to move an element from front to back in the editor window, the element list refreshes accordingly.
\(\left.\begin{array}{|l|l|}\hline \text { "Type" } & \begin{array}{l}\text { Element type and symbol, as used in the "Visualization Toolbox" view, as well as } \\ \text { the element number that specifies the display layer. \# } 0 \\ = \\ \hline \text { "X", layer furthest back. }\end{array} \\ \hline \text { "Y" } & \begin{array}{l}\text { Position of the upper left corner of the element }(0,0=\text { upper left corner of the } \\ \text { visualization area). }\end{array} \\ \hline \text { "Width" } & \text { Dimensions of the element (in pixels). } \\ \hline \text { "Height" } & \text { "ID" } \\ \hline \text { "Name" } & \text { Internally assigned element identifier } \\ \hline \text { "Access Rights" } & \text { Element name as defined in "Properties } \rightarrow \text { Element name". } \\ \hline \text { "Tab Order" } & \begin{array}{l}\text { The lock symbol B indicates the restricted behavior of an element for some user } \\ \text { groups. }\end{array} \\ \hline \begin{array}{l}\text { Position within the order in which you can jump from element to element in the } \\ \text { editor by means of the tab key when the default keyboard usage is activated. } \\ \text { The activation is done in the visualization manager, on the settings tab. Note that } \\ \text { elements within a group or group box have their own order (different hierarchy } \\ \text { level). } \\ \text { The tab order initially corresponds to the order in which the elements are }\end{array} \\ \text { arranged on the layers from back to front ("Type" above). To change the position } \\ \text { in the order for an element, you can specify a different number directly in the } \\ \text { table field. You can also use the "Move to Position" context menu command to } \\ \text { open a dialog for specifying a new position. } \\ \text { Bold fonts indicate changed position specifications. } \\ \text { By removing the displayed value, you exclude the element from the selection } \\ \text { using tab or arrow keys. } \\ \text { You can use the "Reset to Default" context menu command to reset a changed } \\ \text { position to the original position. This can be done simultaneously for a multise- } \\ \text { lection of elements when they do not belong to different hierarchy levels (group- } \\ \text { ings). }\end{array}\right\}\)

See also
- \({ }^{4}\) Chapter 6.4.5.21.4.1.1 "Visualization Editor" on page 3393
- \({ }^{*} \gg\) Chapter 6.4.5.21.2.5 "Command 'Order'" on page 3344
- (3) "Moving the visualization element forward and back" on page 2875

\section*{Command 'Activate Keyboard Usage'}

Symbol:
Function: This command activates and deactivates the keyboard usage when a visualization is executed in online mode (integrated in CODESYS).

Call: Menu bar: "Visualization"; context menu.
Requirement: A visualization is open.
When this command is active, the visualization executes the keyboard events that you specified as a visualization user.
When the command is inactive, CODESYS executes the keyboard events that you specify.
See also
- \({ }^{\leftrightarrows}\) Chapter 6.4.5.6.5 "Configuring Keyboard Shortcuts" on page 2892

\section*{Command＇Order＇}

Function：The command makes further commands available．They are for specifying the order of the elements in levels，since elements in the rear levels are concealed by those in the front levels．

Call：Menu＂Visualization＂，context menu
Requirement：The visualization elements are positioned behind one another．
See also
－\({ }^{〔}\) Chapter 6．4．5．5．3＂Positioning the Element，Adapting Size and Layer＂on page 2874

\section*{Command
＇Bring to Front＇}

Symbol：
Function：The command positions the selected visualization element in the front level．The element becomes completely visible．

Call：Menu＂Visualization \(\rightarrow\) Order＂，context menu
\begin{tabular}{ll} 
Command & Symbol：\(\square\) \\
＇Bring One to & Function：The command positions the selected visualization element one level further forwards． \\
Front＇ & Call：Menu＂Visualization \(\rightarrow\) Order＂，context menu
\end{tabular}

\section*{Command＇Send Symbol：喕 \\ to Back＇}

Function：The command positions the selected visualization element in the back level．
Call：Menu＂Visualization \(\rightarrow\) Order＂，context menu

\section*{Command＇Send Symbol：咟}

One to Back＇
Function：The command positions the selected visualization element one level further back－ wards．

Call：Menu＂Visualization \(\rightarrow\) Order＂，context menu

\section*{Command＇Alignment＇}

Function：the command makes further commands available．It is used for the alignment of visualization elements in the window area of the visualization．

Call：Menu＂Visualization＂，context menu
See also
－\({ }^{\Perp}\) Chapter 6．4．5．5．3＂Positioning the Element，Adapting Size and Layer＂on page 2874

\section*{Command＇Align Symbol：唈 \\ Left＇}

Function：the command aligns the selected visualization elements along a line through the left－hand edge of the element that is positioned furthest left．

Call：Menu＂Visualization \(\rightarrow\) Alignment＂，context menu
Requirement：Several elements are selected．

\footnotetext{
Command＇Align Symbol：可
Top＇
Function：the command aligns the selected visualization elements along a line through the upper edge of the element that is positioned highest．
}

Call：Menu＂Visualization \(\rightarrow\) Alignment＂，context menu
Requirement：Several elements are selected．

Command＇Align Symbol：\＃
Right＇
Function：the command aligns the selected visualization elements along a line through the right－hand edge of the element that is positioned furthest right．

Call：Menu＂Visualization \(\rightarrow\) Alignment＂，context menu
Requirement：Several elements are selected．

Command＇Align Symbol：颌
Bottom＇
Function：the command aligns the selected visualization elements along a line through the lower edge of the element that is positioned lowest．

Call：Menu＂Visualization \(\rightarrow\) Alignment＂，context menu
Requirement：Several elements are selected．

Command＇Align Symbol：串
Function：the command aligns the selected visualization elements to their common vertical center．
Call：Menu＂Visualization \(\rightarrow\) Alignment＂，context menu
Requirement：Several elements are selected．

Command＇Align Symbol：吗
Horizontal Cen－ ter＇

Function：The command aligns the selected visualization elements to their common horizontal center．
Call：Menu＂Visualization \(\rightarrow\) Alignment＂，context menu
Requirement：Several elements are selected．

Command＇Make Symbol：哳
Horizontal
Spacing Equal＇
Function：The command aligns the selected visualization elements so that the elements posi－ tioned furthest left and furthest right retain their position and the elements between them are positioned with the same horizontal spacing．

Call：Menu＂Visualization \(\rightarrow\) Alignment＂，context menu
Requirement： 3 or more elements are selected．The first element is blue，while the other elements are displayed in grey．

Command＇In－Symbol：
crease Hori－
zontal Spacing＇
Function：The command aligns the selected visualization elements so that the blue element retains its position and the other elements are positioned with a larger horizontal spacing．The spacing increases by 1 pixel each time．

Call：Menu＂Visualization \(\rightarrow\) Alignment＂，context menu
Requirement：Several elements are selected．

Command＇De－Symbol：\(\underset{++\infty}{\square 0}\) crease Hori－ zontal Spacing＇

Function：The command aligns the selected visualization elements so that the blue element retains its position and the other elements are positioned with a smaller horizontal spacing．The spacing decreases by 1 pixel each time．
Call：Menu＂Visualization \(\rightarrow\) Alignment＂，context menu
Requirement：Several elements are selected．
\begin{tabular}{ll}
\begin{tabular}{ll} 
Command＇Re－ \\
move Horizontal & Symbol：\(\stackrel{\text { on }}{++}\) \\
Spacing＇
\end{tabular} & \begin{tabular}{l} 
Function：The command aligns the selected visualization elements so that the blue element \\
retains its position and the other elements are positioned with no horizontal spacing between \\
them．
\end{tabular} \\
& Call：Menu＂Visualization \(\rightarrow\) Alignment＂，context menu \\
& Requirement：Several elements are selected．
\end{tabular}
\begin{tabular}{ll} 
Command＇Make & Symbol：宮 \\
Vertical Spacing & Function：The command aligns the selected visualization elements so that the uppermost and \\
Equal＇ & \\
& \begin{tabular}{ll} 
lowermost elements retain their position and the elements between them are positioned with the \\
& same vertical spacing．
\end{tabular} \\
& Call：Menu＂Visualization＂，context menu
\end{tabular}

Call：Menu＂Visualization＂，context menu
Requirement： 3 or more elements are selected．The first element is blue，while the other elements are displayed in grey．

Command＇In－ crease Vertical Spacing＇

Symbol：品 \(\downarrow\)
Function：The command aligns the selected visualization elements so that the blue element retains its position and the other elements are positioned with a larger vertical spacing．The spacing increases by 1 pixel each time．
Call：Menu＂Visualization \(\rightarrow\) Alignment＂，context menu
Requirement：Several elements are selected．

Command＇De－ crease Vertical Spacing＇

Symbol：㗊 \(\ddagger\)
Function：The command aligns the selected visualization elements so that the blue element retains its position and the other elements are positioned with a smaller vertical spacing．The spacing decreases by 1 pixel each time．
Call：Menu＂Visualization \(\rightarrow\) Alignment＂，context menu
Requirement：Several elements are selected．
\begin{tabular}{ll} 
Command＇Re－ & Symbol：\(g_{\uparrow}^{*}\) \\
move Vertical & \begin{tabular}{l} 
Function：The command aligns the selected visualization elements so that the blue element \\
Spacing＇ \\
retains its position and the other elements are positioned with no horizontal spacing between \\
them．
\end{tabular} \\
& Call：Menu＂Visualization \(\rightarrow\) Alignment＂，context menu
\end{tabular}

Requirement：Several elements are selected．

\section*{Command＇Make Symbol：：}

Same Width＇
Function：The command makes the width of the selected visualization elements the same as the width of the blue selected element．

Call：Menu＂Visualization \(\rightarrow\) Alignment＂，context menu

Requirement: Several elements are selected. The first element is blue, while the other elements are displayed in grey.


The command does not work with lines or polygons.

\section*{Command 'Make Symbol:}

\section*{Same Height'}

Function: The command makes the height of the selected visualization elements the same as the height of the blue selected element.
Call: Menu "Visualization \(\rightarrow\) Alignment", context menu
Requirement: Several elements are selected. The first element is blue, while the other elements are displayed in grey.

The command does not work with lines or polygons.

\section*{Command 'Make Symbol: 䩿}

Same Size'
Function: The command makes the size of the selected visualization elements the same as the size of the blue selected element.

Call: Menu "Visualization \(\rightarrow\) Alignment", context menu
Requirement: Several elements are selected. The first element is blue, while the other elements are displayed in grey.


The command does not work with lines or polygons.

Command 'Size Symbol: 学
to Grid'
Function: The command aligns the size and position of the selected visualization elements to the grid.

Call: Menu "Visualization \(\rightarrow\) Alignment", context menu
Requirement: Several elements are selected.


The command does not work with lines or polygons.

\section*{Command 'Group'}

Symbol:
Function: The command groups the selected visualization elements and displays them as one.
Call: Menu "Visualization", context menu

Requirement: At least 2 elements are selected.
To select more elements you can drag a window around the desired elements with the mouse. Alternatively you can click on the desired elements while keeping the [Shift] key pressed.

To select all elements you can open the context menu of the visualization editor and choose the "Select All" command.

You can also drag and drop elements to a group. For that, press the [Shift] key while dragging the element to the group. Meanwhile the cursor changes its appearance (display a small plus sign).

See also
- \(y^{\wedge}\) Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874
- \({ }^{4}\) Chapter 6.4.5.21.2.8 "Command 'Ungroup'" on page 3348
- \(\Leftrightarrow\) Chapter 6.4.5.21.2.22 "Command 'Select None'" on page 3365

\section*{Command 'Ungroup'}

Symbol: 迹
Function: The command ungroups elements again.
Call: Menu "Visualization", context menu
Requirement: A grouping is selected.
See also
- \({ }^{\text {® }}\) Chapter 6.4.5.21.2.7 "Command 'Group'" on page 3347

\section*{Command 'Frame Selection'}

Function: The command opens the "Frame Configuration" dialog.

\section*{Call:}
- Menu bar: "Visualization"
- Click the "Configure" button in the "References" property.

Requirement: A "Frame" element or "Tabs" element is selected in the editor. The "Element Properties" view is open.

Dialog 'Frame
Configuration'
The dialog allows you to select one or more of all available visualizations. The selected visualizations are displayed at runtime in the window area of the "Frame" element or "Tabs" element.

\section*{NOTICE!}

Visualizations can be nested at any depth by means of "Frame" elements. In order to use the "Switch to any visualization" frame selection type without any problems, a "Frame" must not contain more than 21 referenced visualizations.
For more information, see also the description for the "Input configuration" of an element: Action "Switch frame visualization".

Table 656：＂Available Visualizations＂
\begin{tabular}{|l|l|}
\hline ＂By Visualization Name＂ & \begin{tabular}{l} 
The list of available visualizations of the project and libraries is sorted alphabeti－ \\
cally．
\end{tabular} \\
\hline ＂By Type or Instance＂ & \begin{tabular}{l} 
The list of available visualizations of the project and libraries is sorted by type or \\
instance．
\end{tabular} \\
\hline Input field for a filter & \begin{tabular}{l} 
If a filter text is specified，then only those visualizations whose names contain \\
the filter text are listed．
\end{tabular} \\
\hline 复 & Project with project visualizations below it \\
\hline \(\mathbf{0}\) & Library with project visualizations below it \\
\hline
\end{tabular}

Table 657：＂Selected Visualizations＂
\begin{tabular}{|c|c|}
\hline 凅＂Add＂ & \begin{tabular}{l}
Click the symbol to add a visualization to the list of selected visualizations． Requirement：This is selected in＂Available Visualizations＂． \\
Hint：To add a visualization，double－click a visualization in＂Available Visualizations＂．
\end{tabular} \\
\hline \(\times\)＂Delete＂ & Click the symbol to delete a visualization from the list． Requirement：This is selected in＂Selected Visualizations＂． \\
\hline \multicolumn{2}{|l|}{\begin{tabular}{l}
The visualizations are automatically numerically indexed via the order in the list．The top visualization has the index 0 ．The next visualization has the index 1 and so on． \\
Note：A＂Frame＂and a＂Tabs＂element use the variables specified in the index of the＂Switch frame variable＂ property．
\end{tabular}} \\
\hline ＋＂Move Up＂ & Click the symbol to move a visualization up in the list． Requirement：This is selected in＂Selected Visualizations＂． \\
\hline ＋＂Move Down＂ & Click the symbol to move a visualization down in the list． Requirement：This is selected in＂Selected Visualizations＂． \\
\hline
\end{tabular}

See also
－\({ }^{*} \gg\) Chapter 6．4．5．21．5．6＂Visualization Element＇Frame＂＂on page 3478
－乡 Chapter 6．4．5．21．5．10＂Visualization Element＇Tabs＇＂on page 3509
－«＂Element property＇Switch frame variable＂＂on page 3292

\section*{Command＇Background＇}

Symbol：覑
Function：The dialog＂Background＂opens．You can define here whether the background of the visualization is colored or displayed with an image．
Call：Menu＂Visualization＂，context menu
See also
－\({ }^{4}\) Chapter 6．4．5．5．8＂Designing a background＂on page 2884

\section*{Dialog＇Back－ ground＇}

Table 658：＂Color Settings＂
\begin{tabular}{|l|l|}
\hline ＂Use Color＂ & \begin{tabular}{l} 
V：Background in color \\
Color defined as a style color or as a fixed value． \\
\hline
\end{tabular} \\
\hline
\end{tabular}

Table 659: "Image Setting"
\begin{tabular}{|c|c|}
\hline "Use Image" & \begin{tabular}{l}
Display of a background image \\
Reference to an image from an image pool in the project, formally specified as an instance path: <Name of the image pool>.<ID> \\
Example: \\
- ImagePool_A.Factory \\
- ImagePool_B.ID_B
\end{tabular} \\
\hline
\end{tabular}

\section*{Command 'Multiply Visu Element'}

Symbol:
Function: The command opens the "Multiply Visu Element" dialog, which contains a configuration derived from the template element and the array declaration. You can rearrange the elements here, as well as their quantity and the index access to the array data. When you exit the dialog, a field of similar elements is created from the template element. In the properties of the new elements, array variables are now configured with precise array indexes. These new elements are those in which you have configured an array variable with index access placeholders in the template.

Call: Menu bar: "Visualization"; context menu
Requirement: The visualization is active and a configured template element is selected.

Dialog 'Multiply
Visu Element'
Table 660: Tab "Basic Settings"
\begin{tabular}{|c|c|}
\hline "Total number of elements" & The total number is determined by the index range of the placeholders, including the setting on the "Advanced Settings" tab. The layout of the elements can be one-dimensional (as a column or row) or two-dimensional (as a table field). \\
\hline "Horizontal" & \begin{tabular}{l}
Number of elements per row \\
Default: Number of array components (index range) of the placeholder \$FIRSTDIM\$ \\
Example for array: axLampIsOn: ARRAY[0..4] OF BOOL; = 5
\end{tabular} \\
\hline "Vertical" & \begin{tabular}{l}
Number of rows required for the layout of all elements \\
Default \\
- When using index access placeholder \$FIRSTDIM\$: If the index range of the placeholder is less than five, then the layout of elements is horizontal. If the index range is greater than five, then the layout the elements is quadratic whenever possible. \\
- When using index access placeholders \$FIRSTDIM\$ and \$SECONDDIM\$: The number of horizontal elements is equal to the number of index ranges specified by the placeholder \$FIRSTDIM\$. The number of vertical elements is equal to the number of index ranges specified by the placeholder \$SECONDDIM\$.
\end{tabular} \\
\hline "Offset between elements" & \begin{tabular}{l}
Distance between the new elements; affects the positions of the new elements \\
- " 0 ": The frames of the elements overlap by one pixel. \\
- "1": The elements touch. \\
- " \(<n>\) ": A distance of \(n-1\) pixel is visible between the elements.
\end{tabular} \\
\hline "Horizontal" & \begin{tabular}{l}
Distance between the elements within a row (in pixels) \\
Example: 2 for a distance of one pixel
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Vertical" & \begin{tabular}{l} 
Distance between the elements within the columns (in pixels) \\
Example for a distance of three pixels: 4
\end{tabular} \\
\hline "Arrangement of elements" & \begin{tabular}{l} 
Origin from which the new elements are positioned and arranged \\
If "Vertical" or "Horizontal" <> 1 \\
- "From top left" \\
- "From top right" \\
- "From bottom left" \\
- "From bottom right" \\
If "Horizontal" or "Vertical" \(=1\) \\
- "From top" \\
- "From bottom"
\end{tabular} \\
\hline "Orientation" & \begin{tabular}{l} 
Determines the layout of the elements in the field (row by row, or column by \\
column) \\
- "Line by line" \\
- "Column by column"
\end{tabular} \\
\hline & \begin{tabular}{l} 
"Preview"
\end{tabular} \\
\hline
\end{tabular}

Table 661: Tab "Advanced Settings"
\begin{tabular}{|l|l|}
\hline "Array access" & \begin{tabular}{l} 
Based on the template element, the precise index for accessing the array vari- \\
able is calculated for each new element. The calculation is based on the array \\
index limits as specified in the array declaration. The settings are also taken into \\
account here.
\end{tabular} \\
\hline "1st dimension" & \begin{tabular}{l} 
Calculation guideline for the index of the first dimension that replaces \\
\$FIRSTDIMS \\
The first new element gets the value specified below in "Start index" in the first \\
dimension. The other elements each get an index incremented by "Increment" \\
until an index is calculated for all elements. \\
Example \\
- "Start index": 1 \\
- "Increment": 1
\end{tabular} \\
\hline "2nd dimension" & \begin{tabular}{l} 
Calculation guideline for the index of the second dimension that replaces \\
\$SECONDDIM\$ \\
The first new element gets the value specified below in "Start index" in the \\
second dimension. The other elements each get an index incremented by \\
"Increment". \\
Example \\
- "Start index": 1 \\
- "Increment": 1
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "OK" & \begin{tabular}{l} 
First, it is validated whether the calculated indices are in the index range of the \\
array variable. If so, then the elements that match the template element are \\
created and arranged as a field (row, column, or table). The placeholder indexes \\
are replaced by the calculated indexes.
\end{tabular} \\
\hline
\end{tabular}
```

Example
Declaration of
array variables VAR
asTexts_Example: ARRAY[1..2,1..2] OF STRING :=
[
'1A Text', '2A Text',
'1B Text', '2B Text'
];
asToolTips Example: ARRAY[1..2,1..2] OF STRING :=
[
'1A Tooltip', '2A Tooltip',
'1B Tooltip', '2B Tooltip'
];
axUserInput_Example: ARRAY[1..2,1..2] OF BOOL;
END_VAR

```

Visualization with template element and its property configuration


Table 662: Dialog 'Multiply Visu Element'
\begin{tabular}{|l|l|}
\hline Tab "Basic Settings" & \\
\hline "Total number of elements" & \\
\hline "Horizontal" & 2 \\
\hline "Vertical" & 2 \\
\hline & \\
\hline
\end{tabular}


See also
- \({ }^{\circ}\) Chapter 6.4.5.11.3 "Configuring and Multiplying Visualization Elements as Templates" on page 2918

\section*{Command 'Configure Display Settings of Trend'}

Symbol:
Function: When you execute this command in "Visualization" or in the context menu, the "Edit Display Settings" dialog opens

\section*{Call:}
- Menu bar: "Visualization"
- Context menu of a "Trend" element in the visualization editor
- Property "Diagram"

Requirement: A trend is selected in the active visualization editor.

Tab "X Axis"
\begin{tabular}{|l|l|}
\hline "Grid" & \(\boxed{\square}\) : Trend diagram with grid lines in the X-direction in the selected color \\
\hline "Font" & Font for the axis label \\
\hline
\end{tabular}

\section*{Tab "Y axis"}

Table 663: "Display mode"
\begin{tabular}{|c|c|}
\hline - "Auto": & O: The visualization scales automatically. \\
\hline - "Fixed" & O: Fixed range from "Minimum" to "Maximum" \\
\hline "Minimum" & \begin{tabular}{l}
Literal, variable (integer data type), or constant variable (integer data type). It contains the initial value of the segment. Requirement: The "Display Mode" is "Fixed". \\
Examples: 20,PLC_PRG.iLimit_Min, GVL.c_iLimit_Min \\
Note: The variable has to have an initial value. This is important for the offline display and the scaling subdivision. Example: iLimit_Min : INT := 20
\end{tabular} \\
\hline "Maximum" & \begin{tabular}{l}
Literal, variable (integer data type), or constant variable (integer data type). It contains the end value of the segment. Requirement: The "Display Mode" is "Fixed". \\
Examples: 80, PLC_PRG.iLimit_Max, GVL.c_iLimit_Max \\
Note: The variable has to have an initial value. This is important for the offline display and the scaling subdivision. Example: iLimit_Max : INT := 80
\end{tabular} \\
\hline "Grid" & V: Trend diagram with grid lines in the Y-direction in the selected color \\
\hline "Description" & V: Text for labeling the Y -axis (for example, \(\mathrm{DC} / \mathrm{mA}\) ) \\
\hline
\end{tabular}

Table 664: "Tick marks"
\begin{tabular}{|l|l|}
\hline "Fixed spacing" & : Axis scale with tick marks for "Distance" and "Subdivisions" \\
\hline "Distance" & Distance between the tick marks (example: 2) \\
\hline "Subdivisions" & Number of subdivisions between tick marks (example: 4) \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Font" & Font for the axis label \\
\hline
\end{tabular}

Table 665: "Background"
\begin{tabular}{|l|l|}
\hline "From visualization style" & Background color as defined in the visualization style \\
\hline "Draw background" & Background color which is selected in the lower input field \\
\hline "No background" & Trend diagram with transparent background \\
\hline \begin{tabular}{l} 
Background color of the trend \\
diagram
\end{tabular} & Requirement: "Draw background" is activated. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Reset" & Resets the settings to the default settings \\
\hline "Use as default" & Saves the settings as default \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Add Y-axis" & \begin{tabular}{l} 
Extends the trend diagram by one Y-axis \\
Result: The "Trend Recording" editor contains an extended selection of Y-axes in \\
the "Additional axes" option of the "Variable Settings".
\end{tabular} \\
\hline "Delete Y-axis" & Deletes the Y-axis of the visible tab. \\
\hline
\end{tabular}

\section*{See also}
- \(\boldsymbol{y y}^{2}\) Chapter 6.4.5.21.2.16 "Command 'Configure Display Settings of Trend'" on page 3359
- Editor 'Trend Recording'

\section*{Command 'Configure Trace'}

Symbol:
Function: This command opens the "Trace Configuration" dialog box.
Call: Context menu of the visualization element; "Trace" property of the visualization element.
Requirement: An element of type "Trace" is open in the editor.

Dialog box
'Trace Configuration'

The tree view shows the trace configuration and allows navigation.
The top entry contains the trace name. When this entry is selected, the "Record Settings" group appears in the adjacent view.

An entry is located here for each variable that data was recorded continuously. When a variable is selected, the "Variable Settings" group appears in the adjacent view.
\begin{tabular}{|l|l|}
\hline "Add variable" & \begin{tabular}{l} 
Adds a new entry to the trace configuration. \\
Result: A blank configuration appears next to the new variable under "Variable \\
Settings". You configure the variable there.
\end{tabular} \\
\hline "Delete variable" & Removes the selected variable. \\
\hline
\end{tabular}

\section*{'Recording Set- \\ tings'}
\begin{tabular}{|l|l|}
\hline \multicolumn{2}{|l|}{ A trigger can be configured in the trace only. } \\
\hline "Task" & Task where data was recorded. \\
\hline "Record condition" & \begin{tabular}{l} 
Recording condition for which the application records data in runtime mode: \\
Variable (BOOL)
\end{tabular} \\
\hline "Comment" & Example: Acquiring data only when all conditions are true. \\
\hline "Resolution" & \begin{tabular}{l} 
Measure for the time stamp that is recorded per data set. \\
- "ms": Time stamp (in milliseconds). \\
\(\bullet \quad " \mu s ": ~ T i m e ~ s t a m p ~(i n ~ m i c r o s e c o n d s) ~ f o r ~ a ~ t a s k ~ c y c l e ~ t i m e ~ o f ~\)
\end{tabular} 1 ms or less
\end{tabular}\(|\)\begin{tabular}{ll} 
"Automatic restart" & \begin{tabular}{l} 
Vecording starts automatically as soon as the trace has been started one \\
time and then the controller was restarted. The trace configuration and the con- \\
tents of the trace buffer are saved persistently to a file on the target system. \\
Format: . trace. csv
\end{tabular} \\
\hline "Display" & The "Edit Appearance" dialog box opens. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Advanced" & The "Advanced Trace Settings" dialog box opens. \\
\hline "Copy from Trace" & \begin{tabular}{l} 
The "Copy Settings from Trace Instance" dialog box opens. If you have already \\
created an existing trace configuration from a trace object, then you can copy the \\
configuration data to the visualization element. To do this, select the respective \\
object.
\end{tabular} \\
\hline
\end{tabular}
'Variable Settings'
\begin{tabular}{|c|c|}
\hline "Variable" & \begin{tabular}{l}
Variable for recorded value. \\
- Variable (valid data type) \\
- property \\
- Reference \\
- Contents of the pointer \\
- Array element (base type with valid data type) \\
- Enumeration (base type with valid data type) \\
Valid data types are all standard types, except STRING, WSTRING, and ARRAY.
\end{tabular} \\
\hline "Parameters" & \begin{tabular}{l}
Parameter whose value is acquired. \\
...: Input assistance lists ale valid parameters of the PLC.
\end{tabular} \\
\hline \(\checkmark\) & Enables toggling between "Variable" and "Parameter" \\
\hline "Attached axis" & \begin{tabular}{l}
Y -axis of the trace diagram for the "Variable". \\
Selection of the standard \(y\)-axis and the additional configured \(y\)-axes \\
Note: The additional configured y -axes are configured in the "Edit Display Settings" dialog box.
\end{tabular} \\
\hline "Display variable name" & \begin{tabular}{l}
The trace graphs are displayed in tooltip with their variable names. \\
If a text is also specified in "Description", then the text is displayed first with the variable names in parentheses. \\
Example: Sensor A (PLC_PRG.iSensor_A) \\
If "Description" does not contain any text, then the "Display Variable Name" property is activated automatically. Then only the name is displayed (example: PLC_PRG.iSensor_A).
\(\qquad\) The trace graphs are displayed in tooltip without their variable names. Only the text in "Description" is displayed.
\end{tabular} \\
\hline "Description" & \begin{tabular}{l}
Text for the tooltip. It is displayed when a visualization user moves the cursor in the trace diagram. \\
Example: Sensor A \\
The text is also entered into the "GlobalTextList" object and can be translated there.
\end{tabular} \\
\hline "Color" & Color of the graph in the diagram. \\
\hline "Line type" & \begin{tabular}{l}
Representation of the graph as a line chart \\
- "Line": Values are linked to form a line. \\
- "Step": Values are linked in the form of steps. \\
- "None": Values are not linked.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Line width" & \begin{tabular}{l} 
In pixels \\
Example: 1
\end{tabular} \\
\hline "Line style" & The display of the line is solid, dash, dot, dash-dot, or dash-dot-dot. \\
\hline "Dot type" & \begin{tabular}{l} 
Representation of the graph as a scatter chart. This configuration entry with the \\
"Line type" determines the appearance of the graph. \\
- "Dot": Each value as a dot. \\
- "Cross": Each value as a cross. \\
- "None" \\
Note: For "Dot" or "Cross", a paint buffer overflow can result from many recorded \\
variables.
\end{tabular} \\
\hline "Warning at minimum" & \begin{tabular}{l} 
V: When below the lower limit, the visualization shows the trace graphs in the \\
alert color.
\end{tabular} \\
\hline "Critical lower limit" & \begin{tabular}{l} 
Minimum Value \\
Example: 10.
\end{tabular} \\
\hline "Color" & Warning color on falling below the limit \\
\hline "Warning at maximum" & \begin{tabular}{l} 
V: When above the upper limit, the visualization shows the trace graphs in the \\
alert color.
\end{tabular} \\
\hline "Critical upper limit" & \begin{tabular}{l} 
Maximum value \\
Example: 90
\end{tabular} \\
\hline "Color" & Warning color on exceeding the limit \\
\hline "Dynamic appearance options" & \begin{tabular}{l} 
Variable (Bool) or as bit access. This controls the visibility of the variables in the \\
trace diagram. \\
- TRUE: Visible \\
- FALSE: Invisible
\end{tabular} \\
\hline "Variable for visibility" & \\
\hline
\end{tabular}

See also
- \& Chapter 6.4.5.21.2.13 "Command 'Configure Trace"' on page 3355

\section*{Command 'Export Trace Configuration'}

Function: This command opens the "Export Trace Configuration" dialog box.
Call: context menu (right-click) the upper node in the tree view of the trace configuration.
Requirement: The dialog box "Trace Configuration" is active and the name of the trace configuration is selected in the tree view (example: Visu_Trace1).

Dialog box 'Ex- This dialog is used for saving the trace configuration to a text file that can be read by the port Trace Con- runtime system. figuration'
\begin{tabular}{|l|l|}
\hline "File name" & Name of text file to be created. \\
\hline "File type" & \begin{tabular}{l} 
"Trace file \({ }^{*}\).trace)": Format that the runtime system component CmpTraceMgr \\
expects for reading.
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{*} \boldsymbol{y}\) Chapter 6.4.5.21.2.13 "Command 'Configure Trace"" on page 3355

\section*{Command 'Insert Elements for Controlling Trace'}

Symbol: 松
Function: The command opens the "Trace Wizard" dialog. In this dialog, you select predefined visualization elements for controlling the trace recording. These elements are then inserted as configured into the visualization editor.
Call: Menu bar: "Visualization"; context menu of the trace element.
Requirement: The view is active and a trace element is selected.

Dialog 'Trace wizard'
\begin{tabular}{|l|l|}
\hline "Control variable" & \begin{tabular}{l} 
Corresponds to the "Control variables" property that is available in the element \\
properties of the trace element. \\
V: The control element for this trace control variable is created in the visualiza- \\
tion editor.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Project variables that are assigned to the control element below the "Input \\
configuration" property. In addition, the project variables are declared as local \\
variables in the visualization when needed (in the interface editor). \\
This list corresponds to the assignments that are defined in the element proper- \\
ties of the trace element. If nothing is configured in the properties of the trace \\
element (no project variables assigned as control variables), then a pre-alloca- \\
tion is offered with default variable names.
\end{tabular} \\
\hline "Type of element to insert" & \begin{tabular}{l} 
For a Boolean variable, this element can be inserted as a button or rectangle. \\
For a string variable, a rectangle or a text field is provided.
\end{tabular} \\
\hline "OK" & \begin{tabular}{l} 
At the closing of the dialog, the selected control elements are inserted into \\
the visualization editor and (when needed) its control variables are created as \\
local variables of the visualization. They are declared in the interface editor and \\
they are used by the control element (property "Input configuration \(\rightarrow\) Toggle \\
G Variable") and by the trace element ("Control variables" property). The control \\
element writes to the variable and the trace element reads the variable.
\end{tabular} \\
\hline
\end{tabular}
```

Example
Standard con-
trol variables: VAR
bResetTrigger : BOOL;
bStart : BOOL;
bStop : BOOL;
bStore : BOOL;
sStoreFilename : STRING;
bRestore : BOOL;
sRestoreFilename : STRING;
END_VAR

```

See also
- \({ }^{\text {• }}\) Chapter 6.4.5.12.2 "Getting started with trace" on page 2926

\section*{Command 'Configure Display Settings of Trend'}

Symbol: 는
Function: When you execute this command in "Visualization" or in the context menu, the "Edit Display Settings" dialog opens.

\section*{Call:}
- Menu bar: "Visualization"
- Context menu of a "Trend" element in the visualization editor
- Property "Diagram"

Requirement: A trend is selected in the active visualization editor.

Tab "X Axis"
\begin{tabular}{|l|l|}
\hline "Grid" & \(\boxed{\square}\) : Trend diagram with grid lines in the X-direction in the selected color \\
\hline "Font" & Font for the axis label \\
\hline
\end{tabular}

\section*{Tab "Y axis"}

Table 666: "Display mode"
\begin{tabular}{|c|c|}
\hline - "Auto": & O: The visualization scales automatically. \\
\hline - "Fixed" & O: Fixed range from "Minimum" to "Maximum" \\
\hline "Minimum" & \begin{tabular}{l}
Literal, variable (integer data type), or constant variable (integer data type). It contains the initial value of the segment. Requirement: The "Display Mode" is "Fixed". \\
Examples: 20,PLC_PRG.iLimit_Min, GVL.c_iLimit_Min \\
Note: The variable has to have an initial value. This is important for the offline display and the scaling subdivision. Example: iLimit_Min : INT := 20
\end{tabular} \\
\hline "Maximum" & \begin{tabular}{l}
Literal, variable (integer data type), or constant variable (integer data type). It contains the end value of the segment. Requirement: The "Display Mode" is "Fixed". \\
Examples: 80,PLC_PRG.iLimit_Max, GVL.C_iLimit_Max \\
Note: The variable has to have an initial value. This is important for the offline display and the scaling subdivision. Example: iLimit_Max : INT := 80
\end{tabular} \\
\hline "Grid" & \(\square\) : Trend diagram with grid lines in the Y -direction in the selected color \\
\hline "Description" & V: Text for labeling the Y-axis (for example, DC/mA) \\
\hline
\end{tabular}

Table 667: "Tick marks"
\begin{tabular}{|l|l|}
\hline "Fixed spacing" & : Axis scale with tick marks for "Distance" and "Subdivisions" \\
\hline "Distance" & Distance between the tick marks (example: 2) \\
\hline "Subdivisions" & Number of subdivisions between tick marks (example: 4) \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Font" & Font for the axis label \\
\hline
\end{tabular}

Table 668: "Background"
\begin{tabular}{|l|l|}
\hline "From visualization style" & Background color as defined in the visualization style \\
\hline "Draw background" & Background color which is selected in the lower input field \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "No background" & Trend diagram with transparent background \\
\hline \begin{tabular}{l} 
Background color of the trend \\
diagram
\end{tabular} & Requirement: "Draw background" is activated. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Reset" & Resets the settings to the default settings \\
\hline "Use as default" & Saves the settings as default \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Add Y-axis" & \begin{tabular}{l} 
Extends the trend diagram by one Y-axis \\
Result: The "Trend Recording" editor contains an extended selection of Y-axes in \\
the "Additional axes" option of the "Variable Settings".
\end{tabular} \\
\hline "Delete Y-axis" & Deletes the Y-axis of the visible tab. \\
\hline
\end{tabular}

See also
- \({ }^{\text {• }}\), Chapter 6.4.5.21.2.12 "Command 'Configure Display Settings of Trend'" on page 3353
- Editor 'Trend Recording'

\section*{Command 'Edit Trend Recording'}

Symbol: 㓂
Function: This command opens the "Trend Recording" object.
Call:
- Menu bar: "Visualization"
- Context menu of a "Trend" element in the visualization editor
- Property "Trend recording"

Requirement: An element of type "Trend recording" is selected in the visualization editor.
See also
- \({ }^{\sharp}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872
- \(\Leftrightarrow\) Chapter 6.4.1.21.2.33 "Object 'Trend Recording'" on page 2545

\section*{Command 'Insert Elements for Controlling the Trend'}

Symbol: 图
Function: When you execute this command in "Visualization", the "Trend Wizard" dialog box opens.

Call: Menu bar: "Visualization"; context menu of a "Trend" element in the visualization editor.
Requirement: A trend is selected in the active visualization editor.

Dialog 'Trend \(\quad \begin{aligned} & \text { Each row of the table contains a control element that can assigned to the trend. The ele- } \\ & \text { ments are placed in the visualization next to the trend. The control elements are saved in the } \\ & \text { wizard' }\end{aligned}\)
"Assigned control elements" property and can be modified there.
\begin{tabular}{|l|l|}
\hline "Attached control element" & \begin{tabular}{l}
\(\boxed{\checkmark}\) : The associated element is available in the visualization and connected with \\
the trend via the property "Assigned Visu element". The element is inserted into \\
the visualization. \\
\(\square\)
\end{tabular} \\
\hline : Deactivating the option does not cause the element to be deleted from the \\
visualization.
\end{tabular}

See also
- 学 Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Command 'Visualization Element Repository'}

Symbol: 田
Function: This command opens the "Visualization Element Repository" dialog box opens for editing the storage location and visualization profile.
Call: Menu bar: "Tools".
Requirement: No project is open.


\section*{NOTICE!}
1. Only an empty directory can be selected as a new storage location for a repository.
2. The "System" repository cannot be modified. This is indicated by the entry in italics in the repository list.

Currently only a single version of an element can be installed.

Dialog box 'Vi-
sualization Ele-
ment Reposi-
tory'
Table 669: Editing the repository
\begin{tabular}{|l|l|}
\hline "Location" & \begin{tabular}{l} 
Storage location for the repository in the file system. The drop-down list contains \\
the configured repositories for visualization elements.
\end{tabular} \\
\hline "Edit locations" & \begin{tabular}{l} 
Opens the "Edit Repositories" dialog box for modifying the repository currently \\
selected in "Location" or for creating a new repository.
\end{tabular} \\
\hline
\end{tabular}

Table 670: "Profile or extension selection"
A profile is a collection of visualization elements in a specific version. These elements originate from one or more libraries. They are available in the toolbox of the visualization editor when the profile is used in the project. You can use an extension to add a specific selection of elements to an existing profile.
Creating and editing a profile is possible only if the CODESYS VisuElement Toolkit is installed. In this case, the buttons on the right side of the dialog box can be used.
\begin{tabular}{|l|l|}
\hline "Create or update profile" & \begin{tabular}{l} 
You can configure a new profile in the dialog or modify an existing one. Then, \\
the "New", "Copy", and "Delete" buttons are operable, as well as the "Installed \\
Elements" and "Available Elements" views.
\end{tabular} \\
\hline "Create or update extension" & \begin{tabular}{l} 
In the dialog, you can configure an extension for the selected profile. Another \\
drop-down list "Extension" appears with all currently available extensions. \\
To configure a new extension, use the "New" or "Copy" buttons (see below). \\
In both cases, the "Specify Visualization Extension" dialog box opens for you \\
to define a new extension. In this dialog box, the "Name", "Company", and \\
"Version" of the extension are displayed. Version syntax: Sequence of numbers \\
and points with a number at the end.
\end{tabular} \\
\hline "Profile" & \begin{tabular}{l} 
Currently selected profile. The drop-down list provides all profiles available in the \\
repository set above.
\end{tabular} \\
\hline "Extension" & \begin{tabular}{l} 
The extension that is currently selected for the specified profile. The drop-down \\
list provides all extensions available for the profile.
\end{tabular} \\
\hline "New" & \begin{tabular}{l} 
Pressing the button opens the "Specify Name of Visualization Profile" dialog or \\
the "Specify Visualization Extension" dialog. Specify a unique name for the new \\
profile, or for an extension also the company name and the version. CODESYS \\
automatically enters the previously used name, appended with "_0". \\
The "Installed elements" list is empty.
\end{tabular} \\
\hline "Copy" & \begin{tabular}{l} 
Pressing the button opens the "Specify Name of Visualization Profile" dialog or \\
the "Specify Visualization Extension" dialog (see above: "New"). \\
The elements of the selected profile are accepted and they appear in the
\end{tabular} \\
"Installed elements" view.
\end{tabular}

Table 671: "Installed Elements"
\begin{tabular}{|l|l|}
\hline "Name, Vendor, Library" & Elements that are assigned to the selected profile. \\
\hline "Uninstall" & \begin{tabular}{l} 
All elements currently selected in the list are uninstalled and removed from the \\
"Available Elements" list.
\end{tabular} \\
\hline "Update code" & \begin{tabular}{l} 
The list is refreshed with any changes in the implementation code of the library \\
POUs.
\end{tabular} \\
\hline "Update all" & \begin{tabular}{l} 
The list is refreshed with any changes in the implementation code and in the \\
interfaces (declaration part) of the library POUs.
\end{tabular} \\
\hline
\end{tabular}

Table 672: "Available Elements"
\begin{tabular}{|l|l|}
\hline "Name" & \begin{tabular}{l} 
Elements that are available in the system and can be installed into the current \\
profile or extension. The selection depends on the installed element libraries and \\
element packages. \\
"Library"
\end{tabular} \\
"Vendor" & \begin{tabular}{l} 
The tree structure displays the libraries with the contained elements below them. \\
Elements display in green are already installed for the specified profile or exten- \\
sion. "Profiles"s shows the elements installed for the profiles.
\end{tabular} \\
"Version" & "Repositories" \\
"Profiles" & \begin{tabular}{l} 
The elements selected in the list are added to the "Installed Elements" view. \\
Existing elements are overwritten.
\end{tabular} \\
\hline "Install element" & \begin{tabular}{l} 
The "Library Repository" dialog box opens where another library can be installed \\
in order to accept its elements in the "Available Elements" view.
\end{tabular} \\
\hline "Install library" &
\end{tabular}
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
"Note current library versions \\
only"
\end{tabular} & \begin{tabular}{l} 
When refreshing the list, only the most current version of the library is \\
searched, not all libraries.
\end{tabular} \\
\hline \begin{tabular}{l} 
"Overwrite profiles without \\
prompting"
\end{tabular} & \begin{tabular}{l} 
: For actions that change the profile, the usual prompt does not appear for \\
confirming the change.
\end{tabular} \\
\hline
\end{tabular}

Dialog box 'Edit Repositories'
\begin{tabular}{|l|l|}
\hline "Storage location, Name" & \begin{tabular}{l} 
For managing the visualization elements, one or more repositories can be used. \\
All currently defined storage locations are listed here with file path and name. \\
The order from top to bottom is also the search order for the visualization ele- \\
ments. \\
File path and name of the storage location selected previously in the "Repository \\
for Visualization Elements" dialog. \\
Note: A storage location "System" is always defined automatically, which cannot \\
be modified or deleted.
\end{tabular} \\
\hline "Add" & \begin{tabular}{l} 
Opens the "Storage Location for Repository" dialog for creating a new storage \\
location or for editing the current storage location. \\
Specify: "Storage location" (file path of an empty directory) and "Name". The \\
name is symbolic (example: "Elements category 1").
\end{tabular} \\
\hline "Remove" & Deletes the repository currently selected in the repository list. \\
\hline "Move Up, Move Down" & \begin{tabular}{l} 
Moves the entries within the list. Note: The repositories are searched from top to \\
bottom.
\end{tabular} \\
\hline
\end{tabular}

\section*{Command 'Visualization Style Repository'}

Symbol:
Function: This command opens the "Visualization Styles" dialog box. It makes it possible to edit visualization style repositories.
Call: Menu bar: "Tools".
See also
- \({ }^{\Perp}\) Chapter 6.4.5.19.3 "Managing visualization styles in repositories" on page 2985

Dialog Box 'Vi-

\section*{sualization}

Styles'
\begin{tabular}{|l|l|}
\hline "Storage location" & \begin{tabular}{l} 
Name of the currently selected repository Preset: "System" \\
v: Lists the repositories installed in the development system.
\end{tabular} \\
\hline "(...)" & \begin{tabular}{l} 
Storage location of the repository \\
Example: (C: \(\backslash\) ProgramData \(\backslash\) CODESYS \(\backslash V i s u a l i z a t i o n ~ S t y l e s) ~\)
\end{tabular} \\
\hline "Edit locations" & The "Edit Repository Locations" dialog box opens. \\
\hline
\end{tabular}

Table 673: "Installed Visualization Styles"
\begin{tabular}{|c|c|}
\hline "Company" & \begin{tabular}{l}
When a company name is specified here, the tree view is filtered and only the styles of the selected company are listed. \\
Preset: "(All companies)". It is not filtered. \\
: Lists all companies that are specified in the styles.
\end{tabular} \\
\hline Windows with styles & Tree view of all versions of the installed visualization styles in the selected repository \\
\hline "Display localized names" & The style name is localized and displayed in the language that is set in CODESYS.
\(\square\) : The style is display as the source name. \\
\hline "Install" & The "Select Visualization Style(s)" dialog box opens. \\
\hline "Uninstall" & The selected style version is removed from the repository. \\
\hline "Preview" & The windows closes. A preview is displayed of the selected style in the selected version. Specific elements are displayed in the style. \\
\hline
\end{tabular}

\section*{Dialog box 'Edit \\ Repository \\ Locations'}

Table 674: "Repositories (elements are searched in that order)"
\begin{tabular}{|l|l|}
\hline "Location" & \begin{tabular}{l} 
Storage location of the configured repository on the development system \\
Example: C : \ProgramData \CODESYS \(\backslash\) Visualization Styles
\end{tabular} \\
\hline "Name" & Preset: System \\
\hline "Add" & \begin{tabular}{l} 
The "Repository Locations" dialog box opens. It makes it possible to manage \\
other repositories.
\end{tabular} \\
\hline "Edit" & \\
\hline "Remove" & \begin{tabular}{l} 
The order in the list of repositories is adapted. It defines the processing order \\
when searching for elements.
\end{tabular} \\
\hline "Move Up" & "Move Down"
\end{tabular}

\section*{Command 'Add Visual Element'}

Function: The command opens a menu containing all available visualization elements as menu items.
Requirement: You have configured the command in the dialog box "Customize" in a way that you have a call in a (any) menu.

When you select an element in the menu, the element is added in the visualization editor in the upper left corner.
See also
- \(\Leftrightarrow\) Chapter 6.4.5.5.2 "Select Element" on page 2873
- \(\#\) Chapter 6.4.1.21.3.9.16 "Command 'Customize"' on page 2667

\section*{Command 'Select None'}

Function: The command cancels at once any selection in the current visualization editor.
Requirement: You have configured the command in the dialog box "Customize" in a way that you have a call in a (any) menu.
See also
- \({ }^{4}\) Chapter 6.4.5.5.2 "Select Element" on page 2873
- ② Chapter 6.4.5.21.2.7 "Command 'Group'" on page 3347
- \(\Leftrightarrow\) Chapter 6.4.1.21.3.9.16 "Command 'Customize"" on page 2667

\section*{Command 'Add Elements for Alarm Acknowledgement'}

Symbol: \(\Delta\)
Function: This command adds buttons automatically to the visualization for acknowledging alarms. It opens an assistant for inserting controls below the table.
Call: Menu bar: "Visualization"; context menu of visualization element "Alarm table"
Requirement: An "Alarm table" visualization element is selected.

Dialog box
'Alarm Table
Wizard'
\begin{tabular}{|l|l|}
\hline "Type of element(s) to insert" & \begin{tabular}{l}
\(\bullet\) "Button" \\
- "Rectangle"
\end{tabular} \\
\hline "Action" & \begin{tabular}{l} 
Ø: A button or a rectangle with the selected function is added to the visualiza- \\
tion.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
If you have already specified a variable for an action, then this is displayed here \\
in the "Variable" column. If you have not defined a variable yet, then a local \\
visualization variable is created automatically.
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{\Perp}\), Chapter 6.4.5.21.5.22 "Visualization Element 'Alarm Table"' on page 3591

\subsection*{6.4.5.21.3 Dialog Boxes}
6.4.5.21.3.1 Dialog 'Access Rights'. ..... 3366
6.4.5.21.3.2 Dialog 'Add Visualization'. ..... 3367
6.4.5.21.3.3 Dialog 'Update Frame Parameters'. ..... 3367
6.4.5.21.3.4 Dialog 'Configure Categories and Items'. ..... 3368
6.4.5.21.3.5 Dialog 'Gradient Editor'. ..... 3369
6.4.5.21.3.6 Dialog 'Input Configuration' ..... 3370
6.4.5.21.3.7 Dialog 'Options' - 'Visualization Styles' ..... 3382
6.4.5.21.3.8 Dialog 'Options' - 'Visualization User Management'. ..... 3383
6.4.5.21.3.9 Dialog Box 'Options' - 'Visualization'. ..... 3384
6.4.5.21.3.10 Dialog 'Project Environment' - 'Visualization Profile'. ..... 3385
6.4.5.21.3.11 Dialog 'Project Environment' - 'Visualization Styles'. ..... 3386
6.4.5.21.3.12 Dialog 'Project Environment' - 'Visualization Symbols'. ..... 3386
6.4.5.21.3.13 Dialog 'Project Settings' - 'Visualization'. ..... 3387
6.4.5.21.3.14 Dialog 'Project Settings' - 'Visualization Profile' ..... 3388
6.4.5.21.3.15 Dialog 'Properties' of Visualization Objects ..... 3388
6.4.5.21.3.16 Dialog 'Selected Alarm Class' ..... 3389
6.4.5.21.3.17 Dialog 'Selected Alarm Group'. ..... 3390
6.4.5.21.3.18 Dialog 'Advanced Trace Settings'. ..... 3391
6.4.5.21.3.19 Dialog 'Display Settings'. ..... 3391

\section*{Dialog 'Access Rights'}

Function: This dialog defines the permissions of user groups for a visualization element.
Call: Click in the "Value" field of the "Access Rights" element property of a visualization element.
Requirement: A visualization element is selected in a visualization element and the "Properties" is open.
\begin{tabular}{|l|l|}
\hline "User Groups" & \begin{tabular}{l} 
Groups that were configured in the "Visualization Manager" (tab \\
"User Management \(\rightarrow\) Groups").
\end{tabular} \\
\hline "Operable" & \(\boxed{\square}:\) The visualization element is available with full functionality. \\
\hline "Only Visible" & \begin{tabular}{l} 
■: The visualization element is visible only and does not provide any function- \\
ality.
\end{tabular} \\
\hline "Invisible" & ■: The visualization element is not displayed. \\
\hline "Group hierarchy is used" & \begin{tabular}{l} 
Display whether the option "Use group hierarchy" is activated in the \\
"Visualization Manager" (tab "User Management \(\rightarrow\) Settings"). \\
A group of a higher hierarchy cannot have fewer permissions for an element \\
than an element of a lower hierarchy.
\end{tabular} \\
\hline
\end{tabular}

If no user is logged in, then the permissions apply for the visualization elements that are configured for the user group "None". If the permissions for a visualization element is restricted, then the group "None" should be granted the lowest permissions.

\section*{See also}
- 学 Chapter 6.4.5.21.4.5 "Tab 'Visualization manager' - 'User management"' on page 3403

\section*{Dialog 'Add Visualization'}

Function: The dialog is used to create a new object of type "Visualization".
Call: Menu bar: "Project \(\rightarrow\) Add Object \(\rightarrow\) Visualization"; context menu of an application
Requirement: An application is selected in the device tree.
\begin{tabular}{|l|l|}
\hline "Name" & \begin{tabular}{l} 
Name of the visualization \\
Example: Visu_A
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline The following settings are displayed only when you add a "Visualization" object to the project for the first time. \\
\hline "Symbol library" & List of all installed symbol libraries \\
\hline "Assigned" & \begin{tabular}{l} 
: Symbol library is selected \\
\\
\\
Hint: CODESYS manages this setting in the project settings.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Add" & \begin{tabular}{l} 
CODESYS creates a new visualization, assigns the selected symbol libraries to \\
the project, and lists them in the "Visualization Toolbox" view.
\end{tabular} \\
\hline
\end{tabular}

See also
- \& Chapter 6.4.5.5.2 "Select Element" on page 2873
- Dialog 'Project Settings' - 'Visualization'
- Command 'Add Object'

\section*{Dialog 'Update Frame Parameters'}

Function: The dialog requests you, after changing an interface in the visualization references concerned, to re-assign the variables for the parameter transfer.
Call: The dialog appears automatically.
Requirement: You have changed the interface of a visualization, for example by adding an additional variable. After that, you have clicked either "File \(\boldsymbol{\rightarrow}\) Save Project" or "Build \(\rightarrow\) Generate Code", or opened a visualization.
\begin{tabular}{|l|l|}
\hline "Parameter" & Hierarchical structure of the interface parameters as a tree view \\
\hline (T) & \begin{tabular}{l} 
Top node of the visualization hierarchy with the name of the visualization. This \\
contains an element of type "Frame" or "Tabs".
\end{tabular} \\
\hline Name of the element ("Frame" or "Tabs" type) \\
\hline " & Name of the referenced visualization \\
\hline <name> & \begin{tabular}{l} 
Interface of the referenced visualization with the new parameters \\
You can edit the parameter transfer here.
\end{tabular} \\
\hline <name> & \begin{tabular}{l} 
Interface of the referenced visualization with the previously valid parameters. \\
You cannot edit the parameter transfer, but you can use it as a template.
\end{tabular} \\
\hline & Variable for the parameter transfer (VAR_INPUT scope) \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Type" & \begin{tabular}{l} 
Data type of the variable \\
Example: InTt
\end{tabular} \\
\hline & \begin{tabular}{l} 
Variable that is transferred as a parameter and with whose value the visualiza- \\
tion is initialized during instantiation. \\
Example: PLC_PRG. iVisNr \\
If the variable lies under the current interface, which is marked in the tree view \\
with "(Current)", then you can edit the parameter transfer. \\
- Click in the field to open the input field.
\end{tabular} \\
& \begin{tabular}{l} 
- Double-click in the field to open the Input Assistant. \\
- Accept the settings by copying assignments in the "Value" column and \\
pasting them into another cell. Use the "Copy" and "Paste"links to do this.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Copy" & \begin{tabular}{l} 
Link for copying an assignment from the "Value" column. \\
Requirement: An assignment is selected.
\end{tabular} \\
\hline "Paste" & \begin{tabular}{l} 
Link for inserting an assignment \\
Requirement: You have copied an assignment.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "OK" & \begin{tabular}{l} 
Click the button to close the dialog and confirm the changes made under \\
"(Recent)". \\
Result: The assignment is entered in the "References" property and on the \\
"Interface Editor" tab.
\end{tabular} \\
\hline
\end{tabular}

See also
- \& Chapter 6.4.5.21.5.6 "Visualization Element 'Frame"' on page 3478
- © Chapter 6.4.5.21.2.1 "Command 'Interface Editor'" on page 3340

\section*{Dialog 'Configure Categories and Items'}

Function: The dialog is used to manage the categories in a tree view. The assigned elements are listed below a category. You can create custom categories and edit the assignment to the visualization elements. The name of the category is displayed in the "Visualization Toolbox" view as a label of the button to open the element selection.

Call: Click the 国 symbol in the "Visualization Toolbox" view.
See also
- © Chapter 6.4.5.5.2 "Select Element" on page 2873
- \& Chapter 6.4.5.21.4.1.2 "View 'Visualization Toolbox"' on page 3394

\section*{Tree view}
\begin{tabular}{|c|c|}
\hline "Category" & \begin{tabular}{l}
Tree view \\
- 战 "<name>": Default category- 䣽"<name>": Custom category Example: "Favorite"
\end{tabular} \\
\hline + & \begin{tabular}{l}
Lists the assigned visualization elements. To remove a selected visualization element, click the [Del] key. \\
Hint: The assignment is created in the "Visualization Toolbox" view with the help of the context menu of a selected element.
\end{tabular} \\
\hline "Active" & V: A button for the category is visible in the "Visualization Toolbox" view. \\
\hline
\end{tabular}

\section*{Toolbar}
\begin{tabular}{|l|l|}
\hline+ & The "Add Category" dialog opens. \\
\hline- or [Del] & \begin{tabular}{l} 
The category selected in the tree view is removed. After you click "OK" to close \\
the dialog, the button is also removed from the "Visualization Toolbox" view.
\end{tabular} \\
\hline
\end{tabular}

Dialog 'Add Cat- Call: Click the + symbol in the "Configure Categories and Items" dialog. egory'
\begin{tabular}{|l|l|}
\hline "Name" & \begin{tabular}{l} 
Name of the category \\
Example: tagA
\end{tabular} \\
\hline "Description" & Example: Tagged with A \\
\hline
\end{tabular}

\section*{Dialog 'Gradient Editor'}

Function: The dialog is for setting the color gradient of visualization elements. If you define two colors, the color graduates between them. If you only select 1 color, the color graduates within this color through its brightness. The detailed settings are for a special specification of the initial position and the angle of the color gradient.
Call: Click in the value field of the property "Gradient settings"
Requirement: You have selected a visualization element in the editor that has the property "Gradient settings".
\begin{tabular}{|l|l|}
\hline "Gradient type" & \begin{tabular}{l}
\(\bullet\) \\
\(\bullet\) "Linear" \\
"Radial" \\
\(\bullet \quad\) "Axial": The color gradient runs along an axis, with the colors extending \\
perpendicular to the axis on both sides.
\end{tabular} \\
\hline "Color 1" & First color of the gradient. \\
\hline "Color 2" & Second color of the gradient. \\
\hline "Transparency" & \begin{tabular}{l} 
Transparency of the associated color. Permissible values: Integers in the range \\
of values from 255 to 0. 255: The color is opaque. 0: The color is fully trans- \\
parent.
\end{tabular} \\
\hline "Standard linear" & \begin{tabular}{l} 
Requirement: "Linear" color gradient. \\
Standard direction of the linear color gradient.
\end{tabular} \\
\hline "Standard radial" & \begin{tabular}{l} 
Requirement: "Radial" color gradient. \\
Standard setting.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Standard axial" & \begin{tabular}{l} 
Requirement: "Axial" color gradient. \\
Direction of the color gradient
\end{tabular} \\
\hline "Angle (degrees)": & Requirement: "Linear" or "Axial" color gradient. \\
\hline "Center X (\%):" & \begin{tabular}{l} 
Requirement: "Radial" color gradient. \\
X-position of the center point (0 - 100\%)
\end{tabular} \\
\hline "Center Y (\%):" & \begin{tabular}{l} 
Requirement: "Radial" color gradient. \\
Y-position of the center point (0 - 100\%)
\end{tabular} \\
\hline "Use one color" & Color gradient between "Color 1" and the same color with a different brightness. \\
\hline "Brightness" & \begin{tabular}{l} 
Requirement: The option "Use one color" is selected. \\
Setting from 0 (black) to 100 (white)
\end{tabular} \\
\hline "Use two colors" & Color gradient between the two selected colors "Color 1" and "Color 2". \\
\hline
\end{tabular}

See also
- ② Chapter 6.4.5.5.4 "Assigning a color" on page 2876
- \(\boldsymbol{y y}^{4}\) Chapter 6.4.5.10.4 "Animating a color display" on page 2914

\section*{Dialog 'Input Configuration'}

Symbol:
Function: The dialog is used to assign input actions to specific input events. It also includes specific settings for the selected input action.

Call: In the "Input configuration" property, click "Configure".
Requirement: An element is selected in the editor.

Input action
'User Manage-
ment'
\begin{tabular}{|l|l|}
\hline "Dialogs and actions" & \begin{tabular}{l} 
Configures which one of the possible user management dialogs or which action \\
follows the input event \\
Note: The dialog used at runtime is configured in the "Dialog Settings" tab of the \\
Visualization Manager. \\
See also \\
- "Login dialog" \\
- "Change password dialog" \\
- "Change configuration dialog" \\
Default: Dialogs from the VisuUserManagement library
\end{tabular} \\
\hline "Login" & \begin{tabular}{l} 
The login prompt opens. \\
Default: VisuUserManagement. Vum_Login in "Login dialog"
\end{tabular} \\
\hline "Logout" & The current user is logged out. \\
\hline "Change User Password" & \begin{tabular}{l} 
The dialog for changing the password opens. \\
Default: VisuUserManagement. VUM_Change Password in "Change password \\
dialog"
\end{tabular} \\
\hline "Open User Configuration" & \begin{tabular}{l} 
The dialog opens for changing the configuration. \\
Default: VisuUserManagement. VUM_UserManagement in "Change \\
configuration dialog"
\end{tabular} \\
\hline
\end{tabular}

See also
- « Chapter 6.4.5.7 "Setting Up User Management" on page 2899
- «"Tab 'Visualization manager' - 'Settings"' on page 3398

Input action
'Close Dialog'
\begin{tabular}{|c|c|}
\hline "Dialog" & \begin{tabular}{l}
The visualization of type "Dialog" that will be closed. \\
\(\nabla\) : List box with all "Dialog" type visualizations available in the project. \\
Example: \\
Default dialogs of the VisuDialogs library, which is usually integrated in the project. \\
- FileOpenSave \\
- Keypad \\
- Login \\
- Numpad \\
- NumpadExtended \\
- TextinputWithLimits \\
Note: The setting in the object property ("Visualization" tab) of a visualization determines whether or not a visualization can be used as a dialog.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Result" & \begin{tabular}{l} 
Return value for closing the dialog. \\
Note: If there are more input actions after closing, then they configured in the \\
"Input configuration \(\rightarrow\) OnDialogClosed" property of the element.
\end{tabular} \\
\hline "None" & \(\boxed{\square}:\) No return value
\end{tabular}

See also
- " Chapter 6.4.5.21.3.15 "Dialog 'Properties' of Visualization Objects" on page 3388

Input action
'Open Dialog'
\begin{tabular}{|l|l|}
\hline "Dialog" & Visualization (type "Dialog"). The dialog opens. \\
& \(\square:\) List box with all dialogs available in the project. \\
& Note: The "VisuDialogs" library provides visualizations (type "Dialog"). \\
& - VisuDialogs. FileOpenSave \\
& - VisuDialogs. Login \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline Transfer parameters of the dialog \\
\hline "Parameter" & \begin{tabular}{l} 
Interface parameter as declared in the interface editor of the visualization \\
Example: filelistProvider
\end{tabular} \\
\hline "Type" & \begin{tabular}{l} 
Data type of the parameter as declared in the interface editor of the visualization. \\
Example: VISU_FBFILELISTPRoVIDER
\end{tabular} \\
\hline "Value" & \begin{tabular}{l} 
Variable (data type corresponds to the data type of the parameter). The value of \\
the variable is read when the dialog opens and passed to the parameter. \\
Example: PLC_PRG.fileListProvider // Instance of function \\
block VisuDialogs.Visu_FbFileListProvider \\
l.a: The input assistance offers all variables available in the entire project.
\end{tabular} \\
\hline
\end{tabular}

Here the return value of the dialog is activated for which the Var_OUTPUT variable and VAR_IN_OUT variable are written. The dialog closes afterwards.
\begin{tabular}{|c|c|}
\hline "Update" "and" w "parameter in case of result" & Note: The parameters are updated before the dialog is closed. Until then, the values are stored temporarily. They are stored as a copy, not as a reference. \\
\hline "None" & V: No return value \\
\hline "OK" & 凹: Defines the return value for which the transfer parameter is written \\
\hline "Cancel" & \\
\hline "Abort" & \\
\hline "Retry" & \\
\hline "Ignore" & \\
\hline "Yes" & \\
\hline "No" & \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Open dialog modal" & \begin{tabular}{l} 
Valy \\
blocked to user input.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Position to open" \\
\hline "Centered" & The dialog opens in the center of the visualization. \\
\hline "Position" & The dialog opens at the position defined by " \(X\) " and " \(Y\) ". \\
\hline " \(X\) " & Position (in pixels) or variable (integer data type) \\
\hline " \(Y\) " & Position (in pixels) or variable (integer data type) \\
\hline
\end{tabular}

See also
- Chapter 6.4.5.17.4 "Calling a dialog in a visualization" on page 2957

Input action
'Change Language'
\begin{tabular}{|l|l|}
\hline "Language" & \begin{tabular}{l} 
Language to be switched \\
Example: en \\
\\
\\
\hline
\end{tabular} : The input assistance offers all languages available in the project. \\
\hline
\end{tabular}

Input action
'Change Shown
Visualization'
Table 675: "Zoom to visualization"
\begin{tabular}{|l|l|}
\hline Visualization that is shown at the user input \\
\hline "Assign" & \begin{tabular}{l} 
Visualization that is selected from all available visualizations in the project or \\
libraries. \\
Example: visMa in
\end{tabular} \\
\hline "Assign expression" & \begin{tabular}{l} 
Variable (STRING) that contains the name of the visualization \\
Example: PLC_PRG.strVisu for the following application code: strVisu: \\
STRING : = 'visMain' ;
\end{tabular} \\
\hline \begin{tabular}{l} 
The order in which visualizations are displayed by user inputs is saved internally. The following options use this \\
information.
\end{tabular} & \begin{tabular}{l|l|}
\hline "Previous shown visualization" & \begin{tabular}{l} 
Ø: Visualization that has already been shown before the current one \\
Requirement: A visualization switch has already occurred.
\end{tabular} \\
\hline "Next shown visualization" & \begin{tabular}{l} 
■: Visualization that is next in the call order after the current one. \\
Requirement: A visualization switch has already occurred which was called by \\
"Previous shown visualization".
\end{tabular} \\
\hline
\end{tabular}
\end{tabular}

Input action 'Ex- Commands are listed here with transfer parameters that the visualization processes when an ecute Com- input event occurs. mand'
\begin{tabular}{|l|l|}
\hline "Configure commands" & - \\
& - "Execute program on the plc" \\
& - "Execute program on client" \\
& - "Print" \\
& - "Navigate to URL (WebVisu)" \\
& - "Create Recipe" \\
& - "Read Recipe" \\
& - "Write Recipe" \\
& - "Write Recipe in File" \\
& - "Load Recipe from File" \\
& - "Delete Recipe" \\
& Click + to add the selected command to the lower command list. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline+ & The command in "Configure commands" is added to the list. \\
\hline- & \begin{tabular}{l} 
The command is removed. \\
Requirement: \(A\) command is selected.
\end{tabular} \\
\hline The order in the list defines the order of execution. \\
\hline \(\boldsymbol{-}\) & The selected command is moved down one position in the list. \\
\hline \(\boldsymbol{-}\) & The selected command is moved up one position in the list. \\
\hline
\end{tabular}

Table 676: Command "Execute program on the plc"
\(\left.\left.\begin{array}{|l|l|l|}\hline \text { "Command" } & \text { "1st parameter" } & \text { "2nd parameter" } \\ \hline \text { ExecutePlcProgram } & \text { 'C: \programs } \backslash \text { notepad.exe' } & \text { 'Notes_A.txt' } \\ \hline \begin{array}{l}\text { EXE file that is executed on } \\ \text { the controller }\end{array} & \begin{array}{l}\text { Program name with directory as } \\ \text { STRING in single straight quotation } \\ \text { The program is executed } \\ \text { on the PLC and therefore } \\ \text { it must not be interactive or } \\ \text { have any user interfaces. } \\ \text { It is possible, for example, } \\ \text { for a program to copy a file. }\end{array} & \end{array} \begin{array}{l}\text { Arguments of the program } \\ \text { as STRING in single straight } \\ \text { quotation marks }\end{array}\right] \begin{array}{l}\text { Example: Name of the file } \\ \text { that the program opens }\end{array}\right\}\)

Table 677: Command "Execute program on client"
\begin{tabular}{|c|c|c|}
\hline "Command" & "1st parameter" & "2nd parameter" \\
\hline ExecuteClientProgram & 'C:\programs \notepad.exe' & 'Notes_A.txt' \\
\hline \begin{tabular}{l}
EXE file that is executed on the display variant. Exception: WebVisu. \\
The program is executed within the context of the display variant. After this, the program may be interactive and have a user interface.
\end{tabular} & Program name with directory as STRING in single straight quotation marks & \begin{tabular}{l}
Arguments of the program as STRING in single straight quotation marks \\
Example: Name of the file that the program opens
\end{tabular} \\
\hline
\end{tabular}

\section*{NOTICE!}

If the visualization is displayed as a CODESYS WebVisu, then no program (EXE file) can be started.

Table 678: Command "Navigate to URL (WebVisu)"
\begin{tabular}{|c|c|c|}
\hline "Command" & "1st parameter" & "2nd parameter" \\
\hline NavigateURL & 'http://en.wikipedia.org' PLC PRG.stURL & 'replace' \\
\hline \begin{tabular}{l}
The visualization navigates to the web page of the URL. \\
Requirement: The visualization is executed as a CODESYS WebVisu.
\end{tabular} & \begin{tabular}{l}
URL \\
- As a literal in single straight quotation marks \\
- As a variable (STRING)
\end{tabular} & \begin{tabular}{l}
If a parameter is not specified, then the web page is displayed in a new window or a new tab. \\
If 'replace' is specified, then the CODESYS WebVisu is replaced by the web page.
\end{tabular} \\
\hline
\end{tabular}

Table 679: Command "Read Recipe"
\begin{tabular}{|l|l|l|}
\hline "Command" & "1st parameter" & "2nd parameter" \\
\hline ReadRecipe & \begin{tabular}{l} 
'RecipeDefinitionForModule \\
s'
\end{tabular} & 'RecipeModuleA' \\
\hline
\end{tabular}
\begin{tabular}{|l|l|l|}
\hline & \begin{tabular}{l} 
Name of the recipe definition \\
\(\bullet\) \\
\(\bullet\) \\
\(\bullet\)
\end{tabular} As a literal & Name of the recipe \\
\(\bullet \quad\) As a literal \\
\hline •
\end{tabular}

At visualization runtime, the controller reads the actual values from the variables of the recipe definition and writes them to the specified recipe. The values are saved implicitly (to a file on the controller) and shown in the recipe definition in the Recipe Manager of CODESYS. In other words, the recipe that is managed in CODESYS is updated with values from the controller.

Table 680: Command "Write Recipe"
\begin{tabular}{|l|l|l|}
\hline "Command" & "1st parameter" & "2nd parameter" \\
\hline WriteRecipe & PLC_PRG.stRecipeDef & PLC_PRG.stRecipe \\
\hline & \begin{tabular}{l} 
Name of the affected recipe defini- \\
tion
\end{tabular} & \begin{tabular}{l} 
Name of the recipe (from \\
the recipe definition)
\end{tabular} \\
& \(\bullet\) As a literal & • As a literal \\
& \(\bullet\) As a variable (STRING) & \(\bullet\) As a variable (STRING) \\
\hline
\end{tabular}

At visualization runtime, the values of the recipe are written to the variables on the controller as they are in the Recipe Manager.

Table 681: Command "Save Recipe in File"
\begin{tabular}{|l|l|l|}
\hline "Command" & "1st parameter" & "2nd parameter" \\
\hline SaveRecipeAs & PLC_PRG.stRecipeDef & PLC_PRG.stRecipe \\
\hline & \begin{tabular}{l} 
Name of the affected recipe defini- \\
tion \\
\(\bullet \quad\) As a literal \\
\(\bullet \quad\) As a variable (STRING)
\end{tabular} & \begin{tabular}{l} 
Name of the affected recipe \\
that is updated and saved \\
to a file
\end{tabular} \\
\(\bullet \quad\) As a literal \\
\(\bullet \quad\) As a variable (STRING) \\
Optional parameter: If you \\
do not specify a transfer \\
parameter here, then the \\
values from the recipe vari- \\
ables are saved only the file \\
that is specified later. The \\
implicit recipe files are not \\
updated.
\end{tabular}

At visualization runtime, the "Save Recipe as" dialog opens and prompts the user for a file name and a storage location on the controller. The file name must not be <recipe>.<recipe definition>. The file extension is .txtrecipe.

The user can then save the file that includes the actual values from the recipe variables. If a transfer parameter is not specified in the 2nd parameter, then the file is saved without changing an implicit recipe file. If a transfer parameter is given in the 2nd parameter, then the implicit recipe file is also updated.

Note: If the "Save recipe changes to recipe files automatically" option is selected in the "Recipe Manager - General" tab, then the recipe definition in CODESYS and the implicit recipe files are kept the same automatically.
Note: Implicit (automatically generated) recipe files exist on the controller with names in the following syntax: <recipe>. <recipe definition>.txtrecipe. These are typically used in the application as a buffer when reading and writing recipe variables.

Table 682: Command "Load Recipe from File"
\begin{tabular}{|l|l|l|}
\hline "Command" & "1st parameter" & "2nd parameter" \\
\hline LoadRecipeFrom & PLC_PRG.stRecipeDef & PLC_PRG.stRecipe \\
\hline & \begin{tabular}{l} 
Name of the affected recipe defini- \\
tion \\
\(\bullet ~ A s ~ a ~ l i t e r a l ~\)
\end{tabular} & Name of the affected recipe \\
- As a variable (STRING) & • As a literal \\
\(\bullet\) • As a variable (STRING) \\
\hline \begin{tabular}{l} 
At visualization runtime, the "Load Recipe" dialog opens. It provides the visualization user \\
with a file list that is located in the file system of the controller and filters by the extension \\
txtrecipe. The selected file is downloaded. Then the recipes from the file are written to the \\
implicit files and read to the given recipe in the recipe definition of the Recipe Manager. \\
Requirement: The file was created with the SaveRecipeAs command.
\end{tabular} \\
\hline
\end{tabular}

Table 683: Command "Create Recipe"
\begin{tabular}{|c|c|c|}
\hline "Command" & "1st parameter" & "2nd parameter" \\
\hline CreateRecipe & PLC_PRG.stRecipeDef & PLC_PRG.stRecipe_New \\
\hline & \begin{tabular}{l}
Name of the affected recipe definition \\
- As a literal \\
- As a variable (STRING)
\end{tabular} & \begin{tabular}{l}
Name of the new recipe \\
- As a literal \\
- As a variable (STRING)
\end{tabular} \\
\hline \multicolumn{3}{|l|}{At visualization runtime, a new recipe is created in the given recipe definition.} \\
\hline
\end{tabular}

Table 684: Command "Delete Recipe"
\begin{tabular}{|l|l|l|}
\hline "Command" & "1st parameter" & "2nd parameter" \\
\hline DeleteRecipe & PLC_PRG.stRecipeDef & PLC_PRG.stRecipe \\
\hline & \begin{tabular}{l} 
Name of the affected recipe defini- \\
tion \\
\(\bullet \quad\) As a literal \\
\(\bullet \quad\) As a variable (STRING)
\end{tabular} & \begin{tabular}{l} 
Name of the recipe \\
\\
\\
\hline
\end{tabular} \\
\hline
\end{tabular}

Table 685: Command "Print"
\begin{tabular}{|l|l|l|}
\hline "Command" & "1st parameter" & "2nd parameter" \\
\hline Print & \begin{tabular}{l} 
Optional: File name for the vis- \\
ualization screen to be printed \\
(example: 'Start screen')
\end{tabular} & \begin{tabular}{l} 
A second parameter cannot \\
be specified for the Print \\
command.
\end{tabular} \\
\hline \begin{tabular}{l} 
The default "Printer" dialog opens while the visualization is running. In the dialog, you select \\
a printer and configure additional print settings. When you confirm the dialog, the currently \\
displayed visualization screen is printed.
\end{tabular} \\
Note: The command can be executed in the TargetVisu only.
\end{tabular}

\section*{See also}
- Changing Values with Recipes
- Object 'Recipe Definition'

Input action When the input event occurs, the display switches to another visualization within one frame.
'Switch Frame Visualization'
\begin{tabular}{|l|l|}
\hline "Frame selection type" & \\
\hline "Switch local visualization" & The "Frame Selection" group is visible. \\
\hline "Switch to any visualization" & The "Frame and visualization selection" group is visible. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline Requirement: "Switch local visualization" is selected. \\
\hline & \begin{tabular}{l} 
"Frame selection" \\
tions of all frames that contain the active visualization. The referenced visualiza- \\
the respective frame. \\
Example: frame, as determined in the "References" property of \\
- GenElemInst_1 \\
visEllipse
\end{tabular} \\
& \begin{tabular}{l} 
GenElemInst_2 \\
visRectangle
\end{tabular} \\
\hline "Assign selection" & \begin{tabular}{l} 
The selection in the "Frame selection" input field is accepted. Then it appears in \\
the "Selected frame" and "Selected visualization" settings. \\
Requirement: A visualization is selected in the "Frame selection" input field.
\end{tabular} \\
\hline "Selected Frame" & \begin{tabular}{l} 
Name of the frame to be switched to \\
Example: MainArea \\
Hint: Use the "Assign selection" command for changing the setting here.
\end{tabular} \\
\hline "Selected Visu" & \begin{tabular}{l} 
Name of the switched visualization. \\
Example: visMainArea \\
Hint: Use the "Assign selection" command for changing the setting here.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline \multicolumn{2}{|l|}{ Requirement: The "Switch to any visualization" option is selected. } \\
\hline \begin{tabular}{l} 
"Frame and visualization \\
selection"
\end{tabular} & Contains the frame to be switched to \\
\hline "Assign" & \begin{tabular}{l} 
Frame to be switched to (with complete path). The index determines the visuali- \\
zation. \\
Example: visMain.frameA.visB.frameB \\
The path is specified in the following syntax: <visualization \\
name>.<frame name> <visuali zation name>. <frame name> \} \\
Caution: Visualizations can be nested at any depth by means of frame elements. \\
In order to use the "Switch to any visualization" frame selection type without any \\
problems, a frame must not contain more than 21 referenced visualizations.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Assign expression" & \begin{tabular}{l} 
Variable (STRING). Contains the path of the frame to be switched to \\
Example: strFrane: STRING : = 'visMain.frameA.visB.frameB' ;
\end{tabular} \\
\hline "Index to select" & \begin{tabular}{l} 
Index that determines which of the referenced visualizations is displayed \\
\\
\\
\\
\\
\\
\\
\\
\\
• As an integer \\
Example :PLC_PRG.iIndex \\
Note: The referenced visualizations of a frame are indexed automatically \\
according to their order.
\end{tabular} \\
\hline
\end{tabular}

Requirement: The project contains visualizations that form a structure.
See also
- 若 "Dialog 'Frame Configuration"" on page 3348

Input action 'Write Variable'

The configuration of the input action defines how a visualization user specifies a value and to which variable the value is written.

Check all inputs for there validity. Be sure that only values within the range can be added to a numeric field. Depending on the datatype of the input value, the limitations can be different.
\begin{tabular}{|l|l|}
\hline \multicolumn{2}{|l|}{\begin{tabular}{l} 
"Input type" \\
How the input is prompted.
\end{tabular}} \\
\hline "Default": & \begin{tabular}{l} 
An input field also opens, or if necessary a virtual keyboard (when the display \\
variant does not have a physical keyboard). \\
Note: The default option for text input at runtime is set in the Visualization \\
Manager: "Dialog Settings" tab, "Settings for Default Text Input".
\end{tabular} \\
\hline "Text input" & \begin{tabular}{l} 
An input field appears. You use the keyboard to specify a number or a text. \\
Requirement: The display variant has a keyboard as input device.
\end{tabular} \\
\hline "Text input with limits" & \begin{tabular}{l} 
An input field appears. You use the keyboard to specify a number or a text. The \\
field also shows the range of values for the input. When a limit is passed, the \\
input value is displayed in red. \\
Requirement: The display variant has a keyboard as input device.
\end{tabular} \\
\hline "VisuDialogs.Keypad" & A virtual keyboard opens. You use it to specify a number or a text. \\
\hline "VisuDialogs.Numpad" & A virtual keyboard opens. You use it to specify a number. \\
\hline "VisuDialogs.NumpadExtended & \begin{tabular}{l} 
A virtual keyboard opens. You use it to specify a number. Hexadecimal and \\
exponential notation are also permitted here.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline \multicolumn{2}{|l|}{ "Choose variable to edit" } \\
\hline "Use text output variable" & \begin{tabular}{l} 
O: The input value is written to the text output variable of the element. This is \\
the variable that is assigned in the "Text variable \(\rightarrow\) Text" property.
\end{tabular} \\
\hline "Use another variable" & \begin{tabular}{l} 
O: Variable where the input value is written. \\
Example: PLC_PRG. iVariable
\end{tabular} \\
\hline "Initial display format" & \begin{tabular}{l} 
Placeholder with format definition. It defines the output format for the variable \\
value and the input limits. \\
Example: \(\% 2.3 \mathrm{f}\) for displaying the value as a decimal fraction.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Min" & \begin{tabular}{l} 
Minimum value of the input limit. If a user specifies a lesser value, then it is not \\
accepted. \\
- As a fixed value \\
- As a variable (data type corresponds to selected variable)
\end{tabular} \\
\hline "Max" & \begin{tabular}{l} 
Maximum value of the input limit. If a user specifies a greater value, then it is not \\
accepted. \\
- As a fixed value \\
- As a variable (data type corresponds to selected variable)
\end{tabular} \\
\hline "Dialog title" & \begin{tabular}{l} 
Text displayed in the title bar of the dialog. Optional. \\
- As a fixed string
\end{tabular} \\
& \begin{tabular}{l} 
Example: Insert value
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l}
\hline "Password field" & \(\boxed{\checkmark}\) : Unseen text input. *** is shown instead of the input text.
\end{tabular}

Table 686: "Position to open input dialog"
\begin{tabular}{|l|l|}
\hline "Use global setting (from the \\
Visualization Manager)"
\end{tabular}\(\quad\)\begin{tabular}{l} 
O: This option is applies only for use in a TargetVisu or WebVisu. The settings \\
are used which are available in the "Dialog Settings" tab of the Visualization \\
Manager.
\end{tabular}

\section*{See also}
- \({ }^{\text {® }}\) Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329
- \({ }^{*}\), Chapter 6.4.5.21.3.15 "Dialog 'Properties' of Visualization Objects" on page 3388
- \({ }^{\text {• }}\) Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329
- 告"Tab 'Visualization manager' - 'Settings'" on page 3398

Input action 'Ex-
ecute ST Code'
\begin{tabular}{|l|l|}
\hline Input field & Editor for code as Structured Text \\
\hline
\end{tabular}

Input action
'Toggle Variable'
\begin{tabular}{|l|l|}
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL). It toggles between TRUE and FALSE for an input event. \\
Example: PLC_PRG.bSwitch
\end{tabular} \\
\hline
\end{tabular}

Input action 'File With the "File Transfer" input action, a file can be transferred from an operating variant (target or Transfer' web visualization) to the PLC as well as to and from the PLC. This works either by means of a file transfer ("Type": "File") or streaming ("Type": "Streaming").

The action has the effect that a file selection dialog is displayed in the visualization at runtime. There the visualization user can select a file which will be transferred either to or from the PLC: For a transfer from a PLC to the visualization, the "Save File" dialog opens. For the transfer from the visualization to the PLC, the "Open File" dialog opens.

\section*{"Transfer"}
\begin{tabular}{|l|l|}
\hline "Direction" & Direction of file transfer \\
\hline "From PLC to Visualization" & \begin{tabular}{l} 
The object specified in "File name" or "Streaming instance name" is transferred \\
from the PLC to the visualization. \\
The "Save File" file selection dialog is displayed in the visualization at runtime.
\end{tabular} \\
\hline "From Visualization to PLC" & \begin{tabular}{l} 
The file specified by the visualization user is transferred to the PLC and saved in \\
the file path specified in "File name" or "Streaming instance name". \\
The "Open File" file selection dialog is displayed in the visualization at runtime.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Type" & \begin{tabular}{l} 
Determines how the file is transferred \\
\\
\\
\(\bullet\) \\
•Transfer type "File": By file transfer \\
\\
\hline
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Type" & \begin{tabular}{l}
"File" \\
The data transfer is done by file transfer.
\end{tabular} \\
\hline "File name" & \begin{tabular}{l}
File path (type STRING) which describes the file in the file system \\
- Variable \\
Example: strTransferFile: STRING; \\
- Literal with relative path \\
Example: '/Recipes/Recipe_1.txt' saves the file in the directory Recipes. \\
- Literal with placeholder \$PLCLOGIC\$ \\
PlcLogic is the default resolution for the directory placeholder \$PLCLOGIC\$. \\
Example: ' \(\$ \$\) PLCLOGIC\$\$/test.txt' saves the file in the directory PlcLogic. \\
Example: ' \(\$\) PLCLOGIC\$\$/MyData/test.txt' saves the file in the directory PlcLogic/MyData. \\
- Literal with placeholder \$VISU\$ visu is the default resolution for the placeholder \$VISU\$. \\
Example: ' \(\$\) SVISU\$\$/test.txt' save the file in the subdirectory PlcLogic/visu. Alternatively, 'visu/test.txt' can also be specified. \\
- Literal with absolute path \\
Example: 'E: \temp\test.txt' \\
Note: These kinds of file paths are not always supported. \\
Note: If a user specifies the file path in the visualization by means of a "Text Field" element, the masking character \$ must not be included: \$VISU\$ / dummy.txt \\
Note: In the case that the file path is specified by the user, it should be checked by the application in order to prevent files from being read or overwritten accidentally.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Type" & "Streaming" \\
The data transfer is done by streaming.
\end{tabular}\(|\)\begin{tabular}{ll|}
\hline "Streaming instance name" & \begin{tabular}{l} 
Instance path (type IVisuStreamWriter or IVisuStreamReader) which \\
describes the object in the file system of the controller \\
Type IVisuStreamReader for transfer direction "From PLC to Visualization" \\
Type IVisuStreamWriter for transfer direction "From Visualization to PLC"
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Control flags" & \begin{tabular}{l}
Note: The variable is evaluated only for transfer direction "From Visualization to PLC". \\
Variable (type DWORD) \\
Determines how the object (file or instance object) is handled on the file system of the PLC. Two flags are provided for this with which the variable can be set. \\
- Flag 1: \\
VisuElems.VisuEnumFileTransferControlFlags.UseOriginalFi leName \\
- Flag 2 : \\
VisuElems.VisuEnumFileTransferControlFlags.ConfirmFileOv erwriteInPlc \\
Options \\
- No flag set: \\
The user selects a file which is saved in the path specified in "File name" or "Streaming instance name". \\
- Flag 1 is set: \\
The path, which is specified by the user at visualization runtime, is applied and used as the path in the PLC file system. \\
- Both flags are set: \\
The path is also checked. If an object already exists in the path specified on the client side, then a message prompt is displayed in the visualization. There the visualization user can confirm that the file will be overwritten. \\
Example: dwControlFlag
\end{tabular} \\
\hline
\end{tabular}
```

Example The transfer direction is "From Visualization to PLC" (write).
Example: A new recipe file Recipes/Recipe_2021.txt has been created in the visualiza-
tion device. The visualization user selects this file and wants to save the file on the PLC under
the same name. Because the control flags are set accordingly, a message window opens and
the visualization user can confirm that the file will be overwritten.

```
```

PROGRAM PLC_PRG

```
PROGRAM PLC_PRG
VAR
VAR
    xVisuToggle : BOOL;
    xVisuToggle : BOOL;
    dwControlFlag : DWORD:=
    dwControlFlag : DWORD:=
VisuElems.VisuEnumFileTransferControlFlags.UseOriginalFileName +
VisuElems.VisuEnumFileTransferControlFlags.UseOriginalFileName +
VisuElems.VisuEnumFileTransferControlFlags.ConfirmFileOverwriteInPlc
VisuElems.VisuEnumFileTransferControlFlags.ConfirmFileOverwriteInPlc
;
;
    strFileName: STRING := '/Recipes/Recipe_new.txt';
    strFileName: STRING := '/Recipes/Recipe_new.txt';
END_VAR
```

END_VAR

```

Table 687: "Status Variables"
\begin{tabular}{|c|c|}
\hline "Transfer active" & Boolean variable (optional) TRUE: The transfer is in progress. \\
\hline "Transfer successful" & Boolean variable (optional) TRUE: The transfer has completed successfully. \\
\hline "Error code" & \begin{tabular}{l}
- 0: No errors \\
- 1: Unspecified error \\
- 2: Cancellation of file dialog \\
- 3: Other file transfer in progress \\
- 4: Error during file transfer \\
- 5: Cancellation by timeout \\
- 6: File read error - The file is not available or cannot be read. \\
- 7: No device support for file transfer \\
Possible causes: \\
- CODESYS WebVisu: File transfer is not possible by default. \\
- Communication with a controller of a version < 3.5.11: Functionality not implemented. \\
- Communication with a controller of a version >=3.5.11: File transfer not activated (device description). \\
Note: In this case, contact the CODESYS support team.
\end{tabular} \\
\hline
\end{tabular}

\section*{Dialog 'Options' - 'Visualization Styles'}

Symbol: 略
Function: This dialog is used for configuring the display of library visualizations and visualizations in the POUs view in the visualization editor. In addition, it is used for configuring the tab "Visualization Manager" - "Settings" (group "Style Settings").
Call: Menu bar: "Tools \(\rightarrow\) Options" ("Visualization Styles" category).
These settings are not applied at visualization runtime. In runtime mode, only the settings of the visualization manager are available in the "Settings" tab.

See also
- © Chapter 6.4.5.19 "Applying Visualization Styles" on page 2979

\section*{'Style Configu-}
ration for Libraries and Global Visualizations'
\begin{tabular}{|l|l|}
\hline These settings are applied for library visualizations and for visualizations in the POUs view. \\
\hline "Use no visualization style" & \begin{tabular}{l} 
○: Display without using style properties. Elements are displayed as defined by \\
presets.
\end{tabular} \\
\hline \begin{tabular}{l} 
"Use the following visualization \\
style"
\end{tabular} & ○: Style with style properties used for displaying visualizations. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
"Derive visualization style \\
automatically"
\end{tabular} & \begin{tabular}{l} 
○: Display with the style that was selected in the application in the visualization \\
manager (when possible). Therefore, the display is derived from this style. \\
It is actually possible for this to cause an incorrect display. Then the fallback \\
solution is used.
\end{tabular} \\
\hline \begin{tabular}{l} 
"Fallback if no visualization \\
style could be derived "
\end{tabular} & \begin{tabular}{l} 
Another style that is applied after the selected style. Then a style property is \\
assigned from the style specified here. This is done for element properties that \\
could not be assigned style properties.
\end{tabular} \\
\begin{tabular}{l} 
Requirement: The selected style causes a device-specific, deficient display on \\
the display variant.
\end{tabular} \\
\hline
\end{tabular}

\section*{'Style Selection'}

The drop-down list of "Selected style" can be configured in the visualization manager ("Settings" tab, "Style settings" group).
"Display all versions"
All other styles of the repository, including the selected style, are listed for selection, but only in the latest version. If newer versions are installed for the selected style, then these are also listed.

All installed styles in all installed versions are available for selection.

\section*{'Style for New \\ Visualization \\ Managers'}
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
"Last used: <style, version, \\
vendor>"
\end{tabular} & \begin{tabular}{l} 
Style that is selected automatically when you add a new visualization application. \\
Note: It is actually possible that a display variant is displayed another way
\end{tabular} \\
\begin{tabular}{ll} 
"Preset: <style, version, \\
vendor>"
\end{tabular} & \\
\hline "<style, version, vendor>" & \\
\hline
\end{tabular}

\section*{Dialog 'Options' - 'Visualization User Management'}

Symbol:
Function: The options define the use of visualization user management for global visualizations in the "POUs" view and for visualizations that are linked from libraries.

Call: Menu bar: "Tools".
Requirement: A visualization user management exists.

Table 688: "User Management Configuration for Libraries and Global Visualizations"
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
"Do not use visualization user \\
management"
\end{tabular} & \begin{tabular}{l} 
The affected visualizations behave as when no user management is configured. \\
"Use the following visualization \\
user group list"
\end{tabular} \\
\hline \begin{tabular}{l} 
" You can edit the list. \\
\(\bullet \quad\)\begin{tabular}{l} 
The list is created in the "Visualization manager" " "User management \\
\(\rightarrow\) Groups") by clicking "Export groups for global visualizations".
\end{tabular} \\
\hline management automatically"
\end{tabular} & \begin{tabular}{l} 
The affected visualizations use the user management configuration of the visual- \\
ization manager selected here. \\
The drop-down list shows all visualization managers of the project. \\
If this is not possible, then the user groups are used from the option "Use the \\
following user group list for the visualization".
\end{tabular} \\
\hline
\end{tabular}

The user management for a visualization in the "Devices" view is configured in the "Visualization Manager" (tab "User Management").

See also
- \({ }^{\star}\) Chapter 6.4.5.21.4.5 "Tab 'Visualization manager' - 'User management"" on page 3403

\section*{Dialog Box 'Options' - 'Visualization'}

Symbol: 田
Function: The dialog serves for the configuration of the visualization editor and during runtime it serves the configuration of the Integrated Visualization.
Call: Main menu "Tools \(\boldsymbol{\rightarrow}\) Options", category "Visualization"

\section*{Tab 'General'}

These settings will not be applied for the following visualization clients:
CODESYS TargetVisu, CODESYS WebVisu.

Table 689: "Presentation options (visualization editor in the programming system)"
\begin{tabular}{|l|l|}
\hline "Fixed" & The visualization maintains its original size \\
\hline "Isotropic" & The visualization maintains its proportions \\
\hline "Anisotropic" & The visualization adapts to the size of the visualization window \\
\hline "Antialiased Drawing" & \begin{tabular}{l}
\(\boxed{\bigvee}:\) The visualization is drawn with the help of antialiasing methods. This applies \\
while you are editing and also when the visualization is running as Diagnosis \\
Visualization.
\end{tabular} \\
\hline
\end{tabular}

Table 690: "Editing options"
"Link to toggle/tap variable when appropriate"
, The placeholder "<toggle/tap variable>" in the visualization element properties is enabled.

Effect: If you drag an element having the property "Color variable
\(\rightarrow\) Toggle color" in the visualization editor, this property will be configured with the placeholder "<toggle/tap variable>".
The following elements are affected: "Button", "Frame", "Image", "Line", "Pie", "Polygon", "Rectangle", "Text field", "Scrollbar".

See also
- \({ }^{4}\) Chapter 6.4.5.21.5.11 "Visualization Element 'Button"' on page 3514
- \({ }^{4}\) Chapter 6.4.5.21.5.6 "Visualization Element 'Frame'" on page 3478
- \({ }^{2}>\) Chapter 6.4.5.21.5.5 "Visualization Element 'Image'" on page 3463
- ② Chapter 6.4.5.21.5.2 "Visualization Element 'Line'" on page 3425
- 乡y Chapter 6.4.5.21.5.4 "Visualization Element 'Pie"' on page 3450
- \(\Leftrightarrow\) Chapter 6.4.5.21.5.3 "Visualization Element 'Polygon', 'Polyline', 'Bézier Curve'" on page 3437
- \(\gg\) Chapter 6.4.5.21.5.1 "Visualization Element 'Rectangle', 'Rounded Rectangle', 'Ellipse"" on page 3413
- « \({ }^{*}\) Chapter 6.4.5.21.5.14 "Visualization Element 'Text Field'" on page 3538
- « Chapter 6.4.5.21.5.15 "Visualization Element 'Scroll Bar'" on page 3550

Tab 'Grid'
\begin{tabular}{|l|l|}
\hline "Visible" & \begin{tabular}{l} 
V: The visualization editor contains a grid. The spacing of the grid lines is \\
defined by "Size".
\end{tabular} \\
\hline "Active" & \begin{tabular}{l} 
目: The visualization elements get aligned to the grid, defined by "Size", even \\
if the grid lines are not visible. When you insert or move an element, its center \\
will be positioned on the grid. When you modify the size of an element, you can \\
move the position markers onto grid lines only. Elements already available in a \\
visualization, will not be aligned automatically, until you change their position.
\end{tabular} \\
\hline & Spacing of the grid lines in pixel. \\
\hline "Size" &
\end{tabular}

Tab 'File
options'
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
"Text list files for textual \\
"IntelliSense"
\end{tabular} & \begin{tabular}{l} 
File name and path of a file of type . csv. The file contains texts in the format of \\
a text list. \\
The file entries will be available when using the function "List Components" as \\
input assistance. \\
Note: You can create this file as an export file of the global text list. For this \\
purpose use the command "Import/Export Text Lists".
\end{tabular} \\
\hline "Visualization Directories" & \\
\hline "Text list files" & \begin{tabular}{l} 
Storage path for text lists. \\
Note: This setting will be used in CODESYS only if no storage path for "Text list \\
files" is defined in the "Project Settings", category"Visualization".
\end{tabular} \\
\hline "Image files" & \begin{tabular}{l} 
Storage path for image files. Multiple paths get separated by semicolons. \\
CODESYS uses this path for example when exporting or importing image files. \\
Note: This setting will be used in CODESYS only if no storage path for "Image \\
files" is defined in the "Project Settings", category"Visualization".
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{4}\) Chapter 6.4.1.9.10 "Managing text in text lists" on page 1891
- \({ }^{*}\) Chapter 6.4.1.21.3.21.6 "Command 'Import/Export Text Lists'" on page 2729

See also
- « Chapter 6.4.5.21.3.13 "Dialog 'Project Settings' - 'Visualization'" on page 3387
- « Chapter 6.4.5.21.4.7 "Object 'TargetVisu'" on page 3408

\section*{Dialog 'Project Environment' - 'Visualization Profile'}

Function: The dialog displays the current visualization profile of the project. The profile can be updated here.
Call: Main menu "Project \(\rightarrow\) Project Environment", Tab "Visualization Profile".
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
"Current visualization profile in \\
project"
\end{tabular} & The currently set visualization profile of the opened project. \\
\hline \begin{tabular}{l} 
"Recommended, newest \\
profile"
\end{tabular} & The newest profile \\
\hline "Action" & \\
\hline "Do not update" & The visualization profile of the project remains unchanged. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Update to x.x.x.X" & CODESYS updates the project to the chosen visualization profile. \\
\hline \begin{tabular}{l} 
"Check for updates when \\
loading this project"
\end{tabular} & \begin{tabular}{l} 
: CODESYS checks for new profiles when the project is opened. If there are \\
updates available an update dialog opens automatically. \\
\(\square\) \\
open automatically any longer.
\end{tabular} \\
\hline "Set All to Newest" & CODESYS updates the profile. \\
\hline
\end{tabular}

\section*{Dialog 'Project Environment' - 'Visualization Styles'}

Function: The dialog displays the currently used visualization style of the project. The visualization style can be updated here.
Call: Main menu "Project \(\rightarrow\) Project Environment", tab "Visualization Styles"
\begin{tabular}{|l|l|}
\hline "For the following visualization styles currently in use, newer versions are available:" \\
\hline "Visualization Styles" & Version of the currently set visualization style of the opened project. \\
\hline "Current" & Current version of the visualization style, for example 3.5.6.0 \\
\hline "Recommended" & Recommenden version of the visualization style, for example 3.5.7.0 \\
\hline "Action" & The visualization style of the project remains unchanged. \\
\hline "Do not update" & CODESYS updates the project to the version of the chosen visualization style. \\
\hline "Update to x.x.x.x" & \begin{tabular}{l} 
"Check for updates when \\
loading the project"
\end{tabular} \\
\begin{tabular}{l} 
updates available an update dialog opens automatically. \\
and \\
: No check of the version. The update dialogs do not open automatically any \\
longer.
\end{tabular} \\
\hline "Set All to Newest" & CODESYS updates the version. \\
\hline
\end{tabular}

See also
- \# Chapter 6.4.5.19 "Applying Visualization Styles" on page 2979

\section*{Dialog 'Project Environment' - 'Visualization Symbols'}

Function: The dialog lists installed symbol libraries and allows for you to assign symbol libraries to a project.
Call: Menu bar: "Project \(\rightarrow\) Project Environment", "Visualization Symbols" tab
Requirement: The open project contains a visualization and has been saved with a compiler version < 3.5.7.0. CODESYS recognizes symbol libraries in compiler version 3.5.7.0 and higher.
\begin{tabular}{|l|l|}
\hline "Symbol library" & List of all installed symbol libraries \\
\hline "Active" & \begin{tabular}{l} 
岛: Symbol library is selected for the project. CODESYS provides its symbols in \\
the "Visualization Toolbox" view. \\
\\
\(\square\)
\end{tabular} \\
\hline
\end{tabular}

See also
- Chapter 6.4.5.5.2 "Select Element" on page 2873

\section*{Dialog＇Project Settings＇－＇Visualization＇}

\section*{Symbol：田}

Function：The dialog is used to configure the project－wide settings for objects of type ＂Visualization＂．

Call：Menu bar：＂Project \(\rightarrow\) Project Settings＂，＂Visualization＂category
Requirement：A project is open．

\section*{Tab＇General＇}

Table 691：＂Visualization Directories＂
\begin{tabular}{|l|l|}
\hline ＂Text list files＂ & \begin{tabular}{l} 
Directory which contains text lists that are available in the project to configure \\
texts for different languages．CODESYS uses the directory，for example to \\
import or export text lists．
\end{tabular} \\
\hline ＂Image files＂ & \begin{tabular}{l} 
After clicking \\
tion of a directory in the file system．
\end{tabular} \\
\hline & \begin{tabular}{l} 
Directory which contains image files that are available in the project．Multiple \\
folders are separated with a semicolon．CODESYS uses the directory，for \\
example to import or export image files． \\
After clicking \(\ldots, . .\). the＂Select Directory＂dialog opens which allows for the selec－ \\
tion of a directory in the file system．
\end{tabular} \\
\hline
\end{tabular}

Table 692：＂Advanced＂
＂Activate property handling in all element properties＂
\(\checkmark\) ：You can also configure a visualization element with a property 異 in those of its properties in which you select an IEC variable．Then CODESYS creates additional code for the property handling when a visualization is compiled．
Requirement：Its IEC code contains at least an object of type＂Interface property＂ （a property 那）．

\section*{PLC＿PRG（PRG）}

Property＿A
Get
Set
Requirement：＂Visible＂is selected．

See also
－Object＇Property＇

\section*{Tab＇Symbol}

\section*{Libraries＇}

Table 693：＂Visualization Symbol Libraries＂
\begin{tabular}{|l|l|}
\hline ＂Symbol libraries＂ & \begin{tabular}{l} 
List of all installed symbol libraries \\
Example：VisuSymbols
\end{tabular} \\
\hline ＂Assigned＂ & \begin{tabular}{l} 
Ø：Symbol library is selected in the project and CODESYS makes it available in \\
the＂Visualization ToolBox＂view of a visualization． \\
\(\square: S y m b o l ~ l i b r a r y ~ i s ~ i n s t a l l e d ~ i n ~ t h e ~ l i b r a r y ~ r e p o s i t o r y, ~ b u t ~ C O D E S Y S ~ d o e s ~ n o t ~\) \\
make it available in the＂Visualization ToolBox＂view of a visualization．
\end{tabular} \\
\multicolumn{2}{|l|}{\begin{tabular}{l} 
See also \\
• « Chapter 6．4．5．21．3．2＂Dialog＇Add Visualization＂＂on page 3367
\end{tabular}}
\end{tabular}

\section*{Dialog 'Project Settings' - 'Visualization Profile'}

Symbol: 田
Function: The dialog box enables the setting of the visualization profile.
Call: Menu "Project \(\rightarrow\) Project Settings", category "Visualization Profile"
Requirement: A project is open.

Table 694: "Visualization Profile"
\begin{tabular}{|l|l|}
\hline "Specific profile" & \begin{tabular}{l} 
Profile that CODESYS uses in the project and that determines the visualization \\
elements that are available in the project. \\
The selection list contains all the profiles installed so far.
\end{tabular} \\
\hline
\end{tabular}

\section*{Dialog 'Properties' of Visualization Objects}

Function: This dialog is used for configuring object-dependent properties.
Call: Menu bar: "View"; context menu of the visualization object in the "Devices" view or " POUs" view.

Tab 'General'
\begin{tabular}{|l|l|}
\hline "Name" & Example: visMain \\
\hline "Object type" & visualization \\
\hline "Open with" & visualization \\
\hline
\end{tabular}

See also
- \(\Leftrightarrow\) Chapter 6.4.1.21.4.11.2 "Dialog Box 'Properties' - 'Common"' on page 2753

Tab 'Access This tab is used for defining which user group can execute which actions on the object. Control' See also
- \(\boldsymbol{y y}^{\boldsymbol{y}}\) Chapter 6.4.1.21.4.11.7 "Dialog 'Properties' - 'Access Control"" on page 2757

Tab 'Visualiza- This tab assigns a visualization type to a visualization.
tion'
In addition, it includes settings for window size that are used at runtime.

Table 695: "Use visualization as"
\begin{tabular}{|l|l|}
\hline "Visualization" & Visualization type for an ordinary or referenced visualization. Preset. \\
\hline "Dialog" & \begin{tabular}{l} 
Visualization type for a visualization that opens as a dialog in its own window for \\
an input event. The input action for this is "Open dialog". The "Close dialog" input \\
action closes the window. \\
Tip: A dialog usually includes an "OK" button or "Cancel" button at the bottom \\
edge for confirming or rejecting user input, and for closing the dialog. A simple \\
dialog or a dialog prompt includes only a question or information and buttons for \\
closing the dialog with either "Yes" or "No". A dialog is part of a user interface. \\
While a dialog is open, the rest of the user interface is usually disabled.
\end{tabular} \\
\hline "Numpad / keypad / dialog for \\
input configuration" & \begin{tabular}{l} 
Visualization type for a visualization that displays a virtual numeric keypad or a \\
virtual keyboard. It appears when the user is prompted to specify text. The input \\
action for this is "Write variable". \\
Note: The interface of this visualization must also conform with the interfaces \\
for the standard visualizations for the numeric keypad or that keyboard that \\
provides the VisuDialogs library: Numpad, Keypad, NumpadExtended, or \\
TextinputwithLimits. \\
Tip: The VisuDialogs library contains templates for virtual keyboards or \\
numeric keypads.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Dialog is opaque" & \begin{tabular}{l}
\(\boxed{\square}\) \\
positive effect on the character and input performance. \\
Use this option when your drawn dialog is rectangular and opaque, containing no \\
transparent parts.
\end{tabular} \\
\hline "Use automatic detected & o: The size is determined so that all visualization elements are enclosed. \\
visualization size" & "Include background image" \\
\hline \begin{tabular}{l} 
: All elements and the background image are completely visible. \\
\(\square\)
\end{tabular} \\
\hline \begin{tabular}{l} 
"Use specified visualization elements are visible, but a larger background image is truncated. \\
size"
\end{tabular} & \begin{tabular}{l} 
o: The values "Height" and "Width" define the window size of the visualization \\
(in pixels).
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Internal" & \begin{tabular}{l} 
: The visualization is internal. It is used exclusively as an internal module of a \\
complete visualization in a library. \\
\\
\\
\\
\\
\\
When editing as a library project while the project is open in CODESYS, an \\
internal visualization is handled like all visualizations. The internal visualization \\
appears in drop-down lists. Or in the visualization manager ("Visualizations" tab). \\
The internal visualizations that include a linked library are not visible to you.
\end{tabular} \\
\hline
\end{tabular}

See also
- ⓢ "Dialog 'Frame Configuration'" on page 3348

Tab 'Build' This tab includes options for compiling the object.
See also
- \(\Leftrightarrow\) Chapter 6.4.1.21.4.11.5 "Dialog 'Properties' - 'Build"' on page 2755

\section*{Dialog 'Selected Alarm Class'}

Function: In this dialog box, you define the alarm classes that are considered for the alarm table or alarm banner.

Call: Property "Alarm configuration" / "Alarm classes" of the alarm table or alarm banner visualization element.
Requirement: An alarm table visualization element or alarm banner visualization element is added to the visualization.
\begin{tabular}{|l|l|}
\hline "Available Alarm Classes" & Shows all alarm classes created in the project. \\
\hline "Selected Alarm Classes" & The alarm classes in this column are displayed in the alarm table. \\
\hline "All" & \(\boxed{\square}:\) All alarm classes are listed in an alarm table. \\
\hline\(\gg\) & Moves all available alarm classes to the "Selected Alarm Classes" column. \\
\hline\(>\) & Moves the selected alarm classes to the "Selected Alarm Classes" column. \\
\hline\(>\) & Removes the selected alarm classes from the "Selected Alarm Classes" column. \\
\hline\(\ll\) & Removes all selected alarm classes from the "Selected Alarm Classes" column. \\
\hline
\end{tabular}

See also
- 垒 Chapter 6.4.5.21.5.22 "Visualization Element 'Alarm Table"" on page 3591
- \(\sum^{*}\) Chapter 6.4.5.21.5.22 "Visualization Element 'Alarm Table"' on page 3591
- \({ }^{4}\) Chapter 6.4.5.9 "Visualizing alarm management" on page 2909

\section*{Dialog 'Selected Alarm Group'}

Function: In this dialog box, you define the alarm groups that are considered for the alarm table or alarm banner.
Call: Property "Alarm configuration" / "Alarm groups" of the alarm table or alarm banner visualization element.
Requirement: An alarm table visualization element or alarm banner visualization element is added to the visualization.
\begin{tabular}{|c|c|}
\hline "Available Alarm Groups" & Shows all alarm groups created in the project. \\
\hline "Selected Alarm Groups" & The alarm groups in this column are displayed in the alarm table. \\
\hline "All" & V: All alarm groups are listed in an alarm table. \\
\hline >> & Moves all available alarm groups to the "Selected Alarm Groups" column. \\
\hline > & Moves the selected alarm groups to the "Selected Alarm Groups" column. \\
\hline - & Removes the selected alarm groups from the "Selected Alarm Groups" column. \\
\hline << & Removes all alarm groups from the "Selected Alarm Groups" column. \\
\hline
\end{tabular}

See also
- \({ }^{4}\) Chapter 6.4.5.21.5.22 "Visualization Element 'Alarm Table'" on page 3591
- \(\sum^{*}\) Chapter 6.4.5.21.5.22 "Visualization Element 'Alarm Table"' on page 3591
- \({ }^{4}\), Chapter 6.4.5.9 "Visualizing alarm management" on page 2909

\section*{Dialog 'Advanced Trace Settings'}

Function: The recording rate of the "Trace" visualization element is configured in this dialog box.

Call
- Properties: "Trace" as, "Advanced"
- Context menu: "Configure trace", "Advanced"

Requirement: A trace is selected in the active visualization editor.

Table 696
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
"Measurement in every nth \\
cycle"
\end{tabular} & \begin{tabular}{l} 
The task where the trace is running is the basis for the measurement. The \\
measurement interval is a multiple of the trace task according to the selected \\
value. The measurement interval is displayed on the right side.
\end{tabular} \\
\hline "Buffer size (samples)". & \begin{tabular}{l} 
The number of measurements is calculated according to the time range of the \\
x-axis.
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{*}\), Chapter 6.4.5.21.2.13 "Command 'Configure Trace'" on page 3355

\section*{Dialog 'Display Settings'}

Function: The dialog includes the configuration for the display settings of the trace diagram (for both the X -axis and Y -axis) and provides a preview in the trace diagram.
Call: "Display" button in "Trace Configuration" dialog.
\begin{tabular}{|l|l|}
\hline "Add Y-Axis" & \begin{tabular}{l} 
Extends the trace diagram by one Y-axis. \\
Result: The "Trace Configuration" dialog contains an extended selection of Y- \\
axes in the "Additional axes" option of the variable settings.
\end{tabular} \\
\hline "Delete Y-axis" & Deletes the Y-axis with the visible tab. \\
\hline
\end{tabular}

Tab 'X-axis'
\begin{tabular}{|c|c|}
\hline "Display Mode" & \begin{tabular}{l}
Scaling \\
- "Auto": \(\boxtimes\) : CODESYS Scales automatically. \\
- "Fixed length": \(\downarrow\) : CODESYS displays a segment of constant "Length". \\
- "Fixed": \(\boxed{\square}\) : CODESYS displays a segment from "Minimum" to "Maximum".
\end{tabular} \\
\hline "Minimum" & Initial value of the segment. Requirement: The "Display Mode" is "Fixed". \\
\hline "Maximum" & End value of the segment. Requirement: The "Display Mode" is "Fixed". \\
\hline "Length" & Constant length of the segment. \\
\hline "Grid" & Diagram with vertical grid lines. Select the line color from the list box of colors. \\
\hline
\end{tabular}

Table 697: "Tick marks"
\begin{tabular}{|l|l|}
\hline "Fixed spacing" & \(\boxed{\square}:\) CODESYS draws tick marks with "Distance" and "Subdivisions". \\
\hline "Distance" & Distance between tick marks \\
\hline "Subdivisions" & Number of subdivisions between two tick marks \\
\hline
\end{tabular}
\begin{tabular}{|l|l}
\hline "Font" & Font for the X-axis
\end{tabular}

Tab 'Y-Axis'
\begin{tabular}{|c|c|}
\hline "Display Mode" & \begin{tabular}{l}
Scaling \\
- "Auto": \(\square\) : CODESYS Scales automatically. \\
- "Fixed": \(\checkmark\) : CODESYS displays a segment from "Minimum" to "Maximum".
\end{tabular} \\
\hline "Minimum" & \begin{tabular}{l}
Literal, variable (integer data type), or constant variable (integer data type). It contains the initial value of the segment. Requirement: The "Display Mode" is "Fixed". \\
Examples: 20,PLC_PRG.iLimit_Min, GVL.c_iLimit_Min \\
Note: The variable has to have an initial value. This is important for the offline display and the scaling subdivision. Example: iLimit_Min : INT := 20
\end{tabular} \\
\hline "Maximum" & \begin{tabular}{l}
Literal, variable (integer data type), or constant variable (integer data type). It contains the end value of the segment. Requirement: The "Display Mode" is "Fixed". \\
Examples: 80,PLC_PRG.iLimit_Max, GVL.c_iLimit_Max \\
Note: The variable has to have an initial value. This is important for the offline display and the scaling subdivision. Example: \(\qquad\)
\end{tabular} \\
\hline "Grid" & \(\checkmark\) : Diagram with a grid line. Select the line color from the list box of colors. \\
\hline "Label" & \(\checkmark\) : The description is displayed on the axis. \\
\hline
\end{tabular}

Table 698: "Tick marks"
\begin{tabular}{|l|l|}
\hline "Fixed spacing" & V: CODESYS draws tick marks with "Distance" and "Subdivisions". \\
\hline "Distance" & Distance between tick marks \\
\hline "Subdivisions" & Number of subdivisions between two tick marks \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Font" & Font for the Y-axis \\
\hline
\end{tabular}

\section*{Preview of the} trace diagram

\begin{tabular}{|l|l|}
\hline "Background color" & \(\bullet\) \\
& \begin{tabular}{l} 
• "No background": Transparent display without background color. \\
\(\bullet \quad\) "Fraw background": Background color according to selection below. \\
style.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Reset" & CODESYS resets all settings to the defaults. \\
\hline "Use as default" & CODESYS saves the settings as default \\
\hline
\end{tabular}

\subsection*{6.4.5.21.4 Objects}
6.4.5.21.4.1 Object 'Visualization' and visualization editor ..... 3393
6.4.5.21.4.2 Object 'Visualization manager' ..... 3398
6.4.5.21.4.3 Tab 'Visualization Manager' - 'Default Hotkeys' ..... 3402
6.4.5.21.4.4 Tab 'Visualization manager' - 'Visualizations'. ..... 3402
6.4.5.21.4.5 Tab 'Visualization manager' - 'User management'. ..... 3403
6.4.5.21.4.6 Tab 'Visualization Manager' - 'Font'. ..... 3407
6.4.5.21.4.7 Object 'TargetVisu'. ..... 3408
6.4.5.21.4.8 Object 'WebVisu'. ..... 3409

\section*{Object 'Visualization' and visualization editor}

Symbol:
The object represents a single visualization. You can insert a visualization under an application or, so that it is available project-wide, under the root node of the view "Devices" or directly in the view "POUs". You can open the visualization editor for editing by double-clicking on the object entry in the device tree or in the view POUs.
See also
- \(\quad\) Chapter 6.4.5.21.3.15 "Dialog 'Properties' of Visualization Objects" on page 3388
- \(\Leftrightarrow\) Chapter 6.4.5.21.4.1.1 "Visualization Editor" on page 3393

\section*{Visualization Editor}

The visualization editor opens when you double-click a visualization object.

- (1) Graphical editor: Here you create the visualization from the visualization elements which are provided in the visualization toolbox view.
- (2) View "Visualization Toolbox": available visualization elements
- (3) View "Properties": Configuration editor for the visualization element currently selected in the editor area
- (4) Menu "Visualization": Commands for working in the visualization editor

The "Visualization" menu contains, for example, commands for opening additional editors.
- (5) "Interface Editor": Declaration of variables which can be used to parameterize references of the visualization.
- (6) "Hotkeys Configuration": Definition of keyboard shortcuts for inputs on the visualization in online mode.
- (7) "Element List": List of all elements used in the visualization; possibility to change their position on the Z-axis.

See also
- \(y^{*}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872
- \#y Chapter 6.4.5.6.5 "Configuring Keyboard Shortcuts" on page 2892
- \({ }^{4}\) Chapter 6.4.5.17.3 "Calling a Visualization with an Interface" on page 2951
- \(\quad y^{\prime}\) Chapter 6.4.5.21.4.1.2 "View 'Visualization Toolbox'" on page 3394
- \(\quad\) Chapter 6.4.5.21.4.1.3 "View 'Properties' of a visualization element" on page 3396
- \(\xi^{*}\) Chapter 6.4.5.21.2 "Commands" on page 3339
- \(\sum^{*}\) Chapter 6.4.5.21.2.1 "Command 'Interface Editor'" on page 3340
- \({ }^{4}\), Chapter 6.4.5.21.2.2 "Command 'Keyboard Configuration'" on page 3341
- \({ }^{4}\) Chapter 6.4.5.21.2.3 "Command 'Visualization Element List"' on page 3342

\section*{View 'Visualization Toolbox'}

\section*{Symbol: \({ }^{8}\)}

Function: The view provides the elements that can be used in the editor. The individual elements are assigned with specific categories. There is a button for each category. The elements of selected categories are displayed with thumbnails which can be dragged into the editor. In addition to the standard categories, you can also define your own categories. You can resize the thumbnails with the slider or perform a full-text search of element names.

\section*{Call: Menu bar: "View \(\rightarrow\) "}

Requirement: A visualization is active.

- (1) "Visualization Toolbox" view
- (2) Toolbar with commands
- (3): Buttons fro selecting element categories
- (4) Selection of individual visualization elements
- (5) Controls

See also
- Command 'Toolbox'

Toolbar with commands
\begin{tabular}{|l|l|}
\hline Symbol：葍 & Only one button can be selected． \\
\hline Symbol：葍 & Multiple buttons can be selected． \\
\hline Symbol：国 & The＂Configure Categories and Items＂dialog opens． \\
\hline
\end{tabular}

See also
－\(\xi^{\wedge}\) Chapter 6．4．5．21．3．4＂Dialog＇Configure Categories and Items＇＂on page 3368

Buttons for selecting ele－ ment categories
\begin{tabular}{|l|l|}
\hline ［Shift \(]+\) click a button & \begin{tabular}{l} 
Changes the selection of the category and the selection type（single or multiple \\
selection possible）
\end{tabular} \\
\hline Right－click a button & The context menu opens． \\
\hline
\end{tabular}

Table 699：Context menu of a button
\begin{tabular}{|l|l|}
\hline ＂Hide Category＂ & Removes the button．Then the category is removed from view． \\
\hline ＂Enable Category＂ & \begin{tabular}{l} 
The button turns green and the category is enabled，irrespective of the selection \\
type．
\end{tabular} \\
\hline ＂Disable Category＂ & \begin{tabular}{l} 
The button turns gray and the category is disabled，irrespective of the selection \\
type．
\end{tabular} \\
\hline
\end{tabular}

See also
－\({ }^{4}\) Chapter 6．4．5．5．2＂Select Element＂on page 2873
－\({ }^{4}\) Chapter 6．4．5．21．3．4＂Dialog＇Configure Categories and Items＇＂on page 3368
－Command＇Toolbox＇

Selection of vis－The visualization elements are displayed as thumbnails and labeled with names．The selection ualization ele－depends on the search query in or on the chosen buttons．
ments
\begin{tabular}{|l|l|}
\hline Slider & To resize of the thumbnails． \\
\hline ＂with input field & For a full－text search by element name of all available elements \\
\hline ＂＜number＞items＂ & \begin{tabular}{l} 
Number of visualization element items that are currently displayed as a result of \\
the selected buttons and the search query in \\
\end{tabular} \\
\hline
\end{tabular}

\section*{View＇Properties＇of a visualization element}

Symbol：
Function：This view is used for configuring the element properties of the selected visualization element．

Call：Menu bar：＂View \(\rightarrow\) Element Properties＂
See also
－② Chapter 6．4．5．5＂Designing a visualization with elements＂on page 2872

\section*{Menu bar}

Table 700: "Filters"
\begin{tabular}{|l|l|}
\hline "All categories" & List of all element properties \\
\hline "Default" & List of the most frequently used element properties \\
\hline "Simple" & \begin{tabular}{l} 
List of certain basic element properties, such as "Texts", "Colors", and "Input \\
configuration"
\end{tabular} \\
\hline "Animation" & List of element properties for animation with variables \\
\hline "Colors" & List of element properties for designing with color \\
\hline "Texts" & List of element properties for designing with text \\
\hline "Input" & List of element properties for configuring user input \\
\hline
\end{tabular}

Table 701: "Sort"
\begin{tabular}{|l|l|}
\hline "Sort by type" & \(\checkmark:\) : Element properties are sorted by the original order of categories. \\
\hline "Sort by name" & \(\boxed{ } 1\) : Element properties are sorted in alphabetical order. \\
\hline
\end{tabular}

Table 702: "Order"
\begin{tabular}{|l|l|}
\hline "Sort ascending" & \(\square:\) The properties are sorted from A to Z. \\
\hline "Sort descending" & \(\square:\) The properties are sorted from Z to A. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Expert" & \begin{tabular}{l}
\(\square:\) The table includes all properties. The menu command "Filter \\
\(\rightarrow\) Show all categories" is enabled at the same time.
\end{tabular} \\
\hline
\end{tabular}

\section*{Element proper-}
ties display in a
table
\begin{tabular}{|l|l|}
\hline Column "Property" & Element properties of the selected element \\
\hline Column "Value" & The assigned value is applied in the editor view. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
Double-click in the "Value" \\
column
\end{tabular} & \begin{tabular}{l} 
A line editor, drop-down list, or dialog opens for editing the value. \\
\(\ldots\) \\
\hline opens the "Input Assistant" dialog for help, for example when assigning
\end{tabular} \\
\hline Single-click for a selected field & variables or image references.
\end{tabular}
\[
\begin{aligned}
& \text { A style selected in the visualization manager can include single, predefined } \\
& \text { element properties. As a result, these do not appear in this view because a } \\
& \text { fixed value is already assigned to them. They do not have to be configured } \\
& \text { anymore. }
\end{aligned}
\]

Visualizations can be configured with device-specific restrictions that block the availability of element properties.

Device-specific restrictions :
- Elements with restricted availability
- Fonts with restricted availability
- Colors with restricted display
- Image formats with restricted display
- Maximum number of visualization elements
- Maximum number of visualizations below the device

\section*{Object 'Visualization manager'}

Symbol:
The visualization manager manages the configuration settings for all display variants of the visualizations of the current application.
The object is automatically inserted when a visualization object is inserted below the application. On a double-click the configuration dialog opens with several tabs.


If the device employed supports display variants of the visualization, the visualization manager automatically brings along the corresponding objects (CODESYS WebVisu, CODESYS TargetVisu).
If the device employed supports CODESYS TargetVisu, the visualization manager automatically brings along the corresponding object CODESYS TargetVisu.

See also
- \({ }^{\circledR}\) Chapter 6.4.5.18 "Configuring and executing display variants" on page 2973
- \(\Leftrightarrow\) Chapter 6.4.5.21.4.7 "Object 'TargetVisu'" on page 3408
- \({ }^{\bullet} \gg\) Chapter 6.4.5.21.4.8 "Object 'WebVisu'"' on page 3409

Tab 'Visualization manager' 'Settings'

Symbol: 囲
Function: the tab contains settings for all visualizations that are available application-wide.
'General set-
tings'
\begin{tabular}{|c|c|}
\hline "Use unicode strings" & \(\checkmark\) : The visualization codes character strings as Unicode. \\
\hline "Use CurrentVisu variable" & \begin{tabular}{l}
the application knows and uses the global variable VisuElems.CurrentVisu of the type STRING. It contains the name of the currently active visualization at the runtime of the application. The application can read from the variable in order to obtain the name of the currently active visualization. The application can cause a visualization change by a write access. \\
Requirement: the application contains a visualization that calls further visualizations. \\
Example \\
- Variable assignment: VisuElems.CurrentVisu:=strVisuName; \\
- Visualization name assignment: VisuElems.CurrentVisu:='visu1';
\end{tabular} \\
\hline
\end{tabular}
'Style settings'
The "Visualization Style Editor" enables new styles to be generated, checked and installed in the visualization styles repository.
\begin{tabular}{|l|l|}
\hline "Selected style" & \begin{tabular}{l} 
- \\
- Style from the visualization styles repository that every visualization in the \\
application uses, for example "Flat Style". \\
"<None>": The visualization displays its elements without style or according \\
to the internal default. A standard dialog appears instead of a selection list \\
for selection in the element properties "Color" und "Font".
\end{tabular} \\
\hline ⿴ & \begin{tabular}{l} 
Opens a selection list with styles that are installed in the visualization styles \\
repository.
\end{tabular} \\
\hline "Display all versions (for \\
experts only)" & \begin{tabular}{l} 
all The selection list contains only the latest version of each selected style and \\
displayed. \\
a: The selection list contains all versions of all installed styles.
\end{tabular} \\
\hline Button 3 & \begin{tabular}{l} 
Opens a selection list with commands for the use of the "Visualization Styles \\
Editor".
\end{tabular} \\
\hline "Open Style Editor" & The "Visualization Styles Editor" opens. \\
\hline "Create and edit derived style" & \begin{tabular}{l} 
The "Visualization Styles Editor" opens with the dialog "Create a new \\
visualization style". The dialog contains the settings for the first configuration \\
step. \\
Requirement: a style is selected in "Selected style".
\end{tabular} \\
\hline "Copy and edit the selected \\
style" & \begin{tabular}{l} 
The "Visualization Style Editor" opens with the dialog "Open existing style as a \\
copy". The dialog contains the settings for the first configuration step. \\
Requirement: a style is selected in "Selected style".
\end{tabular} \\
\hline "Preview" & The elements displayed represent the style specified in "Selected style". \\
\hline
\end{tabular}

See also
- \& Chapter 6.4.5.19 "Applying Visualization Styles" on page 2979
- Chapter 6.4.5.19.3 "Managing visualization styles in repositories" on page 2985
- \& Chapter 6.4.5.22.1 "Dialog 'Create a New Visualization Style"' on page 3749
- \({ }^{*}\) Chapter 6.4.5.22.2 "Dialog 'Open Existing Style as a Copy"' on page 3749
- \# Chapter 6.4.5.21.3.7 "Dialog 'Options' - 'Visualization Styles'" on page 3382

\section*{'Language set- \\ tings'}

\footnotetext{
'Settings for default text input'

For an element with standard text input, a dialog that supports the input appears at runtime. You can specify which dialog appears.
}
\begin{tabular}{|c|c|}
\hline "Numpad" & Dialog that calls the visualization if a user activates the input field for a number at runtime. The dialog represents a numeric keypad. Default: "VisuDialogs.Numpad" \\
\hline "Keypad" & Dialog that calls the visualization if a user activates the input field for a text at runtime. The dialog represents a keyboard. Preset: "VisuDialogs.Keypad" \\
\hline "Use text input with limits" & \begin{tabular}{l}
Requirement: CODESYS TargetVisu or CODESYS WebVisu are configured as display variants and the "standard text input" is "keyboard". The visualization then supports input via keyboard at runtime. The input thus generally takes place via an input field.
Instead of the input field you can call a dialog that displays the value range for inputs with a limited value range. \\
Default:"VisuDialogs. TextinputWithLimits". This dialog displays the value range and doesn't accept any value outside these limits.
\end{tabular} \\
\hline \multicolumn{2}{|r|}{\begin{tabular}{l}
See also \\
- \({ }^{4}\) Chapter 6.4.5.21.4.7 "Object 'TargetVisu"' on page 3408 \\
- „y Chapter 6.4.5.21.4.8 "Object 'WebVisu'" on page 3409
\end{tabular}} \\
\hline
\end{tabular}

\section*{'Settings for user management dialogs'}

You can configure your visualization with a user management. To do this, configure an input to an element that causes a user management dialog to appear. The VisuUserManagement library contains ready-to-use dialog visualizations for this purpose. The library is located in the installation directory, for example in \(\mathrm{C}: \backslash\) Program Files (x86) \3S CODESYS \(\backslash C O D E S Y S \backslash P r o j e c t s \backslash V i s u \backslash D i a l o g s \backslash V i s u U s e r M g m t D i a l o g s . l i b r a r y . ~\)

You can also use other visualizations as user management dialogs. To do that you have to change the defaults here.
\begin{tabular}{|l|l|}
\hline "Login dialog" & \begin{tabular}{l} 
User management dialog that enables logging in; typically a request to enter a \\
username and a password. It appears upon an input event on an element that \\
executes as a consequential action "User management", action "Login". \\
Preset: VisuUserManagement. VUM_Login
\end{tabular} \\
\hline "Change password dialog" & \begin{tabular}{l} 
User management dialog that enables a password to be changed; typically \\
a request to enter the current password and a new one. It appears upon \\
an input event on an element that executes as a consequential action "User \\
management", action "Change user password". \\
Preset: VisuUserManagement. ChangePassword
\end{tabular} \\
\hline "Change configuration dialog" & \begin{tabular}{l} 
User management dialog that enables a configuration change of the user man- \\
agement, i.e. typically a display of the current user configuration and a possibility \\
to change it. It appears upon an input event on an element that executes as a \\
consequential action "User management", action "Open user configuration". \\
Preset: VisuUserManagement. vum_UserManagement
\end{tabular} \\
\hline
\end{tabular}

See also
- * \(\boldsymbol{y}^{2}\) Chapter 6.4.5.7 "Setting Up User Management" on page 2899
- \(\Leftrightarrow\) Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration'" on page 3370
'Additional set-
tings'
\begin{tabular}{|l|l|}
\hline "Activate multitouch handling" & \begin{tabular}{l} 
V: At runtime the visualization expects user inputs via gestures and touch \\
events. \\
Elements concerned
\end{tabular} \\
& \begin{tabular}{ll} 
- Elements with input configuration \\
- Element of the type "Frame" \\
- Component of the type "Tab control"
\end{tabular} \\
\hline "Activate semi-transparent & \begin{tabular}{l} 
目: The visualization draws the elements in a semi-transparent color. \\
drawing"
\end{tabular} \\
& \begin{tabular}{l} 
To do this you can additionally specify a graduation value for the transparency \\
when defining a color. The transparency is defined in the "Transparency" prop- \\
erty. \\
The leading byte is evaluated in color variables.
\end{tabular} \\
& \begin{tabular}{l} 
Preset: Activated. Requirement: you create a new visualization and the display \\
variants can paint semi-transparently.
\end{tabular} \\
\hline "Activate standard keyboard & - [Tab] \\
handling" & - [Shift] + [Tab] \\
& - [Input] \\
& - [Up arrow] \\
& - [Down arrow] \\
& - [Right arrow] \\
- [Left arrow]
\end{tabular}

See also
- 父 Chapter 6.4.5.5.4 "Assigning a color" on page 2876
- \({ }^{*}\), Chapter 6.4.5.21.1 "Keyboard Shortcuts for Default Keyboard Action" on page 3338

\section*{'Extended set-}
tings'
Table 703: "Memory settings"
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
"Size of memory for \\
visualization"
\end{tabular} & Memory size in bytes allocated by the visualization at runtime. Preset: "400000" \\
\hline \begin{tabular}{l} 
"Size of the paintbuffer (per \\
client)"
\end{tabular} & \begin{tabular}{l} 
Memory size in bytes allocated by the visualization per display variant and used \\
for painting actions. Preset: "50000"
\end{tabular} \\
\hline
\end{tabular}

Table 704: "File transfer mode"
\begin{tabular}{|l|l|}
\hline "Transfer visualization files to \\
the PLC"
\end{tabular} \begin{tabular}{l} 
O: When downloading the application from the visualization directories to the \\
controller, CODESYS copies image files and text list files that the visualization \\
references. A CODESYS TargetVisu needs the files on the PLC and similarly the \\
dialogs that support a numerical input or a keyboard input.
\end{tabular}\(|\)\begin{tabular}{ll} 
Use local visualization files & \begin{tabular}{l} 
O: The visualization uses image files and text list files from local directories \\
Note: In order to achieve that the visualization can access the files stored locally, \\
it is necessary thatthat the file paths are relative. The paths are given in dialog \\
box "Project \(\rightarrow\) Project settings" in tab "Visualization". \\
Note: It is also necessary that the link type of a image is "Link to file". The link \\
type is specified in the image pool.
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{\wedge} \boldsymbol{y}\) Chapter 6.4.5.18.4 "Configure File Transfer Mode" on page 2978
- \(\quad\) Chapter 6.4.5.21.3.13 "Dialog 'Project Settings' - 'Visualization'" on page 3387
- \(\boldsymbol{y}^{2}\) Chapter 6.4.5.21.3.9 "Dialog Box 'Options' - 'Visualization"" on page 3384
- \({ }^{*} \boldsymbol{y}\) Chapter 6.4.1.21.2.15 "Object 'Image Pool"' on page 2468


Table 705: "Client settings"
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
"Maximum number of \\
visualization clients"
\end{tabular} & \begin{tabular}{l} 
Limits the number of visualization clients that are executed at the same time. \\
If you configure the elements so that they vary depending on the display var- \\
iant, then you have to limit the number of display variants. A visualization is \\
given an ID at runtime that identifies the display variant and then processes \\
data accordingly. CODESYS can the query the ID using the system variable \\
CURRENTCLIENTID and thus obtains the information as to which of the running \\
variants is concerned. \\
Example: arr [CURRENTCLIENTID] . dwColor \\
Requirement: VisuGlobalClientManager library is integrated in the project. \\
\\
Tip: You can find in the CODESYS store. example "Global Client Manager"
\end{tabular} \\
\hline
\end{tabular}
"Transfer both svg images and This option is visible only if both a WebVisu and a TargetVisu exist. It concerns converted images" images in svg format only.
The option is available if the device description for the controller of the TargetVisu does not support the svg (full) format.
\(\square\) : The images are transmitted in the png or bmp formats (for TargetVisu) and additionally in svg format (for WebVisu).

Not all settings are available with an integrated CODESYS visualization.

\section*{Tab 'Visualization Manager' - 'Default Hotkeys'}

Symbol:
The tab includes a list of configured keyboard shortcuts that are valid for all visualizations available throughout the application. Therefore, the tab is the central location for defining keyboard shortcuts for all visualizations that are below an application.
Keyboard shortcuts of the default keyboard action are not listed here.
The tab is similar to the "Keyboard Configuration" tab and provides the same editing options.
See also
- «2"Tab 'Keyboard configuration"' on page 3341

Tab 'Visualization manager' - 'Visualizations'
Symbol: 田

Function: The tab lists all visualizations that are available project-wide and enables an assignment of the visualizations for the loading behaviour, depending on the display variants.

\section*{Tab 'Flags'}
\begin{tabular}{|l|l|}
\hline "Standard behaviour" & \begin{tabular}{l}
\(\square:\) The visualizations of the application and the actually referenced visualiza- \\
tions are automatically loaded to the target system. The activated checkboxes \\
show which one that is. \\
\(\square:\) The loading behaviour is explicitly defined for each visualization. \\
Hint: use the explicit selection if you reference visualizations indirectly via IEC \\
variables.
\end{tabular} \\
\hline "Visualizations" & \begin{tabular}{l} 
The list contains all created visualizations from the device tree and the POU \\
view.
\end{tabular} \\
\hline "Dialogs" & \begin{tabular}{l} 
The list contains all the referenceable visualizations that are available via the \\
libraries of the library management.
\end{tabular} \\
\hline
\end{tabular}

Only those visualizations selected here using checkboxes are loaded.
"Remote target visualization, \(\quad\) The column settings affect the loading behaviour for the display variants "remote target visualization, web target visualization", "target visualization" and "web visualization". visualization,"

\section*{Tab 'Visualization manager' - 'User management'}

\section*{Symbol:}

The "User management" tab is used for creating and configuring the user management for visualizations and their users and groups.
\begin{tabular}{|l|l|}
\hline If a user management has not been configured yet, then the following buttons are available: \\
\hline \begin{tabular}{l} 
"Create empty user \\
management"
\end{tabular} & The user management opens. The "None" group is created. \\
\hline "Create user management with \\
default groups and users" & The user management opens. The following groups and users are created: \\
& \begin{tabular}{l} 
- "Admin" group with "Admin" user \\
- "Service" group with "Service" user \\
- "Operator" group "Operator" user \\
\\
\\
- "None" group
\end{tabular} \\
\hline
\end{tabular}

\section*{Project with multiple visualization user managements}

Table 706: "Choose between local and remote user management"
\begin{tabular}{|l|l|}
\hline Requirement: The project includes several devices with a visualization user management. \\
\hline "Use local user management" & The user management of this visualization manager is used for the visualization. \\
\hline \begin{tabular}{l|l} 
"Use remote user \\
management"
\end{tabular} & \begin{tabular}{l} 
Drop-down list with all devices of the project that have their own visualization \\
user management.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{ll} 
Visualization & Even if there are no display variants of the visualization in the application, it may be required \\
user manage- & that the visualization user management is located on the controller. This is the case, for \\
ment on the & example, when HMIs connect to the controller. \\
controller for a & \\
visualization & \\
without dis- \\
play variant
\end{tabular}

Requirement: The visualization does not have any display variants. This means that the objects "Web visualization", "Target visualization", or "Remote target visualization" are inserted below the visualization manager.

\section*{Tab 'Groups'}
\begin{tabular}{|l|l|}
\hline "Group name" & When you click the node, all users are listed that belong to the group. \\
\hline "Automatic logout" & \(\boxed{\checkmark}\) : The "Logout time" input field is active and editable. \\
\hline "Logout time" & \begin{tabular}{l} 
Input field for integer value \\
Drop-down list for time unit "Min", "Sec", or "Hr"
\end{tabular} \\
\hline \begin{tabular}{l} 
"Permission to change user \\
data"
\end{tabular} & \begin{tabular}{l}
\(\checkmark:\) The group is granted permission to edit user data when the visualization is in \\
online mode.
\end{tabular} \\
\hline "Description" & \begin{tabular}{l} 
The text is visible in the development system only. It is not downloaded to the \\
controller.
\end{tabular} \\
\hline "ID" & Unique ID for each group. Assigned automatically by the system. \\
\hline Add a new group & \begin{tabular}{l} 
In the last row of the table, click in the "Group name" field and specify the name \\
for the new group.
\end{tabular} \\
\hline Delete a group & Select a group and press [Del]. The "None" group cannot be deleted. \\
\hline
\end{tabular}

Table 707: Buttons
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
"Update visualizations / \\
hotkeys"
\end{tabular} & \begin{tabular}{l} 
Opens the "Update visualizations and hotkeys" dialog box. \\
Update, if groups were changed at a time when visualizations or keyboard short- \\
cuts already had restricted permissions.
\end{tabular} \\
\hline "List usage of groups" & \begin{tabular}{l} 
List of visualizations and keyboard shortcuts with restricted permissions. \\
The list is displayed in the "Messages" view.
\end{tabular} \\
\hline \begin{tabular}{l} 
"Export groups for global \\
visualizations"
\end{tabular} & \begin{tabular}{l} 
The defined group names are transmitted to "Tools \(\rightarrow\) Options \\
\(\rightarrow\) Visualization user management". They are then listed in "Use the following \\
user group list for the visualization". The list can be changed there as well.
\end{tabular} \\
\hline \begin{tabular}{l} 
"Delete complete user \\
management"
\end{tabular} & \begin{tabular}{l} 
The user management is deleted and the start view is shown with the following \\
buttons: \\
"Create empty user management" and \\
"Create user management with default groups and users".
\end{tabular} \\
\hline "Export user management" & \begin{tabular}{l} 
The drop-down list opens. \\
\(\bullet\) "Before V3.5 SP6" \\
\(\bullet\) "V3.5 SP6 and later" \\
A standard dialog opens for saving the user management as a CSV file with any \\
name in any directory.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Import user management" & \begin{tabular}{l} 
A standard dialog opens for importing a user management. the user manage- \\
ment must be a CSV file.
\end{tabular} \\
\hline \(\boldsymbol{\uparrow} \downarrow\) & \begin{tabular}{l} 
Moves the selected group one line up/down, thus changing the hierarchy of the \\
group.
\end{tabular} \\
\begin{tabular}{l} 
A group of a higher hierarchy cannot have fewer permissions for an element \\
than a group of a lower hierarchy.
\end{tabular} \\
\hline
\end{tabular}

Table 708: Dialog box "Update visualizations and hotkeys"
\begin{tabular}{|l|l|}
\hline This dialog updates only visualization elements and keyboard shortcuts with configured permissions. \\
\hline "Add new group" & \begin{tabular}{l} 
Drop-down list with all new created groups of this user management. \\
Requirement: A new user group was created.
\end{tabular} \\
\hline "Setting for new group" & \begin{tabular}{l} 
- "new group in visualization / hotkey will get the right like group": Drop-down \\
list with all existing groups of this user management \\
"new group should get the following right" \\
\(-\quad\) "for visualization elements": Drop-down list with the permissions: \\
"Operable", "Only visible", and "lnvisible". \\
- "for hotkeys": Drop-down list with the permissions: "Operable", "Not \\
operable".
\end{tabular} \\
\hline "Delete not existing groups" & \begin{tabular}{l} 
If no affected visualization elements or keyboard shortcuts are found for \\
updating, then this is displayed as a message in the "Messages" view \\
("Visualization" category).
\end{tabular} \\
\hline "Rename groups" "Update" & \begin{tabular}{l} 
Updates the permissions of the affected visualization elements and keyboard \\
shortcuts
\end{tabular} \\
\hline
\end{tabular}

Tab 'User'
\begin{tabular}{|l|l|}
\hline "Login name" & \begin{tabular}{l} 
Name for the user to log in to the visualization at runtime. \\
This name is unique.
\end{tabular} \\
\hline "Full name" & This name may exist more than one time in the user management. \\
\hline "Password" & \begin{tabular}{l} 
Encrypted by CODESYS. By default, the "Login name" is displayed here. \\
If you click the "Password" field of a selected line, then the "Change password" \\
dialog box opens.
\end{tabular} \\
\hline "User group" & \begin{tabular}{l} 
Group(s) that the user belongs to. \\
Clicking the "User group" field of a selected user opens the dialog box "User \\
groups the user belongs to". \\
- "Groups" \\
- "Assigned":
\end{tabular} \\
\hline "V: The user is assigned to this group.
\end{tabular}

Table 709: Buttons
\begin{tabular}{|l|l|}
\hline "Upload user from device" & \begin{tabular}{l} 
The data of the user management is uploaded from the controller. \\
If user data is already configured. then it is overwritten.
\end{tabular} \\
\hline "Download user to device" & \begin{tabular}{l} 
The data of the user management is downloaded to the controller. \\
The existing user management on the controller is overwritten.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Export user management" & The drop-down list opens. \\
& \begin{tabular}{l}
\(\bullet \quad\) "Before V3.5 SP6" \\
•V3.5 SP6 and later" \\
A standard dialog opens for saving the user management as a CSV file with any \\
name in any directory.
\end{tabular} \\
\hline "Import user management" & \begin{tabular}{l} 
A standard dialog opens for selecting the user management (in CSV format) \\
from the file system.
\end{tabular} \\
\hline
\end{tabular}

Tab 'Settings'
\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|l|}{"Settings for download of user data"} \\
\hline "Download user data on every login" & The data of the user management is downloaded to the controller at login. Existing data is overwritten. \\
\hline "Never download user data on login" & The data of the user management is never downloaded to the controller, even if it changes. \\
\hline "Allow decision on every download" & A "Warning" dialog box opens for you to accept or refuse the download. \\
\hline \multicolumn{2}{|l|}{"Access rights for elements"} \\
\hline "Use group hierarchy" & \begin{tabular}{l}
The permissions can be granted to the group hierarchy of the "Groups" tab only. \\
The group in the first line of the "Group" list is the highest in the hierarchy. A group of a higher hierarchy cannot have fewer permissions for an element than a group of a lower hierarchy.
\end{tabular} \\
\hline \multicolumn{2}{|l|}{"Logout behavior"} \\
\hline "Change to start visualization at logout" & Switches at logout to the visualization that is configured as the "Start visualization" in the respective display variant. \\
\hline
\end{tabular}

CSV file with the The data for user management is saved to a CSV file in the following format:
data for user management
- User groups: ID;group name; automatic logoff TRUE/FALSE; logoff time; unit logoff time;permission to change user date TRUE/FALSE
- Users: login name; full name;password encrypt TRUE/ FALSE;password;group ID;user deactivated TRUE/FALSE

Use this format when you want to edit data for user management by means of any tool. If you set password encrypt to FALSE, then an unencrypted password can be used. In the example, the unencrypted password Yellow was specified for the user Hugo. If you import the CSV file with the command "Import user management", then the password is encrypted automatically.

\section*{Example}
```

V1.0.0.1
Usergroups:
1;Admin;TRUE;1;Minute;TRUE
3;Operator;FALSE;1;Minute;FALSE
7;Service;FALSE;1;Minute;FALSE
0;None;FALSE;1;Minute;FALSE
4;Early and late shift;FALSE;1;Minute;FALSE
2;Early shift;TRUE;1;Minute;FALSE
6;Late shift;FALSE;1;Minute;FALSE
User:
Service;Service;TRUE;C08298D42A35732CFFB7DF43771B7607;2;FALSE
Operator;Operator;TRUE;3D94AB9540B025B07773DE7037F19837;3;FALSE
John;Blue;TRUE;62ED5DE29E5DD4164A01F3AF1B81EFA0;4;FALSE
Paul;White;TRUE;01E2CBD4AE5442D9EACE33669549A3CC;2;FALSE
Hugo;Green;FALSE;Yellow;6;FALSE

```

\section*{Tab 'Visualization Manager' - 'Font'}

Symbol:
Function: This tab provides settings for adapting the font and font size in the visualization according to the language. The settings apply to all visualizations of the application, including the visualization manager.

Table 710: "Language Specific Font Settings"
\begin{tabular}{|l|l|}
\hline "Language" & \begin{tabular}{l} 
Language used in the project. A column is created for each language. All text \\
lists, including those from integrated libraries, are scanned for this.
\end{tabular} \\
\hline "Font" & Font used by the visualization depending on the language. \\
\hline "Size factor" & \begin{tabular}{l} 
The factor affects the type size of all texts in the visualization. \\
Preset: 1 \\
If the factor is smaller than 1, this leads to a reduction of the type size. If the \\
factor is 1, all texts are displayed unchanged as defined in "Properties".
\end{tabular} \\
\hline Red highlighting of a cell & \begin{tabular}{l} 
The highlighted language is no longer present in the text lists of the project or the \\
libraries. \\
This highlighting is not available in runtime mode.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline Context menu of a selected table row \\
\hline "Delete Language" & \begin{tabular}{l} 
The associated column is removed. This is advisable above all if settings in the \\
column are highlighted in red.
\end{tabular} \\
\hline "Copy Language Settings" & All settings in the column are copied to the clipboard. \\
\hline "Paste Language Settings" & All settings in the column are overwritten with the values from the clipboard. \\
\hline
\end{tabular}

Table 711: "General Font Settings"
\begin{tabular}{|l|l} 
"Automatic decrease of font \\
size" & \begin{tabular}{l} 
V: If the text to be displayed does not fit in the text field in the set format, then \\
the font size is decreased automatically until the text fits completely in the text \\
field. \\
Tip: This prevents a text from being truncated when changing to a language \\
that needs more space. The requirement is that a font is available which has a \\
sufficiently small font.
\end{tabular} \\
\hline
\end{tabular}

See also
- « Chapter 6.4.5.8 "Setting Up Multiple Languages" on page 2906

\section*{Object 'TargetVisu'}

\section*{Symbol: 圆}

Function: The object is used for configuring CODESYS TargetVisu in order to display the visualization directly on the controller of an integrated or connected panel.
Requirement: The CODESYS control runtime environment is equipped with the CODESYS TargetVisu component. The object itself is inserted below the visualization manager.
CODESYS TargetVisu can be executed on different platforms, from embedded controllers to powerful PC-based systems on different operating systems. Therefore, it can be run on Windows, Windows Embedded CE, Linux, QNX, or VxWorks. A ready-made adaptation to the graphics interface of the systems is available on these operating systems. An adaptation is required for embedded controllers or other operating systems. In addition, there are device manufacturers that integrate visualizations into external applications by means of ActiveX controls.
\begin{tabular}{|l|l|}
\hline "Start Visualization" & \begin{tabular}{l} 
Name of the visualization where the start is displayed as CODESYS TargetVisu. \\
Hint: Use input assistance for selecting another visualization.
\end{tabular} \\
\hline "Update rate (ms)" & \begin{tabular}{l} 
Refresh rate (in milliseconds) in the visualization \\
Example: 200
\end{tabular} \\
\hline "Show used visualizations" & \begin{tabular}{l} 
The link opens the "Visualizations" tab in the "Visualization manager" editor. \\
The tab provides information of the visualizations loaded on the display variants.
\end{tabular} \\
\hline \multicolumn{2}{c}{ See also }
\end{tabular}
- \({ }^{\circ}\) ) Chapter 6.4.5.21.4.4 "Tab 'Visualization manager' - 'Visualizations'" on page 3402

Table 712: "Scaling Options"
\begin{tabular}{|c|c|}
\hline "Fixed" & O: Fixed size of the visualization (original size). \\
\hline "Isotropic": & The size of the visualization is adapted to the dimensions of the display device, retaining the proportions of the visualization. \\
\hline "Anisotropic": & The size of the visualization is adjusted to the size of the display device, for example a screen. \\
\hline "Use scaling options for dialogs" & The dialogs, also for keypad and numpad, are scaled like the visualization (drawn with the same scaling factor). This is an advantage when a dialog was created to match the visualization because then they are scaled together. \\
\hline "Use automatically detected client size" & \(\checkmark\) : The visualization fills the screen of the display device completely. \\
\hline "Use specified client size" & The values in "Client height" and "Client width" are used for the size of the visualization. The visualization fills this screen area only. \\
\hline "Client height" & Height of the visualization (in pixels). \\
\hline "Client width" & Width of the visualization (in pixels). \\
\hline
\end{tabular}

Table 713: "Presentation Options"
\begin{tabular}{|l|l|}
\hline "Antialiased drawing" & \begin{tabular}{l} 
V: Antialiasing is used in the visualization editor for drawing a visualization as a \\
TargetVisu and a TargetVisu variant. \\
Hint: If a horizontal or vertical line is drawn blurry on a specific visualization \\
platform, then this can be corrected by an offset of 0.5px in the direction of the \\
line thickness (see element property "Absolute movement", option "Use REAL \\
values"). Requirement: The platform in use supports using REAL coordinates.
\end{tabular} \\
\hline
\end{tabular}

Table 714: "Default Text Input"
\begin{tabular}{|l|l|}
\hline "Input with" & \\
\hline "Touchscreen" & \begin{tabular}{l} 
Text input on the display variant with touchscreen. The keypad or numpad dialog \\
opens.
\end{tabular} \\
\hline "Keyboard" & \begin{tabular}{l} 
Text input on the display variant with an ordinary keyboard or a virtual keyboard \\
(on Linux for example)
\end{tabular} \\
\hline \begin{tabular}{l} 
Effect: \\
When you configure a user input for default text input, select an input configuration for input action "Write \\
variable", and configure the "Input type" as "Default", then the settings are used here.
\end{tabular} \\
\hline
\end{tabular}

See also
- \& Chapter 6.4.5.6 "Configuring user inputs" on page 2885
- «y "Input action 'Write Variable"" on page 3378

\section*{Object 'WebVisu'}

\section*{Symbol: 6}

Function: The object is used to configure the web-based display variant for remote display of the visualization of the controller in a web browser. This allows for remote access, remote monitoring, as well as service and diagnostics of an application over the Internet.
Requirement: The object is inserted below the Visualization Manager, and the target system has a web server with CODESYS WebVisu support. The web server allows for the communication between the target system and the web browser.
See also
- « Chapter 6.4.5.18.2 "Executing as CODESYS WebVisu" on page 2974
\begin{tabular}{|l|l|}
\hline "Start Visualization" & \begin{tabular}{l} 
Name of the visualization where the start is displayed as CODESYS WebVisu. \\
Hint: Use the Input Assistant to select another visualization.
\end{tabular} \\
\hline "Name of the .htm file" & \begin{tabular}{l} 
Base URL of the web page. The URL is also specified as the address in the web \\
browser. \\
Example: http: / /localhost: \(8080 /\) webvisu.htm \\
\\
\\
\begin{tabular}{l} 
Note: If you use a BeagleBone Black as a visualization device, then you have \\
to note that a BeagleBone Black uses port 9090 for its web server. A valid IP \\
address is as follows: http://192.168.7.2:9090/webvisu.htm
\end{tabular} \\
\hline
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Use as default page" & \begin{tabular}{l}
\(\boxed{V}\) : The page specified in "Name of .htm file" is preset as the default page. \\
Now this page will always open when a user specified in the web browser \\
the IP address and port of the web server that is running on the controller: \\
http://<IP address web server>: <port web server>. \\
Example: http://localhost: 8080 \\
Notice: Even if you have created multiple web visualizations, you can activate \\
this option for exactly one web page only and therefore preset only one page as \\
the default page.
\end{tabular} \\
\hline "Update rate (ms)" & Refresh rate (in milliseconds) in the web browser \\
\hline \begin{tabular}{ll} 
"Default communication buffer \\
size"
\end{tabular} & \begin{tabular}{l} 
Default size for communication buffer (in bytes). Defines the maximum available \\
memory for data transfer between the web client and the web server. \\
Example: 50000
\end{tabular} \\
\hline "Show Used Visualizations" & \begin{tabular}{l} 
The link opens the "Visualizations" tab in the "Visualization Manager" editor. \\
The tab provides information about the visualizations downloaded to the display \\
variants.
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{\mu}\) Chapter 6.4.5.21.4.4 "Tab 'Visualization manager' - 'Visualizations"' on page 3402
- \({ }^{*}\) Chapter 6.4.5.18.2 "Executing as CODESYS WebVisu" on page 2974

Table 715: "Scaling Options"
\begin{tabular}{|l|l|}
\hline "Fixed" & \begin{tabular}{l} 
O: Fixed size of the visualization. The values used are "Client height" and "Client \\
width".
\end{tabular} \\
\hline "Isotropic" & \begin{tabular}{l} 
O: The size of the visualization is adapted to the dimensions of the web browser, \\
retaining the proportions of the visualization.
\end{tabular} \\
\hline "Anisotropic" & O: The size of the visualization is adapted to the web browser. \\
\hline \begin{tabular}{l} 
"Use scaling options for \\
dialogs"
\end{tabular} & \begin{tabular}{l} 
( The dialogs (also for keypad and numpad) are scaled as the visualization \\
(drawn with the same scaling factor). This is an advantage when a dialog was \\
created to match the visualization because then they are scaled together.
\end{tabular} \\
\hline "Client height" & Height of the visualization (in pixels). \\
\hline "Client width" & Width of the visualization (in pixels). \\
\hline
\end{tabular}

Table 716: "Presentation Options"
\begin{tabular}{|l|l}
\hline "Antialiased drawing" & \(\boxed{ }\) : Antialiasing is used when drawing the visualization in the web browser.
\end{tabular}

Table 717: "Input handing options"
\begin{tabular}{|l|l|}
\hline "Standard text input with" & \begin{tabular}{l}
\(\bullet\)\begin{tabular}{c} 
"Touchscreen": Text input on the WebVisu with touchscreen. The keypad or \\
numpad dialog opens. \\
"Keyboard": Text input on the WebVisu with an ordinary keyboard or a virtual \\
keyboard (on Android OS for example)
\end{tabular} \\
Effect: \\
When you configure a user input for default text input, select an input configura- \\
tion for input action "Write Variable", and configure the "Input type" as "Default", \\
then the settings are used here.
\end{tabular} \\
\hline "Treat touch as mouse actions" & \begin{tabular}{l} 
V. On devices with a touchscreen, gestures are treated as mouse actions. This \\
option is required, for example, to operate a slider or scrollbar on a touch device.
\end{tabular} \\
\hline
\end{tabular}

\section*{See also}
- \(\triangleq\) Chapter 6.4.5.6 "Configuring user inputs" on page 2885
- \& "Input action 'Write Variable"' on page 3378

\subsection*{6.4.5.21.5 Visualization Elements}
6.4.5.21.5.1 Visualization Element 'Rectangle', 'Rounded Rectangle', 'El- lipse'. ..... 3413
6.4.5.21.5.2 Visualization Element 'Line' ..... 3425
6.4.5.21.5.3 Visualization Element 'Polygon', 'Polyline', 'Bézier Curve'. ..... 3437
6.4.5.21.5.4 Visualization Element 'Pie' ..... 3450
6.4.5.21.5.5 Visualization Element 'Image' ..... 3463
6.4.5.21.5.6 Visualization Element 'Frame', ..... 3478
6.4.5.21.5.7 Visualization Element 'Label' ..... 3493
6.4.5.21.5.8 Visualization Element 'Combo Box, Array'. ..... 3497
6.4.5.21.5.9 Visualization Element 'Combo Box, Integer'. ..... 3503
6.4.5.21.5.10 Visualization Element 'Tabs'.... ..... 3509
6.4.5.21.5.11 Visualization Element 'Button' ..... 3514
6.4.5.21.5.12 Visualization Element 'Group Box'... ..... 3526
6.4.5.21.5.13 Visualization Element 'Table'. ..... 3531
6.4.5.21.5.14 Visualization Element 'Text Field'. ..... 3538
6.4.5.21.5.15 Visualization Element 'Scroll Bar'. ..... 3550
6.4.5.21.5.16 Visualization Element 'Slider' ..... 3559
6.4.5.21.5.17 Visualization Element 'Spin Box'. ..... 3565
6.4.5.21.5.18 Visualization Element 'Invisible Input' ..... 3572
6.4.5.21.5.19 Visualization Element 'Check Box' ..... 3577
6.4.5.21.5.20 Visualization Element 'Progress Bar' ..... 3582
6.4.5.21.5.21 Visualization Element 'Radio Buttons' ..... 3586
6.4.5.21.5.22 Visualization Element 'Alarm Table' ..... 3591
6.4.5.21.5.23 Visualization Element 'Alarm Banner' ..... 3600
6.4.5.21.5.24 Visualization Element 'Bar Display' ..... 3606
6.4.5.21.5.25 Visualization Element 'Meter 90́.. ..... 3612
6.4.5.21.5.26 Visualization Element 'Meter \(180^{\circ}\) '. ..... 3619
6.4.5.21.5.27 Visualization Element 'Meter' ..... 3626
6.4.5.21.5.28 Visualization Element 'Potentiometer'. ..... 3633
6.4.5.21.5.29 Visualization Element 'Histogram' ..... 3641
6.4.5.21.5.30 Visualization Element 'Image Switcher'. ..... 3646
6.4.5.21.5.31 Visualization Element 'Lamp' ..... 3651
6.4.5.21.5.32 Visualization Element 'Dip Switch', 'Power Switch', 'Push Switch', 'Push Switch LED', 'Rocker Switch'. ..... 3656
6.4.5.21.5.33 Visualization Element 'Rotary Switch' ..... 3660
6.4.5.21.5.34 Visualization Element 'Trace' ..... 3665
6.4.5.21.5.35 Visualization Element 'Trend' ..... 3671
6.4.5.21.5.36 Visualization Element 'Legend' ..... 3679
6.4.5.21.5.37 Visualization Element 'ActiveX' ..... 3683
6.4.5.21.5.38 Visualization Element 'Web Browser'. ..... 3687
6.4.5.21.5.39 Visualization Element 'Busy Symbol, Cube' ..... 3691
6.4.5.21.5.40 Visualization Element 'Busy Symbol, Flower' ..... 3695
6.4.5.21.5.41 Visualization Element 'Text Editor' ..... 3699
6.4.5.21.5.42 Visualization Element 'Path3D' ..... 3704
6.4.5.21.5.43 Visualization Element 'Control Panel' ..... 3707
6.4.5.21.5.44 Visualization Element 'Date Range Picker', ..... 3721
6.4.5.21.5.45 Visualization Element 'Time Range Picker' ..... 3726
6.4.5.21.5.46 Visualization Element 'Date Picker' ..... 3730
6.4.5.21.5.47 Visualization Element 'Analog Clock' ..... 3737

\section*{Visualization Element 'Rectangle', 'Rounded Rectangle', 'Ellipse'}

Symbol:

Category: "Basic"
The "Rectangle", "Rounded Rectangle", and "Ellipse" are the same type of element. They can be converted into another element type by changing the "Element type" property.

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Optional \\
Example: Werkstueck_3 \\
\\
Hint: Assign individual names for elements so that they are found faster in the \\
element list.
\end{tabular} \\
\hline "Type of element" "Rectangle", "Rounded Rectangle"," "Ellipse" \\
\hline
\end{tabular}

Element property 'Position'

The position defines the location and size of the element in the visualization window. This is based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|c|c|}
\hline " \(X\) " & \begin{tabular}{l}
The x-coordinate of the upper left corner of the element Specified in pixels \\
Example: 10
\end{tabular} \\
\hline "Y" & \begin{tabular}{l}
The y-coordinate of the upper left corner of the element Specified in pixels \\
Example: 10
\end{tabular} \\
\hline "Width" & Specified in pixels Example: 150 \\
\hline "Height" & \begin{tabular}{l}
Specified in pixels \\
Example: 30
\end{tabular} \\
\hline & Tip: You can change the values in " \(X\) ", " \(Y\) ", "Width", and "Height" by dragging the corresponding symbols 龺- to another position in the editor. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Angle" & \begin{tabular}{l} 
Static angle of rotation (in degrees) \\
Example: 35 \\
The element is displayed rotated in the editor. The point of rotation is the center \\
of the element. A positive value rotates clockwise. \\
Tip: You can change the value in the editor by focusing the element to the \\
handle. When the cursor is displayed as a rotating arrow \\
element about its center as a handle.
\end{tabular} \\
& \begin{tabular}{l} 
(1): Handle can rotate the \\
"Abe: If a dynamic angle of rotation is also configured in the property \\
"Absolute movement \(\rightarrow\) Internal rotation", then the static and dynamic angles \\
of rotation are added in runtime mode. The static angle of rotation acts as an \\
offset.
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\boldsymbol{y y}^{4}\) Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Radius setting Visible only when "Rounded Rectangle" is selected in the "Type of element" property.
\begin{tabular}{|l|l|}
\hline "Radius" & \begin{tabular}{l} 
Rounding of the corners. \\
"From style" \\
"Relative to the element size" \\
"Explicit": Allows for specifying a custom value in the "Value" setting.
\end{tabular} \\
\hline "Value" & \begin{tabular}{l} 
Radius of the rounded corners (in pixels) \\
Example: 5 \\
Requirement: "Explicit" is selected in the "Radius" setting.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center' rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline "Y" & \(Y\)-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus}\) ) to other positions in the editor.

\section*{Element property 'Colors'}
\begin{tabular}{|l|l|}
\hline "Normal state" & \begin{tabular}{l} 
The normal state is in effect if the variable in "Color variables \(\boldsymbol{\rightarrow}\) Toggle color" is \\
not defined or it has the value FALSE.
\end{tabular} \\
\hline "Frame color" & Frame and fill color for the corresponding state of the variable. \\
\hline "Fill color" & \begin{tabular}{l} 
Transparency value (0 to 255) for defining the transparency of the selected color. \\
Example: 255: The color is opaque. 0: The color is completely transparent.
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l} 
The alarm state is in effect if the variable in "Color variables \(\rightarrow\) Toggle color" has \\
the value TRUE.
\end{tabular} \\
\hline "Alarm state" & 目: The element is displayed with a gradient of two colors. \\
\hline "Use gradient color" & The "Gradient editor" dialog box opens. \\
\hline "Gradient setting" & \\
\hline
\end{tabular}

See also
- © Chapter 6.4.5.21.3.5 "Dialog 'Gradient Editor'" on page 3369

Element prop- The properties contain fixed values for setting the look of the element. erty 'Appear-
ance'
\begin{tabular}{|c|c|}
\hline "Line width" & \begin{tabular}{l}
Value in pixels \\
Example: 2 \\
Note: The values 0 and 1 both result in a line weight of 1 pixel. If no line should be displayed, then the "Line style" property must be set to the option "Invisible".
\end{tabular} \\
\hline "Fill attributes" & \begin{tabular}{l}
The way in which the element is filled. \\
- "Filled":The element is filled with the color from property "Colors \(\boldsymbol{\rightarrow}\) Fill color". \\
- "Invisible": The fill color is invisible.
\end{tabular} \\
\hline "Line style" & \begin{tabular}{l}
Type of line representation \\
- "Solid" \\
- "Dashes" \\
- "Dots" \\
- "Dash Dot" \\
- "Dash Dot Dot" \\
- "not visible"
\end{tabular} \\
\hline
\end{tabular}

You can assign variables in the "Appearance variables" property for controlling the appearance dynamically. The fixed values here are overwritten.

\section*{See also}
- * "Element property 'Appearance variables"' on page 3476

Element prop- The properties contains character strings for labeling the element. The character string can also erty 'Texts' contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.
CODESYS accepts the specified texts automatically into the "GlobalTextList" text list. Therefore, these texts can be localized.
\begin{tabular}{|l|l|}
\hline "Text" & \begin{tabular}{l} 
Character string (without single straight quotation marks) for the labeling the \\
element. Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter]. \\
Example: Accesses: \%i \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Text".
\end{tabular} \\
\hline "Tooltip" & \begin{tabular}{l} 
Character string (without single straight quotation marks) that is displayed as the \\
tooltip of an element. \\
Example: Number of valid accesses. \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Tooltip".
\end{tabular} \\
\hline
\end{tabular}

See also
- «y "Element property 'Text variables'" on page 3418
- \(\xi^{4}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872
- \({ }^{4}\) Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329

Element prop- The properties contain fixed values for the text properties.
erty 'Text prop-
erties'
\begin{tabular}{|c|c|}
\hline "Horizontal alignment" & Horizontal alignment of the text within the element. \\
\hline "Vertical alignment" & Vertical alignment of the text within the element. \\
\hline "Text format" & \begin{tabular}{l}
Definition for displaying texts that are too long \\
- "Default": The long text is truncated. \\
- "Line break": The text is split into parts. \\
- "Ellipsis": The visible text ends with "..." indicating that it is not complete.
\end{tabular} \\
\hline "Font" & Example: "Default"
: The "Font" dialog box opens.
Drop-down list with style fonts. \\
\hline "Font color" & Example: "Black"
: The "Color" dialog box opens.
: Drop-down list with style colors. \\
\hline "Transparency" & \begin{tabular}{l}
Whole number (value range from 0 to 255). This determines the transparency of the respective color. \\
Example: 255: The color is opaque. \\
0 : The color is completely transparent. \\
Please note: If the color is a style color and already has a transparency value, then this property is write-protected.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline \multicolumn{1}{|l|}{ "Movement" } \\
\hline " \(X "\) & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|}
\hline "Y" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular}} \\
\hline "Rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle1. \\
The midpoint of the element rotates at the "Center" point. This rotation point is shown as the \(\boldsymbol{\oplus}\) symbol. \\
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
\end{tabular} &  \\
\hline "Scaling" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (integer data type). Causes centric stretching. \\
Example: PLC_PRG.iScaling. \\
The reference point is the "Center" property. \\
The value 1 shrinks the element by a factor of 0.001 . The value 1000 returns the element to its original size.
\end{tabular}} \\
\hline "Interior rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle2. \\
In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. \\
The rotation point is shown as the \(\boldsymbol{\oplus}\) symbol. \\
Note: If a static angle of rotation is specified in the property "Position \(\rightarrow\) Angle", then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.
\end{tabular} &  \\
\hline "Use REAL values" & \begin{tabular}{l}
Note: Only available if the device supports the use of REAL coordinates. \\
: The properties of the absolute movement are interpreted as REAL values. The values are not rounded. \\
The option allows for the individual fine-tuning of drawing the element, for example for the visualization of a smoother rotation. \\
Hint: If a horizontal or vertical line is drawn blurry on a specific visualization platform, then this can be corrected by an offset of 0.5 px in the direction of the line thickness.
\end{tabular} & \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

See also

Element property 'Relative movement'

The properties contains variables for moving the element. The reference point is the position of the element ("Position" property). The shape of the element can change.
\begin{tabular}{|c|c|}
\hline "Movement top-left" & \\
\hline "X" & \begin{tabular}{l}
Variable (integer data type). It contains the number (in pixels) that the left edge is moved horizontally. Incrementing the value moves the element to the right. \\
Example: PLC_PRG.iDeltaX
\end{tabular} \\
\hline " \(Y\) " & \begin{tabular}{l}
Variable (integer data type). It contains the number (in pixels) that the top edge is moved vertically. Incrementing the value moves the element to the down. \\
Example: PLC_PRG.iDeltaY
\end{tabular} \\
\hline "Movement bottom-right" & \\
\hline "X" & \begin{tabular}{l}
Variable (integer data type). It contains the number (in pixels) that the right edge is moved horizontally. Incrementing the value moves the element to the right. \\
Example: PLC_PRG.iDeltaWidth
\end{tabular} \\
\hline " \(Y\) " & \begin{tabular}{l}
Variable (integer data type). It contains the number (in pixels) that the bottom edge is moved vertically. Incrementing the value moves the element to the down. \\
Example: PLC_PRG.iDeltaHeight
\end{tabular} \\
\hline
\end{tabular}

See also
- \#, "Element property 'Absolute movement"' on page 3416

Element prop- These properties are variables with contents that replace a format definition. erty 'Text variables'
\begin{tabular}{|l|l|}
\hline "Text variable" & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses
\end{tabular} \\
& \begin{tabular}{l} 
Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Text". \\
Note: If you specify a variable of type enumeration with text list support, then \\
the name of the enumeration data type is added automatically in angle brackets \\
after the variable name. Example: PLC_PRG.enVar <enumeration name>. \\
Then the symbolic value of the enumeration component is printed instead of the \\
numeric value when text is printed. Refer to the help page for the enumerations.
\end{tabular} \\
\hline "Tooltip variable" & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses InTooltip \\
Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Tooltip".
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\Leftrightarrow\) Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329
- « "Element property 'Texts'" on page 3415
- \(\quad \Rightarrow\) Chapter 6.4.1.20.5.18 "Enumerations" on page 2263

Element property 'Dynamic texts'

Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is currently indexed in the variable.
\begin{tabular}{|c|c|}
\hline "Text list" & \begin{tabular}{l}
Variable (string) or name of the text list as a fixed string in single straight quotation marks. \\
Example: 'Errorlist' \\
*: Drop-down list with the dialogs available in the text lists.
\end{tabular} \\
\hline "Text index" & \begin{tabular}{l}
Text list ID. This refers to the desired output text. \\
- As fixed string with the ID in single straight quotation marks. \\
Example: '1' \\
- As a variable (STRING) for dynamically controlling the text output. \\
Example: strTextID \\
Sample assignment: PLC_PRG.strTextID := '1';
\end{tabular} \\
\hline "Tooltip index" & \begin{tabular}{l}
Text list ID. This refers to the desired output text. \\
- As fixed string with the ID in single straight quotation marks. \\
Example: '2' \\
- As a variable (STRING) for dynamically controlling the text output. \\
Example: strToolTipID \\
Sample assignment: PLC_PRG.strToolTipID := '2';
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\quad \stackrel{y}{c}\) Chapter 6.4.1.21.2.28 "Object 'Text List"" on page 2532

\footnotetext{
Element prop- The variables allow for dynamic control of the text display. erty 'Font variables'
}
\begin{tabular}{|c|c|}
\hline "Font name" & \begin{tabular}{l}
Variable (STRING). Includes the font of the text. \\
Example: PLC_PRG.stFontVar := 'Arial'; \\
The selection of fonts corresponds to the default "Font" dialog.
\end{tabular} \\
\hline "Size" & \begin{tabular}{l}
Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name. \\
- <pt>: Points (default) \\
Example: PLC_PRG.iFontHeight <pt> \\
Code: iFontHeight : INT := 12; \\
- <px> : Pixels \\
Example: PLC_PRG.iFontHeight <px> \\
Code: iFontHeight : INT := 19; \\
If you click in the value field, a drop-down list opens on the right for setting the unit. \\
Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property "Text property \(\rightarrow\) Font".
\end{tabular} \\
\hline "Flags" & \begin{tabular}{l}
Variable (DWORD). Contains the flags for displaying fonts. \\
Flags: \\
- 1: Italics \\
- 2: Bold \\
- 4: Underline \\
- 8: Strikethrough \\
Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: PLC_PRG.dwFontType :=6;
\end{tabular} \\
\hline "Character set" & \begin{tabular}{l}
Variable (DWORD). Contains a character set number for the font. \\
The selection of character set numbers corresponds to the "Script" setting of the standard "Font" dialog.
\end{tabular} \\
\hline "Color" & \begin{tabular}{l}
Variable (DWORD). Includes the color of the text. \\
Example: PLC_PRG.dwColorFont:= 16\#FF000000;
\end{tabular} \\
\hline "Flags for text alignment" & \begin{tabular}{l}
Variable (integer data type). Contains the coding for text alignment. \\
Example: PLC_PRG.dwTextAlignment. \\
Coding: \\
- 0 : Top left \\
- 1: Horizontal center \\
- 2: Right \\
- 4: Vertical center \\
- 8: Bottom \\
Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: PLC_PRG.dwFontType := 5;
\end{tabular} \\
\hline
\end{tabular}

Fixed values for displaying texts are set in "Text properties".

\section*{See also}
- ⓨ "Element property 'Text properties'" on page 3416

Element prop- The Element property is used as an interface for project variables to dynamically control colors erty 'Color varia- at runtime.

\section*{bles'}
\begin{tabular}{|c|c|}
\hline "Toggle color" & \begin{tabular}{l}
The property controls the toggled color at runtime. \\
Value assignment: \\
- FALSE: The element is displayed with the color specified in the "Color" property. \\
- TRUE: The element is displayed with the color specified in the "Alarm color" property. \\
Assignment options: \\
- Placeholder for the user input variable \\
- "<toggle/tap variable>" \\
- "<NOT toggle/tap variable>" \\
The color change is not controlled by its own variable, but by a user input variable. \\
Note: Specify a variable for the mouse events "Tap" or "Toggle" in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both "Toggle" and "Tap", then the variable specified in "Tap" is used. \\
Hint: Click the symbol to insert the placeholder "<toggle/tap variable>". When you activate the "Inputconfiguration", "Tap FALSE" property, then the "<NOT toggle/tap variable>" placeholder is displayed. \\
- Instance path of a project variable (BOOL) \\
Example: PLC_PRG.xColorIsToggeled \\
Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.
\end{tabular} \\
\hline \begin{tabular}{l}
"Normal state" \\
"Alarm state"
\end{tabular} & The properties listed below control the color depending on the state. The normal state is in effect if the variable in "Color variables", "Toggle colo"" is not defined or it has the value FALSE. The alarm state is in effect if the variable in "Colorvariables", "Toggle color" has the value TRUE. \\
\hline "Frame color" & \begin{tabular}{l}
Assignment options: \\
- Variable (DWORD) for the frame color Example: PLC_PRG.dwBorderColor \\
- Color literal Example of green and opaque: 16\#FF00FF00
\end{tabular} \\
\hline "Filling color" & \begin{tabular}{l}
Assignment options: \\
- Variable (DWORD) for the fill color Example: PLC_PRG.dwFillColor \\
- Color literal Example of gray and opaque: 16\#FF888888
\end{tabular} \\
\hline
\end{tabular}


See also
- \& Chapter 6.4.5.10.4 "Animating a color display" on page 2914

Element prop- The properties contain IEC variables for controlling the appearance of the element dynamically. erty 'Appearance variables'
\begin{tabular}{|l|l|}
\hline "Line width" & Variable (integer data type). Contains the line weight (in pixels). \\
\hline "Fill attributes" & Variable (DWORD). Controls whether the fill color of the element is visible. \\
& - Variable value = 0: Filled \\
& - Variable value > 0: Invisible; no fill color \\
\hline "Line style" & Variable (DWORD). Controls the line style. \\
& Coding: \\
& - 0 : Solid line \\
& - 1: Dashed line \\
& - \(2:\) Dotted line \\
& - 3: Line type "Dash Dot" \\
& - 3: Line type "Dash Dot Dot" \\
& - 8: Invisible; no line \\
\hline
\end{tabular}

Fixed values can be set in the "Appearance" property. These values can be overwritten by dynamic variables at runtime.

See also
- \& "Element property 'Appearance"' on page 3427

Element prop- The variables control the element behavior dynamically. erty 'State varia-
bles'
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \(=\) FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

\section*{The "Invisible" property is supported by the "Client Animation" functionality.}

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain the configurations for the user input when using the mouse or keyboard. erty 'Input con- A user input defines an event and one or more actions that are executed when an event occurs. figuration'

The "Configure" button opens the "Input Configuration" dialog. There you can create or edit user inputs.
Configured user inputs are listed below the events. They each include the action that is triggered and the setting in short form.

Example: "Execute ST Code": ßPLC_PRG.i_x := 0;
\begin{tabular}{|l|l|}
\hline "OnDialogClosed" & Input event: The user closes the dialog. \\
\hline "OnMouseClick" & \begin{tabular}{l} 
Input event: The user clicks the mouse button completely in the element area. \\
The mouse button is clicked and released.
\end{tabular} \\
\hline "OnMouseDown" & Input event: The user clicks down on the mouse button. \\
\hline "OnMouseEnter" & Input event: The user drags the mouse pointer to the element. \\
\hline "OnMouseLeave" & Input event: The user drags the mouse pointer away from the element. \\
\hline
\end{tabular}
\(\left.\begin{array}{|l|l|}\hline \text { "OnMouseMove" } & \text { Input event: The user moves the mouse pointer over the element area. } \\ \hline \text { "OnMouseUp" } & \begin{array}{l}\text { Input events: } \\ \text { - } \\ \text { The user releases the mouse button within the element area. It is irrelevant } \\ \text { whether the user has previously pressed the mouse button inside or outside } \\ \text { the element area. }\end{array} \\ \text { - The user presses the mouse button within the element area, leaves the } \\ \text { element area, and then releases the mouse button. } \\ \text { Note: This CODESYS-specific triggering behavior guarantees that actions for } \\ \text { key elements are completed. A key element starts an action for "OnMouseDown" } \\ \text { and ends the action for "OnMouseUp". } \\ \text { Example: A visualization user presses the mouse button within the element area } \\ \text { of the key element and then moves the cursor position so that it lies outside the } \\ \text { element area. The action is ended anyway because "OnMouseUp" is triggered. }\end{array}\right\}\)
\begin{tabular}{|l|l|}
\hline "Tap" & \begin{tabular}{l} 
When a mouse click event occurs, the variable defined in "Variable" is described \\
in the application. The coding depends on the "Tap FALSE" and "Tap on enter if \\
captured" options.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL) that is set on mouse click event. \\
Example: PLC_PRG . bI sTapped \\
TRUE: A mouse click event exists. It lasts as long as the user presses the mouse \\
button over the element. It ends when the button is released.
\end{tabular} \\
& \begin{tabular}{l} 
FALSE: A mouse click event does not exist. \\
Requirement: The "Tap FALSE" option is not activated.
\end{tabular} \\
\hline "Tap FALSE" & \begin{tabular}{l} 
V: The mouse click event leads to a complementary value in "Variable". \\
TRUE: A mouse click event does not exist.
\end{tabular} \\
\hline "Tap on enter if captured" & \begin{tabular}{l} 
FALSE: While the mouse click event exists.
\end{tabular} \\
\hline \begin{tabular}{l} 
pointer is dragged within the element area or not while the mouse button is \\
pressed. \\
TRUE: While the mouse click event exists and the mouse pointer is moved over \\
the element area. \\
FALSE: A mouse click event does not exist. Or the user moves the mouse \\
pointer outside of the element area while the mouse button is pressed. \\
The value is TRUE again as soon as the user moves the pointer back to the \\
element area. The mouse is then captured.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Toggle" & \begin{tabular}{l} 
With the onset of a mouse click event, the variable is set; when the mouse click \\
event is completed, the variable is reset.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL). Its value toggled when the mouse click event is ended. This is \\
when the user releases the mouse button while the mouse pointer is over the \\
element area. \\
If the user releases the mouse button while the mouse pointer is outside of the \\
element area, then the mouse click event is not ended and the value is not \\
toggled. \\
Hint: The user can cancel a started toggle input by dragging the mouse pointer \\
out of the element area.
\end{tabular} \\
\hline "Toggle on up if captured" & \begin{tabular}{l} 
V: The value toggles regardless of where the mouse pointer is when the mouse \\
button is released. The mouse is then captured.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Hotkey" & \begin{tabular}{l}
Keyboard shortcut on the element for triggering specific input actions. \\
When the keyboard shortcut event occurs, the input actions in the "Events" property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.
\end{tabular} \\
\hline "Key" & \begin{tabular}{l}
Key pressed for input action. \\
Example: [T] \\
Note: The following properties appear when a key is selected.
\end{tabular} \\
\hline "Events" & \begin{tabular}{l}
- "None" \\
- "Mouse down": Pressing the key triggers the input actions that are configured in the "OnMouseDown" property. \\
- "Mouse up": Releasing the key triggers the input actions that are configured in the "OnMouseUp" property. \\
- "Mouse down/up": Pressing and releasing the key triggers the input actions that are configured in the "OnMouseDown" property and the "OnMouseUp" property.
\end{tabular} \\
\hline "Shift" & Combination with the Shift key Example: [Shift]+[T]. \\
\hline "Control" & Combination with the Ctrl key Example: [Ctrl]+[T]. \\
\hline "Alt" & Combination with the Alt key Example: \([A I t]+[T]\). \\
\hline
\end{tabular}

All keyboard shortcuts and their actions that are configured in the visualization are listed on the "Keyboard Configuration" tab.

\section*{See also}
- ⓨ Chapter 6.4.5.21.2.2 "Command 'Keyboard Configuration'" on page 3341
- \(\xi^{2}\) Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration'" on page 3370

Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages:
\end{tabular} \\
& \begin{tabular}{ll} 
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
\(\bullet \quad\) "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\xi^{2}\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

See also
- \(\boldsymbol{y y}^{4}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Line'}

Symbol:

\section*{Category: "Basic"}

The element draws a simple line.

\section*{Element proper- \\ ties}
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Optional. \\
Example: Separator_Header_ \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list.
\end{tabular} \\
\hline "Type of element" & "Line" \\
\hline
\end{tabular}

Element prop- The following properties define the position and length of the element in the visualization erty 'Position' window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|c|c|}
\hline "Dots" & \begin{tabular}{l}
"[0]": Coordinates of the starting point \\
"[1]": Coordinate of the end point \\
You can also change the values by dragging the box symbols (마-) to other positions in the editor.
\end{tabular} \\
\hline "Angle" & \begin{tabular}{l}
Static angle of rotation (in degrees). \\
Example: 35 \\
The element is displayed rotated in the editor. The point of rotation is the center of the element. A positive value rotates clockwise. \\
Tip: You can change the value in the editor by focusing the element to the handle. When the cursor is displayed as a rotating arrow \(\cdot\), you can rotate the element about its center as a handle. \\
Example: \\
(1): Handle \\
Note: If a dynamic angle of rotation is also configured in the property "Absolute movement \(\rightarrow\) Internal rotation", then the static and dynamic angles of rotation are added in runtime mode. The static angle of rotation acts as an offset.
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\xi^{4}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center'
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline " \(Y "\) & \(Y\)-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols (ఉ) to other positions in the editor.

Element prop- The properties contain fixed values for setting colors. erty 'Colors'
\begin{tabular}{|l|l|}
\hline "Color" & \begin{tabular}{l} 
Color of the line in normal state. \\
Please note that the normal state is in effect if the expression in the \\
"Color variables \(\rightarrow\) Toggle color" property is not defined or it has the value \\
FALSE.
\end{tabular} \\
\hline "Alarm color" & \begin{tabular}{l} 
Color of the line in alarm state. \\
Please note that the alarm state is in effect if the expression in the \\
"Color variables \(\rightarrow\) Toggle color" property has the value TRUE.
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l} 
Value (0 to 255) for defining the transparency of the selected color. \\
Example \(255:\) The color is opaque. \(0:\) The color is completely transparent.
\end{tabular} \\
\hline
\end{tabular}

See also
- 乡y Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

Element prop- The properties contain fixed values for setting the look of the element.
erty 'Appear-
ance'

Value in pixels
Example: 2
Note: The values 0 and 1 both result in a line weight of one pixel. If no line should be displayed, then the "Line style" property must be set to the option "Invisible".
Type of line representation
- "Solid"
- "Dashes"
- "Dots"
- "Dash Dot"
- "Dash Dot Dot"
- "not visible"
\begin{tabular}{|c|c|}
\hline "Line width" & \begin{tabular}{l}
Value in pixels \\
Example: 2 \\
Note: The values 0 and 1 both result in a line weight of one pixel. If no line should be displayed, then the "Line style" property must be set to the option "Invisible".
\end{tabular} \\
\hline "Line style" & \begin{tabular}{l}
Type of line representation \\
- "Solid" \\
- "Dashes" \\
- "Dots" \\
- "Dash Dot" \\
- "Dash Dot Dot" \\
- "not visible"
\end{tabular} \\
\hline
\end{tabular}

You can assign variables in the "Appearance variables" property for controlling the appearance dynamically. The fixed values are defined here.

\section*{See also}
- ③ "Element property 'Appearance variables'" on page 3476

Element prop- The properties contains character strings for labeling the element. The character string can also erty 'Texts' contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.
CODESYS accepts the specified texts automatically into the "GlobalTextList" text list. Therefore, these texts can be localized.
\begin{tabular}{|l|l|}
\hline "Text" & \begin{tabular}{l} 
Character string (without single straight quotation marks) for the labeling the \\
element. Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter]. \\
Example: Accesses : \%i \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Text".
\end{tabular} \\
\hline "Tooltip" & \begin{tabular}{l} 
Character string (without single straight quotation marks) that is displayed as the \\
tooltip of an element. \\
Example: Number of valid accesses. \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Tooltip".
\end{tabular} \\
\hline
\end{tabular}

See also
- «̌ "Element property 'Text variables'" on page 3430
- \(\xi^{*}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872
- \({ }^{\text {• }}\) Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329

Element prop- The properties contain fixed values for the text properties.
erty 'Text prop-
erties'
\begin{tabular}{|c|c|}
\hline "Horizontal alignment" & Horizontal alignment of the text within the element. \\
\hline "Vertical alignment" & Vertical alignment of the text within the element. \\
\hline "Text format" & \begin{tabular}{l}
Definition for displaying texts that are too long \\
- "Default": The long text is truncated. \\
- "Line break": The text is split into parts. \\
- "Ellipsis": The visible text ends with "..." indicating that it is not complete.
\end{tabular} \\
\hline "Font" & \begin{tabular}{l}
Example: "Default"
: The "Font" dialog box opens. \\
*: Drop-down list with style fonts.
\end{tabular} \\
\hline "Font color" & \begin{tabular}{l}
Example: "Black"
: The "Color" dialog box opens. \\
*: Drop-down list with style colors.
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l}
Whole number (value range from 0 to 255). This determines the transparency of the respective color. \\
Example: 255: The color is opaque. \\
0 : The color is completely transparent. \\
Please note: If the color is a style color and already has a transparency value, then this property is write-protected.
\end{tabular} \\
\hline
\end{tabular}

\section*{Element property 'Absolute movement'}

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|c|c|c|}
\hline \multicolumn{3}{|l|}{"Movement"} \\
\hline "X" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular}} \\
\hline "Y" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the \(Y\) position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular}} \\
\hline "Rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle1. \\
The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol. \\
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
\end{tabular} &  \\
\hline "Scaling" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (integer data type). Causes centric stretching. \\
Example: PLC_PRG.iScaling. \\
The reference point is the "Center" property. \\
The value 1 shrinks the element by a factor of 0.001 . The value 1000 returns the element to its original size.
\end{tabular}} \\
\hline "Interior rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle2. \\
In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. \\
The rotation point is shown as the \(\boldsymbol{\oplus}\) symbol. \\
Note: If a static angle of rotation is specified in the property "Position \(\boldsymbol{\rightarrow}\) Angle", then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.
\end{tabular} &  \\
\hline "Use REAL values" & \begin{tabular}{l}
Note: Only available if the device supports the use of REAL coordinates. \\
The properties of the absolute movement are interpreted as REAL values. The values are not rounded. \\
The option allows for the individual fine-tuning of drawing the element, for example for the visualization of a smoother rotation. \\
Hint: If a horizontal or vertical line is drawn blurry on a specific visualization platform, then this can be corrected by an offset of 0.5 px in the direction of the line thickness.
\end{tabular} & \\
\hline
\end{tabular}
 erty 'Relative movement'
\begin{tabular}{|l|l|}
\hline \multicolumn{1}{|c|}{ "Movement point[0]" } & \begin{tabular}{l} 
Variable (numeric data type). It contains the number (in pixels) that the starting \\
point of the line is moved. \\
Incrementing the \(X\) value moves the element to the right. \\
- "X"
\end{tabular} \\
Incrementing the Y value moves the element to the down.
\end{tabular}

See also
- «4, "Element property 'Absolute movement"' on page 3428

Element prop- These properties are variables with contents that replace a format definition.
erty 'Text varia-
bles'
\begin{tabular}{|c|c|}
\hline "Text variable" & \begin{tabular}{l}
Variable (data type compliant with the format definition). It contains what is printed instead of the format definition. \\
Example: PLC_PRG.iAccesses \\
Note: The format definition is part of the text in the property "Texts \(\boldsymbol{\rightarrow}\) Text". \\
Note: If you specify a variable of type enumeration with text list support, then the name of the enumeration data type is added automatically in angle brackets after the variable name. Example: PLC_PRG.enVar <enumeration name>. Then the symbolic value of the enumeration component is printed instead of the numeric value when text is printed. Refer to the help page for the enumerations.
\end{tabular} \\
\hline "Tooltip variable" & \begin{tabular}{l}
Variable (data type compliant with the format definition). It contains what is printed instead of the format definition. \\
Example: PLC_PRG.iAccessesInTooltip \\
Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Tooltip".
\end{tabular} \\
\hline
\end{tabular}

See also
- MChapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329
- « "Element property 'Texts"' on page 3428
- * Chapter 6.4.1.20.5.18 "Enumerations" on page 2263

Element property 'Dynamic texts'

Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is currently indexed in the variable.
\begin{tabular}{|c|c|}
\hline "Text list" & \begin{tabular}{l}
Variable (string) or name of the text list as a fixed string in single straight quotation marks. \\
Example: 'Errorlist' \\
: Drop-down list with the dialogs available in the text lists.
\end{tabular} \\
\hline "Text index" & \begin{tabular}{l}
Text list ID. This refers to the desired output text. \\
- As fixed string with the ID in single straight quotation marks. \\
Example: '1' \\
- As a variable (STRING) for dynamically controlling the text output. \\
Example: strTextID \\
Sample assignment: PLC_PRG.strTextID := '1';
\end{tabular} \\
\hline "Tooltip index" & \begin{tabular}{l}
Text list ID. This refers to the desired output text. \\
- As fixed string with the ID in single straight quotation marks. \\
Example: '2' \\
- As a variable (STRING) for dynamically controlling the text output. \\
Example: strToolTipID \\
Sample assignment: PLC_PRG.strToolTipID := '2';
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{\text {• }}\), Chapter 6.4.1.21.2.28 "Object 'Text List"' on page 2532

Element prop- The variables allow for dynamic control of the text display. erty 'Font variables'
\begin{tabular}{|c|c|}
\hline "Font name" & \begin{tabular}{l}
Variable (STRING). Includes the font of the text. \\
Example: PLC_PRG.stFontVar := 'Arial'; \\
The selection of fonts corresponds to the default "Font" dialog.
\end{tabular} \\
\hline "Size" & \begin{tabular}{l}
Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name. \\
- <pt>: Points (default) \\
Example: PLC_PRG.iFontHeight <pt> \\
Code: iFontHeight : INT := 12; \\
- <px> : Pixels \\
Example: PLC_PRG.iFontHeight <px> \\
Code: iFontHeight : INT := 19; \\
pt \\
pt \\
px \\
If you click in the value field, a drop-down list opens on the right for setting the unit. \\
Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property "Text property \(\rightarrow\) Font".
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Flags" & \begin{tabular}{l}
Variable (DWORD). Contains the flags for displaying fonts. \\
Flags: \\
- 1: Italics \\
- 2: Bold \\
- 4: Underline \\
- 8: Strikethrough \\
Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: PLC_PRG.dwFontType := 6;
\end{tabular} \\
\hline "Character set" & \begin{tabular}{l}
Variable (DWORD). Contains a character set number for the font. \\
The selection of character set numbers corresponds to the "Script" setting of the standard "Font" dialog.
\end{tabular} \\
\hline "Color" & \begin{tabular}{l}
Variable (DWORD). Includes the color of the text. \\
Example: PLC_PRG.dwColorFont:= 16\#FF000000;
\end{tabular} \\
\hline "Flags for text alignment" & \begin{tabular}{l}
Variable (integer data type). Contains the coding for text alignment. \\
Example: PLC_PRG.dwTextAlignment. \\
Coding: \\
- 0 : Top left \\
- 1: Horizontal center \\
- 2: Right \\
- 4: Vertical center \\
- 8: Bottom \\
Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: PLC_PRG.dwFontType := 5;
\end{tabular} \\
\hline
\end{tabular}

Fixed values for displaying texts are set in "Text properties".

Element prop- The Element property is used as an interface for project variables to dynamically control colors erty 'Color varia- at runtime. bles'
\begin{tabular}{|c|c|}
\hline "Toggle color" & \begin{tabular}{l}
The property controls the toggled color at runtime. \\
Value assignment: \\
- FALSE: The element is displayed with the color specified in the "Color" property. \\
- TRUE: The element is displayed with the color specified in the "Alarm color" property. \\
Assigning the property: \\
- Placeholder for the user input variable \\
- "<toggle/tap variable>" \\
- "<NOT toggle/tap variable>" \\
The color change is not controlled by its own variable, but by a user input variable. \\
Note: Specify a variable for the mouse events "Tap" or "Toggle" in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both "Toggle" and "Tap", then the variable specified in "Tap" is used. \\
Hint: Click the symbol to insert the placeholder "<toggle/tap variable>". When you activate the "Inputconfiguration", "Tap FALSE" property, then the "<NOT toggle/tap variable>" placeholder is displayed. \\
- Instance path of a project variable (BOOL) \\
Example: PLC_PRG.xColorIsToggeled \\
Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.
\end{tabular} \\
\hline "Color" & \begin{tabular}{l}
- Variable (DWORD) for the color \\
Example: PLC_PRG.dwColor \\
- Color literal \\
Example of gray and opaque: 16\#FF888888 \\
Please note that the normal state is in effect if the expression in the "Colorvariables \(\rightarrow\) Toggle color" property is not defined or it has the value FALSE.
\end{tabular} \\
\hline "Alarm color" & \begin{tabular}{l}
Color variable in the alarm state \\
- Variable (DWORD) for the alarm color \\
Example: PLC_PRG.dwAlarmColor \\
- Color literal \\
Example of red and opaque: 16\#FFFF0000 \\
Please note that the alarm state is in effect if the expression in the "Colorvariables \(\rightarrow\) Toggle color" property has the value TRUE.
\end{tabular} \\
\hline
\end{tabular}

The transparency part of the color value is evaluated only if the "Activate semitransparent drawing" option of the visualization manager is selected.

Select the "Advanced" option in the toolbar of the properties view. Then all element properties are visible.

See also
- \& Chapter 6.4.5.10.4 "Animating a color display" on page 2914
- Chapter 6.4.5.21.4.2 "Object 'Visualization manager'" on page 3398

\section*{Element prop- The variables control the element behavior dynamically. erty 'State variables'}
\begin{tabular}{|l|l|}
\hline "Invisible" & Variable (BOOL). Toggles the visibility of the element. \\
& \begin{tabular}{l} 
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \(=\) FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

Element prop- Dynamic definition of the weight of a line element using a variable. erty 'Line width variable'
\begin{tabular}{|l|l|}
\hline "Integer value" & \begin{tabular}{l} 
Variable (integer data type). Defines the line weight of the element (in pixels)., \\
This overwrites the fixed value that is defined in "Appearance \(\rightarrow\) Line weight". \\
Note: The value 0 codes the same as 1 and sets the line weight to one pixel.
\end{tabular} \\
\hline
\end{tabular}

\section*{Element property 'Line style variable'}
\begin{tabular}{|c|c|}
\hline "Integer value" & \begin{tabular}{l}
Variable (integer data type). Defines the appearance of the line at runtime. \\
- 1: Solid \\
- 2: Dashes \\
- 3: Dots \\
- 4: Dash Dot \\
- 5: Dash Dot Dot \\
- 6: Invisible: The line is not drawn.
\end{tabular} \\
\hline
\end{tabular}

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", "X", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain the configurations for the user input when using the mouse or keyboard. erty 'Input con- A user input defines an event and one or more actions that are executed when an event occurs. figuration'
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
The "Configure" button opens the "Input Configuration" dialog. There you can create or edit user inputs. \\
Configured user inputs are listed below the events. They each include the action that is triggered and the setting \\
in short form. \\
Example: "Execute ST Code":
\end{tabular} \\
\hline "Onc__PRG.i_x \(:=0 ;\) \\
\hline "OnDialogClosed" & Input event: The user closes the dialog. \\
\hline "OnMouseDown" & \begin{tabular}{l} 
Input event: The user clicks the mouse button completely in the element area. \\
The mouse button is clicked and released.
\end{tabular} \\
\hline "OnMouseEnter" & Input event: The user clicks down on the mouse button. \\
\hline "OnMouseLeave" & Input event: The user drags the mouse pointer to the element. \\
\hline
\end{tabular}
\(\left.\begin{array}{|l|l|}\hline \text { "OnMouseMove" } & \text { Input event: The user moves the mouse pointer over the element area. } \\ \hline \text { "OnMouseUp" } & \begin{array}{l}\text { Input events: } \\ \text { - } \\ \text { The user releases the mouse button within the element area. It is irrelevant } \\ \text { whether the user has previously pressed the mouse button inside or outside } \\ \text { the element area. }\end{array} \\ \text { - The user presses the mouse button within the element area, leaves the } \\ \text { element area, and then releases the mouse button. } \\ \text { Note: This CODESYS-specific triggering behavior guarantees that actions for } \\ \text { key elements are completed. A key element starts an action for "OnMouseDown" } \\ \text { and ends the action for "OnMouseUp". } \\ \text { Example: A visualization user presses the mouse button within the element area } \\ \text { of the key element and then moves the cursor position so that it lies outside the } \\ \text { element area. The action is ended anyway because "OnMouseUp" is triggered. }\end{array}\right\}\)
\begin{tabular}{|l|l|}
\hline "Tap" & \begin{tabular}{l} 
When a mouse click event occurs, the variable defined in "Variable" is described \\
in the application. The coding depends on the "Tap FALSE" and "Tap on enter if \\
captured" options.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL) that is set on mouse click event. \\
Example: PLC_PRG . bI sTapped \\
TRUE: A mouse click event exists. It lasts as long as the user presses the mouse \\
button over the element. It ends when the button is released.
\end{tabular} \\
& \begin{tabular}{l} 
FALSE: A mouse click event does not exist. \\
Requirement: The "Tap FALSE" option is not activated.
\end{tabular} \\
\hline "Tap FALSE" & \begin{tabular}{l} 
V: The mouse click event leads to a complementary value in "Variable". \\
TRUE: A mouse click event does not exist.
\end{tabular} \\
\hline "Tap on enter if captured" & \begin{tabular}{l} 
FALSE: While the mouse click event exists.
\end{tabular} \\
\hline \begin{tabular}{l} 
pointer is dragged within the element area or not while the mouse button is \\
pressed. \\
TRUE: While the mouse click event exists and the mouse pointer is moved over \\
the element area. \\
FALSE: A mouse click event does not exist. Or the user moves the mouse \\
pointer outside of the element area while the mouse button is pressed. \\
The value is TRUE again as soon as the user moves the pointer back to the \\
element area. The mouse is then captured.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Toggle" & \begin{tabular}{l} 
With the onset of a mouse click event, the variable is set; when the mouse click \\
event is completed, the variable is reset.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL). Its value toggled when the mouse click event is ended. This is \\
when the user releases the mouse button while the mouse pointer is over the \\
element area. \\
If the user releases the mouse button while the mouse pointer is outside of the \\
element area, then the mouse click event is not ended and the value is not \\
toggled. \\
Hint: The user can cancel a started toggle input by dragging the mouse pointer \\
out of the element area.
\end{tabular} \\
\hline "Toggle on up if captured" & \begin{tabular}{l} 
V: The value toggles regardless of where the mouse pointer is when the mouse \\
button is released. The mouse is then captured.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Hotkey" & \begin{tabular}{l}
Keyboard shortcut on the element for triggering specific input actions. \\
When the keyboard shortcut event occurs, the input actions in the "Events" property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.
\end{tabular} \\
\hline "Key" & \begin{tabular}{l}
Key pressed for input action. \\
Example: [T] \\
Note: The following properties appear when a key is selected.
\end{tabular} \\
\hline "Events" & \begin{tabular}{l}
- "None" \\
- "Mouse down": Pressing the key triggers the input actions that are configured in the "OnMouseDown" property. \\
- "Mouse up": Releasing the key triggers the input actions that are configured in the "OnMouseUp" property. \\
- "Mouse down/up": Pressing and releasing the key triggers the input actions that are configured in the "OnMouseDown" property and the "OnMouseUp" property.
\end{tabular} \\
\hline "Shift" & Combination with the Shift key Example: [Shift]+[T]. \\
\hline "Control" & Combination with the Ctrl key Example: [Ctrl]+[T]. \\
\hline "Alt" & Combination with the Alt key Example: \([A t t]+[T]\). \\
\hline
\end{tabular}

All keyboard shortcuts and their actions that are configured in the visualization are listed on the "Keyboard Configuration" tab.

\section*{See also}
- ⓨ Chapter 6.4.5.21.2.2 "Command 'Keyboard Configuration'" on page 3341
- \(\xi^{2}\) Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration'" on page 3370

Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages: \\
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
- "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\(\qquad\)\begin{tabular}{l} 
See also \\
- «3 Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366
\end{tabular}
\end{tabular}

See also
- \({ }^{\text {- }}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Polygon', 'Polyline', 'Bézier Curve'}

Symbol:

Category: "Basic"
The "Polygon", "Polyline", and "Bézier Curve" are the same element type. They can be converted into another type by changing the "Element type" property.

Elements can be dragged to the editor. The element is then drawn with five points: [0] to [4].
Other positions are added as follows: Move the mouse pointer over a corner point; the mouse pointer changes shape. Now if you press and hold [Ctrl] and click the left mouse button, another point is created. You can delete a point by pressing and holding [Shift]+[Ctrl] and click the selected point.

As an alternative, you can select the element in the toolbox area and in the editor click multiple times. At the same time, a connecting line is drawn from one point to the other. End by double-clicking the element or right-clicking it one time.

\section*{Element proper- \\ ties}
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Optional. \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list. \\
Example: Werkstueck_1
\end{tabular} \\
\hline "Type of element" & \begin{tabular}{l} 
- "Polygon" \\
- "Polyline" \\
- "Bézier Curve"
\end{tabular} \\
\hline
\end{tabular}

Element prop- The following properties define the position of the corner points in the visualization window. erty 'Position' These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|c|c|}
\hline "Dots" & \begin{tabular}{l}
[0]..[n]: Coordinates of the corner points \\
Specified in pixels \\
You can also change the values by dragging the box symbols ([世-) to other positions in the editor.
\end{tabular} \\
\hline "Angle" & \begin{tabular}{l}
Static angle of rotation (in degrees). \\
Example: 35 \\
The element is displayed rotated in the editor. The point of rotation is the center of the element. A positive value rotates clockwise. \\
Tip: You can change the value in the editor by focusing the element to the handle. When the cursor is displayed as a rotating arrow \(\mathcal{A}\), you can rotate the element about its center as a handle. \\
(1): Handle \\
Note: If a dynamic angle of rotation is also configured in the property "Absolute movement \(\rightarrow\) Internal rotation", then the static and dynamic angles of rotation are added in runtime mode. The static angle of rotation acts as an offset.
\end{tabular} \\
\hline
\end{tabular}

See also
- 乡y Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center' rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline "Y" & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus}\) ) to other positions in the editor.

\section*{Element prop- \\ erty 'Colors'}
\begin{tabular}{|l|l|}
\hline "Normal state" & \begin{tabular}{l} 
The normal state is in effect if the variable in "Color variables \(\rightarrow\) Toggle color" is \\
not defined or it has the value FALSE.
\end{tabular} \\
\hline "Frame color" & Frame and fill color for the corresponding state of the variable. \\
\hline "Fill color" & "Transparency" \\
\hline Transparency value (0 to 255) for defining the transparency of the selected color. \\
Example: \(255:\) The color is opaque. \(0:\) The color is completely transparent.
\end{tabular}\(|\)\begin{tabular}{ll} 
"Alarm state" & \begin{tabular}{l} 
The alarm state is in effect if the variable in "Color variables \(\rightarrow\) Toggle color" has \\
the value TRUE.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Use gradient color" & \(\boxed{\square}:\) The element is displayed with a gradient of two colors. \\
\hline "Gradient setting" & The "Gradient editor" dialog box opens. \\
\hline
\end{tabular}

See also
- \(y^{\text {® }}\) Chapter 6.4.5.21.3.5 "Dialog 'Gradient Editor"' on page 3369

Element prop- The properties contain fixed values for setting the look of the element. erty 'Appearance'
\begin{tabular}{|c|c|}
\hline "Line width" & \begin{tabular}{l}
Value in pixels \\
Example: 2 \\
Note: The values 0 and 1 both result in a line weight of 1 pixel. If no line should be displayed, then the "Line style" property must be set to the option "Invisible".
\end{tabular} \\
\hline "Fill attributes" & \begin{tabular}{l}
The way in which the element is filled. \\
- "Filled":The element is filled with the color from property "Colors \(\boldsymbol{\rightarrow}\) Fill color". \\
- "Invisible": The fill color is invisible.
\end{tabular} \\
\hline "Line style" & \begin{tabular}{l}
Type of line representation \\
- "Solid" \\
- "Dashes" \\
- "Dots" \\
- "Dash Dot" \\
- "Dash Dot Dot" \\
- "not visible"
\end{tabular} \\
\hline
\end{tabular}

You can assign variables in the "Appearance variables" property for controlling the appearance dynamically. The fixed values here are overwritten.

\section*{See also}
- « "Element property 'Appearance variables'" on page 3476

Element prop- The properties contains character strings for labeling the element. The character string can also erty 'Texts' contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.
CODESYS accepts the specified texts automatically into the "GlobalTextList" text list. Therefore, these texts can be localized.
\begin{tabular}{|l|l|}
\hline "Text" & \begin{tabular}{l} 
Character string (without single straight quotation marks) for the labeling the \\
element. Add a line break by pressing the keyboard shortcut [CtrI] + [Enter]. \\
Example: Accesses: \%i \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Text".
\end{tabular} \\
\hline "Tooltip" & \begin{tabular}{l} 
Character string (without single straight quotation marks) that is displayed as the \\
tooltip of an element. \\
Example: Number of valid accesses. \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Tooltip".
\end{tabular} \\
\hline
\end{tabular}

See also
- «. "Element property 'Text variables"' on page 3443
- \& Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872
- 4 Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329

Element prop- The properties contain fixed values for the text properties. erty 'Text properties'
\begin{tabular}{|c|c|}
\hline "Horizontal alignment" & Horizontal alignment of the text within the element. \\
\hline "Vertical alignment" & Vertical alignment of the text within the element. \\
\hline "Text format" & \begin{tabular}{l}
Definition for displaying texts that are too long \\
- "Default": The long text is truncated. \\
- "Line break": The text is split into parts. \\
- "Ellipsis": The visible text ends with "..." indicating that it is not complete.
\end{tabular} \\
\hline "Font" & \begin{tabular}{l}
Example: "Default"
The "Font" dialog box opens. \\
- Drop-down list with style fonts.
\end{tabular} \\
\hline "Font color" & \begin{tabular}{l}
Example: "Black"
The "Color" dialog box opens. \\
*: Drop-down list with style colors.
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l}
Whole number (value range from 0 to 255). This determines the transparency of the respective color. \\
Example: 255: The color is opaque. \\
0 : The color is completely transparent. \\
Please note: If the color is a style color and already has a transparency value, then this property is write-protected.
\end{tabular} \\
\hline
\end{tabular} erty 'Absolute movement'

Element prop- The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\oplus\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
\hline "Scaling" & \begin{tabular}{l} 
Variable (integer data type). Causes centric stretching. \\
Example: PLC_PRG. iScaling. \\
The reference point is the "Center" property. \\
The value 1 shrinks the element by a factor of 0.001. \\
to its original size.
\end{tabular} \\
\hline \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol. \\
Note: If a static angle of rotation is specified in the \\
property "Position \(\rightarrow\) Angle", then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
"Use REAL values" & \begin{tabular}{l} 
Note: Only available if the device supports the use \\
of REAL coordinates. \\
l: The properties of the absolute movement are \\
interpreted as REAL values. The values are not \\
rounded. \\
The option allows for the individual fine-tuning of \\
drawing the element, for example for the visualiza- \\
tion of a smoother rotation. \\
Hint: If a horizontal or vertical line is drawn blurry \\
on a specific visualization platform, then this can \\
be corrected by an offset of 0.5px in the direction \\
of the line thickness.
\end{tabular} \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

\author{
The properties " \(X\) ", " \(Y\) ", "Rotation", and "Interior rotation" are supported by the "Client Animation" functionality.
}

See also
-

\section*{Element property 'Dynamic points'}
\begin{tabular}{|l|l|}
\hline "Array of points" & \begin{tabular}{l} 
Variable (POINTER TO). Points to an array of the structure \\
VisuElems.VisuStructPoint. The elements ix and iY of \\
VisuStructPoint contain the xy-coordinates of a point The current number of \\
array elements implicitly contains the variable in the property "Number of points". \\
The variable that is assigned to the property "Number of points" contains the \\
number of array elements and therefore the number of corner points. \\
Example: pPoints : POINTER TO ARRAY [0..100] OF \\
VisuElems.VisuStructPoint;
\end{tabular} \\
\hline & \begin{tabular}{l} 
Variable (integer data type): Contains the number of array elements and there- \\
fore the number of corner points for displaying the element. \\
Example: PLC_PRG. iNumberOfPoints : \(=24 ;\) \\
In the example, the element has 24 points. This definition is necessary because
\end{tabular} \\
the individual points are defined by a pointer and this does not allow control over \\
the number of points. \\
Note: In this way, it is possible to adapt the display of the element dynamically by \\
updating the number of corner points.
\end{tabular}

Element prop- These properties are variables with contents that replace a format definition.
erty 'Text varia-
bles'
\begin{tabular}{|l|l|}
\hline "Text variable" & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses
\end{tabular} \\
& \begin{tabular}{l} 
Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Text". \\
Note: If you specify a variable of type enumeration with text list support, then \\
the name of the enumeration data type is added automatically in angle brackets \\
after the variable name. Example: PLC PRG. enVar <enumeration name> \\
Then the symbolic value of the enumeration component is printed instead of the \\
numeric value when text is printed. Refer to the help page for the enumerations.
\end{tabular} \\
\hline & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses InTooltip
\end{tabular} \\
Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Tooltip". \\
\hline
\end{tabular}

See also
- Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329
- «3 "Element property 'Texts"' on page 3440
- « Chapter 6.4.1.20.5.18 "Enumerations" on page 2263

Element prop- Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is erty 'Dynamic texts' currently indexed in the variable.
\begin{tabular}{|c|c|}
\hline "Text list" & \begin{tabular}{l}
Variable (string) or name of the text list as a fixed string in single straight quotation marks. \\
Example: 'Errorlist' \\
: Drop-down list with the dialogs available in the text lists.
\end{tabular} \\
\hline "Text index" & \begin{tabular}{l}
Text list ID. This refers to the desired output text. \\
- As fixed string with the ID in single straight quotation marks. Example: '1' \\
- As a variable (STRING) for dynamically controlling the text output. \\
Example: strTextID \\
Sample assignment: PLC_PRG.strTextID := '1';
\end{tabular} \\
\hline "Tooltip index" & \begin{tabular}{l}
Text list ID. This refers to the desired output text. \\
- As fixed string with the ID in single straight quotation marks. \\
Example: '2' \\
- As a variable (STRING) for dynamically controlling the text output. \\
Example: strToolTipID \\
Sample assignment: PLC_PRG.strToolTipID := '2';
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{\text {• }}\) Chapter 6.4.1.21.2.28 "Object 'Text List"' on page 2532

Element prop- The variables allow for dynamic control of the text display. erty 'Font variables'
\begin{tabular}{|c|c|}
\hline "Font name" & \begin{tabular}{l}
Variable (STRING). Includes the font of the text. \\
Example: PLC_PRG.stFontVar := 'Arial'; \\
The selection of fonts corresponds to the default "Font" dialog.
\end{tabular} \\
\hline "Size" & \begin{tabular}{l}
Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name. \\
- <pt>: Points (default) \\
Example: PLC_PRG.iFontHeight <pt> \\
Code: iFontHeight : INT := 12; \\
- <px> : Pixels \\
Example: PLC_PRG.iFontHeight <px> \\
Code: iFontHeight : INT := 19; \\
If you click in the value field, a drop-down list opens on the right for setting the unit. \\
Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property "Text property \(\rightarrow\) Font".
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Flags" & \begin{tabular}{l}
Variable (DWORD). Contains the flags for displaying fonts. \\
Flags: \\
- 1: Italics \\
- 2: Bold \\
- 4: Underline \\
- 8: Strikethrough \\
Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: PLC_PRG.dwFontType :=6;
\end{tabular} \\
\hline "Character set" & \begin{tabular}{l}
Variable (DWORD). Contains a character set number for the font. \\
The selection of character set numbers corresponds to the "Script" setting of the standard "Font" dialog.
\end{tabular} \\
\hline "Color" & \begin{tabular}{l}
Variable (DWORD). Includes the color of the text. \\
Example: PLC_PRG.dwColorFont:= 16\#FF000000;
\end{tabular} \\
\hline "Flags for text alignment" & \begin{tabular}{l}
Variable (integer data type). Contains the coding for text alignment. \\
Example: PLC_PRG.dwTextAlignment. \\
Coding: \\
- \(0:\) Top left \\
- 1: Horizontal center \\
- 2: Right \\
- 4: Vertical center \\
- 8: Bottom \\
Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: PLC_PRG.dwFontType := 5;
\end{tabular} \\
\hline
\end{tabular}

Fixed values for displaying texts are set in "Text properties".

Element prop- The Element property is used as an interface for project variables to dynamically control colors erty 'Color varia- at runtime. bles'
\begin{tabular}{|c|c|}
\hline "Toggle color" & \begin{tabular}{l}
The property controls the toggled color at runtime. \\
Value assignment: \\
- FALSE: The element is displayed with the color specified in the "Color" property. \\
- TRUE: The element is displayed with the color specified in the "Alarm color" property. \\
Assignment options: \\
- Placeholder for the user input variable \\
- "<toggle/tap variable>" \\
- "<NOT toggle/tap variable>" \\
The color change is not controlled by its own variable, but by a user input variable. \\
Note: Specify a variable for the mouse events "Tap" or "Toggle" in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both "Toggle" and "Tap", then the variable specified in "Tap" is used. \\
Hint: Click the symbol to insert the placeholder "<toggle/tap variable>". When you activate the "Inputconfiguration", "Tap FALSE" property, then the "<NOT toggle/tap variable>" placeholder is displayed. \\
- Instance path of a project variable (BOOL) \\
Example: PLC_PRG.xColorIsToggeled \\
Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.
\end{tabular} \\
\hline \begin{tabular}{l}
"Normal state" \\
"Alarm state"
\end{tabular} & The properties listed below control the color depending on the state. The normal state is in effect if the variable in "Color variables", "Toggle color" is not defined or it has the value FALSE. The alarm state is in effect if the variable in "Colorvariables", "Toggle color" has the value TRUE. \\
\hline "Frame color" & \begin{tabular}{l}
Assignment options: \\
- Variable (DWORD) for the frame color Example: PLC_PRG.dwBorderColor \\
- Color literal Example of green and opaque: 16\#FF00FF00
\end{tabular} \\
\hline "Filling color" & \begin{tabular}{l}
Assignment options: \\
- Variable (DWORD) for the fill color Example: PLC_PRG.dwFillColor \\
- Color literal Example of gray and opaque: \(16 \#\) FF888888
\end{tabular} \\
\hline
\end{tabular}

The transparency part of the color value is evaluated only if the "Activate semitransparent drawing" option of the visualization manager is selected.

Select the "Advanced" option in the toolbar of the properties view. Then all element properties are visible.

\section*{See also}
- \& Chapter 6.4.5.10.4 "Animating a color display" on page 2914

Element prop- The properties contain IEC variables for controlling the appearance of the element dynamically. erty 'Appearance variables'
\begin{tabular}{|l|l|}
\hline "Line width" & Variable (integer data type). Contains the line weight (in pixels). \\
\hline "Fill attributes" & Variable (DWORD). Controls whether the fill color of the element is visible. \\
& - Variable value = 0: Filled \\
& - Variable value > 0: Invisible; no fill color \\
\hline "Line style" & Variable (DWORD). Controls the line style. \\
& Coding: \\
& - \(0:\) Solid line \\
& - 1: Dashed line \\
& - 2: Dotted line \\
& - 3: Line type "Dash Dot" \\
& - 3: Line type "Dash Dot Dot" \\
& - 8: Invisible; no line \\
\hline
\end{tabular}

Fixed values can be set in the "Appearance" property. These values can be overwritten by dynamic variables at runtime.

See also
- ©s "Element property 'Appearance"' on page 3468

Element prop- The variables control the element behavior dynamically. erty 'State varia-
bles'
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
\\
\\
\\
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain the configurations for the user input when using the mouse or keyboard. erty 'Input con- A user input defines an event and one or more actions that are executed when an event occurs. figuration'

The "Configure" button opens the "Input Configuration" dialog. There you can create or edit user inputs.
Configured user inputs are listed below the events. They each include the action that is triggered and the setting in short form.

Example: "Execute ST Code": ßPLC_PRG.i_x := 0;
\begin{tabular}{|l|l|}
\hline "OnDialogClosed" & Input event: The user closes the dialog. \\
\hline "OnMouseClick" & \begin{tabular}{l} 
Input event: The user clicks the mouse button completely in the element area. \\
The mouse button is clicked and released.
\end{tabular} \\
\hline "OnMouseDown" & Input event: The user clicks down on the mouse button. \\
\hline "OnMouseEnter" & Input event: The user drags the mouse pointer to the element. \\
\hline "OnMouseLeave" & Input event: The user drags the mouse pointer away from the element. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "OnMouseMove" & Input event: The user moves the mouse pointer over the element area. \\
\hline "OnMouseUp" & \begin{tabular}{l} 
Input events: \\
- \begin{tabular}{l} 
The user releases the mouse button within the element area. It is irrelevant \\
whether the user has previously pressed the mouse button inside or outside \\
the element area.
\end{tabular} \\
- The user presses the mouse button within the element area, leaves the \\
element area, and then releases the mouse button. \\
Note: This CODESYS-specific triggering behavior guarantees that actions for \\
key elements are completed. A key element starts an action for "OnMouseDown"" \\
and ends the action for "OnMouseUp". \\
Example: A visualization user presses the mouse button within the element area \\
of the key element and then moves the cursor position so that it lies outside the \\
element area. The action is ended anyway because "OnMouseUp" is triggered.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Tap" & \begin{tabular}{l} 
When a mouse click event occurs, the variable defined in "Variable" is described \\
in the application. The coding depends on the "Tap FALSE" and "Tap on enter if \\
captured" options.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL) that is set on mouse click event. \\
Example: PLC_PRG. bIsTapped \\
TRUE: A mouse click event exists. It lasts as long as the user presses the mouse \\
button over the element. It ends when the button is released. \\
FALSE: A mouse click event does not exist. \\
Requirement: The "Tap FALSE" option is not activated.
\end{tabular} \\
\hline "Tap FALSE" & \begin{tabular}{l} 
V: The mouse click event leads to a complementary value in "Variable". \\
TRUE: A mouse click event does not exist. \\
FALSE: While the mouse click event exists.
\end{tabular} \\
\hline "Tap on enter if captured" & \begin{tabular}{l} 
FA: During user input, it is also taken into consideration whether the mouse \\
pointer is dragged within the element area or not while the mouse button is \\
pressed. \\
TRUE: While the mouse click event exists and the mouse pointer is moved over \\
the element area. \\
FALSE: A mouse click event does not exist. Or the user moves the mouse \\
pointer outside of the element area while the mouse button is pressed. \\
The value is TRUE again as soon as the user moves the pointer back to the \\
element area. The mouse is then captured.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Toggle" & \begin{tabular}{l} 
With the onset of a mouse click event, the variable is set; when the mouse click \\
event is completed, the variable is reset.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL). Its value toggled when the mouse click event is ended. This is \\
when the user releases the mouse button while the mouse pointer is over the \\
element area. \\
If the user releases the mouse button while the mouse pointer is outside of the \\
element area, then the mouse click event is not ended and the value is not \\
toggled. \\
Hint: The user can cancel a started toggle input by dragging the mouse pointer \\
out of the element area.
\end{tabular} \\
\hline "Toggle on up if captured" & \begin{tabular}{l} 
V: The value toggles regardless of where the mouse pointer is when the mouse \\
button is released. The mouse is then captured.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Hotkey" & \begin{tabular}{l}
Keyboard shortcut on the element for triggering specific input actions. \\
When the keyboard shortcut event occurs, the input actions in the "Events" property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.
\end{tabular} \\
\hline "Key" & \begin{tabular}{l}
Key pressed for input action. \\
Example: [T] \\
Note: The following properties appear when a key is selected.
\end{tabular} \\
\hline "Events" & \begin{tabular}{l}
- "None" \\
- "Mouse down": Pressing the key triggers the input actions that are configured in the "OnMouseDown" property. \\
- "Mouse up": Releasing the key triggers the input actions that are configured in the "OnMouseUp" property. \\
- "Mouse down/up": Pressing and releasing the key triggers the input actions that are configured in the "OnMouseDown" property and the "OnMouseUp" property.
\end{tabular} \\
\hline "Shift" & Combination with the Shift key Example: [Shift]+[T]. \\
\hline "Control" & Combination with the Ctrl key Example: [Ctrl]+[T]. \\
\hline "Alt" & Combination with the Alt key Example: \([A I t]+[T]\). \\
\hline
\end{tabular}

All keyboard shortcuts and their actions that are configured in the visualization are listed on the "Keyboard Configuration" tab.

\section*{See also}
- ⓨ Chapter 6.4.5.21.2.2 "Command 'Keyboard Configuration'" on page 3341
- \(\xi^{2}\) Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration'" on page 3370

Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages: \\
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
- "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\(\qquad\)\begin{tabular}{l} 
See also \\
- «3 Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366
\end{tabular}
\end{tabular}

See also
- \({ }^{4}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Pie'}

Symbol:

\section*{Category: "Basic"}

The element draws a pie of any angle.

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Example: Error_rate_part_1 \\
Optional \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list.
\end{tabular} \\
\hline "Type of element" & "Pie" \\
\hline
\end{tabular}

Element property 'Position'

The position defines the location and size of the element in the visualization window. This is based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline "X" & \begin{tabular}{l} 
The x-coordinate of the upper left corner of the element \\
Specified in pixels \\
Example: 10
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
The y-coordinate of the upper left corner of the element \\
Specified in pixels \\
Example: 10
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels \\
Example: 30
\end{tabular} \\
\hline & \begin{tabular}{l} 
Tip: You can change the values in " \(X ", " ~ " Y ", ~ " W i d t h ", ~ a n d ~ " H e i g h t " ~ b y ~ d r a g g i n g ~ t h e ~\) \\
corresponding symbols థi- to another position in the editor.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Angle" & \begin{tabular}{l} 
Static angle of rotation (in degrees). \\
Example: 35 \\
The element is displayed rotated in the editor. The point of rotation is the center \\
of the element. A positive value rotates clockwise. \\
Tip: You can change the value in the editor by focusing the element to the \\
handle. When the cursor is displayed as a rotating arrow \\
element about its center as a handle.
\end{tabular} \\
& \begin{tabular}{l} 
(1): Handle can rotate the \\
Note: If a dynamic angle of rotation is also configured in the property \\
"Absolute movement \(\rightarrow\) Internal rotation", then the static and dynamic angles \\
of rotation are added in runtime mode. The static angle of rotation acts as an \\
offset.
\end{tabular} \\
\hline
\end{tabular}

See also
- \(y^{\wedge}\) Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874
\begin{tabular}{|l|l|l|}
\hline "Begin" & \begin{tabular}{l} 
Start angle of the pie. If you also define a variable \\
for the start, then the start angle is calculated from \\
the sum of the values for "Begin" and "Variable for \\
begin".
\end{tabular} & \begin{tabular}{l} 
Example: \\
"Begin": 330 \\
"End": 90
\end{tabular} \\
\hline & \begin{tabular}{l} 
End angle of the pie. If you also define a variable \\
for the end, then the end angle is calculated from \\
the sum of the values for "End" and "Variable for \\
end". \\
The pie is drawn clockwise from the start angle to \\
the end angle.
\end{tabular} \\
\hline "Variable for begin" & The start of the sector is defined dynamically by a variable. \\
\hline "Variable for end" & The end of the sector is defined dynamically by a variable. \\
\hline "Only show circle & V: The pie is drawn without the radius line or filling color. \\
line"
\end{tabular}

\section*{Element prop- \\ erty 'Center'}
\begin{tabular}{|l|l|}
\hline " X " & \begin{tabular}{l} 
Display of the center coordinates. You cannot modify these values here in the \\
properties. \\
If the Pie is selected in the editor, then the center of the Pie (as well as the \\
center of the enveloping box) is visualized with the symbol \(\boldsymbol{\oplus}\). Moreover, the \\
element is decorated with a position, begin, and end boxes that you can move.
\end{tabular} \\
\hline
\end{tabular}

\section*{Element property 'Colors'}
\begin{tabular}{|l|l|}
\hline "Normal state" & \begin{tabular}{l} 
The normal state is in effect if the variable in "Color variables \(\rightarrow\) Toggle color" is \\
not defined or it has the value FALSE.
\end{tabular} \\
\hline "Frame color" & Frame and fill color for the corresponding state of the variable. \\
\hline "Fill color" & \\
\hline "Transparency" & \begin{tabular}{l} 
Transparency value (0 to 255) for defining the transparency of the selected color. \\
Example: \(255:\) The color is opaque. 0: The color is completely transparent.
\end{tabular} \\
\hline "Alarm state" & \begin{tabular}{l} 
The alarm state is in effect if the variable in "Color variables \(\rightarrow\) Toggle color" has \\
the value TRUE.
\end{tabular} \\
\hline "Use gradient color" & \(\boxed{\checkmark}:\) The element is displayed with a gradient of two colors. \\
\hline "Gradient setting" & The "Gradient editor" dialog box opens. \\
\hline
\end{tabular}

See also
- \({ }^{\circ}\), Chapter 6.4.5.21.3.5 "Dialog 'Gradient Editor'" on page 3369

Element prop- The properties contain fixed values for setting the look of the element. erty 'Appearance'
\begin{tabular}{|c|c|}
\hline "Line width" & \begin{tabular}{l}
Value in pixels \\
Example: 2 \\
Note: The values 0 and 1 both result in a line weight of 1 pixel. If no line should be displayed, then the "Line style" property must be set to the option "Invisible".
\end{tabular} \\
\hline "Fill attributes" & \begin{tabular}{l}
The way in which the element is filled. \\
- "Filled":The element is filled with the color from property "Colors \(\rightarrow\) Fill color". \\
- "Invisible": The fill color is invisible.
\end{tabular} \\
\hline "Line style" & \begin{tabular}{l}
Type of line representation \\
- "Solid" \\
- "Dashes" \\
- "Dots" \\
- "Dash Dot" \\
- "Dash Dot Dot" \\
- "not visible"
\end{tabular} \\
\hline
\end{tabular}

You can assign variables in the "Appearance variables" property for controlling the appearance dynamically. The fixed values here are overwritten.

See also
- « "Element property 'Appearance variables'" on page 3476

Element prop- The properties contains character strings for labeling the element. The character string can also erty 'Texts' contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.
CODESYS accepts the specified texts automatically into the "GlobalTextList" text list. Therefore, these texts can be localized.
\begin{tabular}{|l|l|}
\hline "Text" & \begin{tabular}{l} 
Character string (without single straight quotation marks) for the labeling the \\
element. Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter]. \\
Example: Accesses: \%i \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Text".
\end{tabular} \\
\hline "Tooltip" & \begin{tabular}{l} 
Character string (without single straight quotation marks) that is displayed as the \\
tooltip of an element. \\
Example: Number of valid accesses. \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Tooltip".
\end{tabular} \\
\hline
\end{tabular}

See also
- \#2 "Element property 'Text variables"' on page 3456
- Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872
- Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329

Element prop- The properties contain fixed values for the text properties.
\begin{tabular}{|c|c|}
\hline "Horizontal alignment" & Horizontal alignment of the text within the element. \\
\hline "Vertical alignment" & Vertical alignment of the text within the element. \\
\hline "Text format" & \begin{tabular}{l}
Definition for displaying texts that are too long \\
- "Default": The long text is truncated. \\
- "Line break": The text is split into parts. \\
- "Ellipsis": The visible text ends with "..." indicating that it is not complete.
\end{tabular} \\
\hline "Font" & Example: "Default"
The "Font" dialog box opens.
Drop-down list with style fonts. \\
\hline "Font color" & Example: "Black"
The "Color" dialog box opens.
Drop-down list with style colors. \\
\hline "Transparency" & \begin{tabular}{l}
Whole number (value range from 0 to 255). This determines the transparency of the respective color. \\
Example: 255: The color is opaque. \\
0 : The color is completely transparent. \\
Please note: If the color is a style color and already has a transparency value, then this property is write-protected.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" & \multicolumn{1}{|c|}{\begin{tabular}{l} 
"X" \\
\hline Variable (integer data type). Defines the X position (in pixels). \\
Example: PLC_PRG. iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular}} \\
\hline V" & \begin{tabular}{l} 
Variable (integer data type). Defines the Y position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline "Scaling" & \begin{tabular}{l} 
Variable (integer data type). Causes centric stretching. \\
Example: PLC_PRG. iScaling. \\
The reference point is the "Center" property. \\
The value 1 shrinks the element by a factor of 0.001. The value 1000 returns the element \\
to its original size.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Interior rotation" & \begin{tabular}{l} 
Variable (integer data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center"according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\boldsymbol{\oplus}\) symbol. \\
If a static angle of rotation is specified in "Position \\
\(\rightarrow\) Angle", then the static angle of rotation and the \\
angle of rotation are added.
\end{tabular} \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

The " \(X\) ", " \(Y\) ", and "Interior rotation" properties are supported by the "Client Animation" functionality.

Element prop- These properties are variables with contents that replace a format definition. erty 'Text varia-
bles'
\begin{tabular}{|l|l|}
\hline "Text variable" & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses \\
Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Text".
\end{tabular} \\
& \begin{tabular}{l} 
Note: If you specify a variable of type enumeration with text list support, then \\
the name of the enumeration data type is added automatically in angle brackets \\
after the variable name. Example: PLC_PRG.enVar <enumeration name>. \\
Then the symbolic value of the enumeration component is printed instead of the \\
numeric value when text is printed. Refer to the help page for the enumerations.
\end{tabular} \\
\hline & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses InTooltip
\end{tabular} \\
Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Tooltip". \\
\hline
\end{tabular}

See also
- \({ }^{*}\) Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329
- " "Element property 'Texts"' on page 3454
- © Chapter 6.4.1.20.5.18 "Enumerations" on page 2263

Element property 'Dynamic texts'

Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is currently indexed in the variable.
\begin{tabular}{|c|c|}
\hline "Text list" & \begin{tabular}{l}
Variable (string) or name of the text list as a fixed string in single straight quotation marks. \\
Example: 'Errorlist' \\
: Drop-down list with the dialogs available in the text lists.
\end{tabular} \\
\hline "Text index" & \begin{tabular}{l}
Text list ID. This refers to the desired output text. \\
- As fixed string with the ID in single straight quotation marks. \\
Example: '1' \\
- As a variable (STRING) for dynamically controlling the text output. \\
Example: strTextID \\
Sample assignment: PLC_PRG.strTextID := '1';
\end{tabular} \\
\hline "Tooltip index" & \begin{tabular}{l}
Text list ID. This refers to the desired output text. \\
- As fixed string with the ID in single straight quotation marks. \\
Example: '2' \\
- As a variable (STRING) for dynamically controlling the text output. \\
Example: strToolTipID \\
Sample assignment: PLC_PRG.strToolTipID := '2';
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{\text {• }}\), Chapter 6.4.1.21.2.28 "Object 'Text List"' on page 2532

Element prop- The variables allow for dynamic control of the text display. erty 'Font variables'
\begin{tabular}{|c|c|}
\hline "Font name" & \begin{tabular}{l}
Variable (STRING). Includes the font of the text. \\
Example: PLC_PRG.stFontVar := 'Arial'; \\
The selection of fonts corresponds to the default "Font" dialog.
\end{tabular} \\
\hline "Size" & \begin{tabular}{l}
Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name. \\
- <pt>: Points (default) \\
Example: PLC_PRG.iFontHeight <pt> \\
Code: iFontHeight : INT := 12; \\
- <px> : Pixels \\
Example: PLC_PRG.iFontHeight <px> \\
Code: iFontHeight : INT := 19; \\
pt \\
pt \\
px \\
If you click in the value field, a drop-down list opens on the right for setting the unit. \\
Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property "Text property \(\rightarrow\) Font".
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Flags" & \begin{tabular}{l}
Variable (DWORD). Contains the flags for displaying fonts. \\
Flags: \\
- 1: Italics \\
- 2: Bold \\
- 4: Underline \\
- 8: Strikethrough \\
Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: PLC_PRG.dwFontType := 6;
\end{tabular} \\
\hline "Character set" & \begin{tabular}{l}
Variable (DWORD). Contains a character set number for the font. \\
The selection of character set numbers corresponds to the "Script" setting of the standard "Font" dialog.
\end{tabular} \\
\hline "Color" & \begin{tabular}{l}
Variable (DWORD). Includes the color of the text. \\
Example: PLC_PRG.dwColorFont:= 16\#FF000000;
\end{tabular} \\
\hline "Flags for text alignment" & \begin{tabular}{l}
Variable (integer data type). Contains the coding for text alignment. \\
Example: PLC_PRG.dwTextAlignment. \\
Coding: \\
- 0 : Top left \\
- 1: Horizontal center \\
- 2: Right \\
- 4: Vertical center \\
- 8: Bottom \\
Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: PLC_PRG.dwFontType := 5;
\end{tabular} \\
\hline
\end{tabular}

Fixed values for displaying texts are set in "Text properties".

Element prop- The Element property is used as an interface for project variables to dynamically control colors erty 'Color varia- at runtime. bles'
\begin{tabular}{|c|c|}
\hline "Toggle color" & \begin{tabular}{l}
The property controls the toggled color at runtime. \\
Value assignment: \\
- FALSE: The element is displayed with the color specified in the "Color" property. \\
- TRUE: The element is displayed with the color specified in the "Alarm color" property. \\
Assignment options: \\
- Placeholder for the user input variable \\
- "<toggle/tap variable>" \\
- "<NOT toggle/tap variable>" \\
The color change is not controlled by its own variable, but by a user input variable. \\
Note: Specify a variable for the mouse events "Tap" or "Toggle" in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both "Toggle" and "Tap", then the variable specified in "Tap" is used. \\
Hint: Click the symbol to insert the placeholder "<toggle/tap variable>". When you activate the "Inputconfiguration", "Tap FALSE" property, then the "<NOT toggle/tap variable>" placeholder is displayed. \\
- Instance path of a project variable (BOOL) \\
Example: PLC_PRG.xColorIsToggeled \\
Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.
\end{tabular} \\
\hline \begin{tabular}{l}
"Normal state" \\
"Alarm state"
\end{tabular} & The properties listed below control the color depending on the state. The normal state is in effect if the variable in "Color variables", "Toggle color" is not defined or it has the value FALSE. The alarm state is in effect if the variable in "Colorvariables", "Toggle color" has the value TRUE. \\
\hline "Frame color" & \begin{tabular}{l}
Assignment options: \\
- Variable (DWORD) for the frame color Example: PLC_PRG.dwBorderColor \\
- Color literal Example of green and opaque: 16\#FF00FF0 0
\end{tabular} \\
\hline "Filling color" & \begin{tabular}{l}
Assignment options: \\
- Variable (DWORD) for the fill color Example: PLC_PRG.dwFillColor \\
- Color literal Example of gray and opaque: \(16 \#\) FF888888
\end{tabular} \\
\hline
\end{tabular}

The transparency part of the color value is evaluated only if the "Activate semitransparent drawing" option of the visualization manager is selected.

Select the "Advanced" option in the toolbar of the properties view. Then all element properties are visible.

\section*{See also}
- \& Chapter 6.4.5.10.4 "Animating a color display" on page 2914

Element prop- The properties contain IEC variables for controlling the appearance of the element dynamically. erty 'Appearance variables'
\begin{tabular}{|l|l|}
\hline "Line width" & Variable (integer data type). Contains the line weight (in pixels). \\
\hline "Fill attributes" & Variable (DWORD). Controls whether the fill color of the element is visible. \\
& - Variable value = 0: Filled \\
& - Variable value > 0: Invisible; no fill color \\
\hline "Line style" & Variable (DWORD). Controls the line style. \\
& Coding: \\
& - \(0:\) Solid line \\
& - 1: Dashed line \\
& - 2: Dotted line \\
& - 3: Line type "Dash Dot" \\
& - 3: Line type "Dash Dot Dot" \\
& - 8: Invisible; no line \\
\hline
\end{tabular}

Fixed values can be set in the "Appearance" property. These values can be overwritten by dynamic variables at runtime.

\section*{See also}
- „y "Element property 'Appearance"' on page 3468

Element prop- The variables control the element behavior dynamically. erty 'State varia-

\section*{bles'}
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
\\
\\
\\
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \(=\) FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", "X", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain the configurations for the user input when using the mouse or keyboard. erty 'Input con- A user input defines an event and one or more actions that are executed when an event occurs. figuration'
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
The "Configure" button opens the "Input Configuration" dialog. There you can create or edit user inputs. \\
Configured user inputs are listed below the events. They each include the action that is triggered and the setting \\
in short form. \\
Example: "Execute ST Code":
\end{tabular} \\
\hline "Onc__PRG.i_x \(:=0 ;\) \\
\hline "OnDialogClosed" & Input event: The user closes the dialog. \\
\hline "OnMouseDown" & \begin{tabular}{l} 
Input event: The user clicks the mouse button completely in the element area. \\
The mouse button is clicked and released.
\end{tabular} \\
\hline "OnMouseEnter" & Input event: The user clicks down on the mouse button. \\
\hline "OnMouseLeave" & Input event: The user drags the mouse pointer to the element. \\
\hline
\end{tabular}
\(\left.\begin{array}{|l|l|}\hline \text { "OnMouseMove" } & \text { Input event: The user moves the mouse pointer over the element area. } \\ \hline \text { "OnMouseUp" } & \begin{array}{l}\text { Input events: } \\ \text { - } \\ \text { The user releases the mouse button within the element area. It is irrelevant } \\ \text { whether the user has previously pressed the mouse button inside or outside } \\ \text { the element area. }\end{array} \\ \text { - The user presses the mouse button within the element area, leaves the } \\ \text { element area, and then releases the mouse button. } \\ \text { Note: This CODESYS-specific triggering behavior guarantees that actions for } \\ \text { key elements are completed. A key element starts an action for "OnMouseDown" } \\ \text { and ends the action for "OnMouseUp". } \\ \text { Example: A visualization user presses the mouse button within the element area } \\ \text { of the key element and then moves the cursor position so that it lies outside the } \\ \text { element area. The action is ended anyway because "OnMouseUp" is triggered. }\end{array}\right\}\)
\begin{tabular}{|l|l|}
\hline "Tap" & \begin{tabular}{l} 
When a mouse click event occurs, the variable defined in "Variable" is described \\
in the application. The coding depends on the "Tap FALSE" and "Tap on enter if \\
captured" options.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL) that is set on mouse click event. \\
Example: PLC_PRG . bI sTapped \\
TRUE: A mouse click event exists. It lasts as long as the user presses the mouse \\
button over the element. It ends when the button is released.
\end{tabular} \\
& \begin{tabular}{l} 
FALSE: A mouse click event does not exist. \\
Requirement: The "Tap FALSE" option is not activated.
\end{tabular} \\
\hline "Tap FALSE" & \begin{tabular}{l} 
V: The mouse click event leads to a complementary value in "Variable". \\
TRUE: A mouse click event does not exist.
\end{tabular} \\
\hline "Tap on enter if captured" & \begin{tabular}{l} 
FALSE: While the mouse click event exists.
\end{tabular} \\
\hline \begin{tabular}{l} 
pointer is dragged within the element area or not while the mouse button is \\
pressed. \\
TRUE: While the mouse click event exists and the mouse pointer is moved over \\
the element area. \\
FALSE: A mouse click event does not exist. Or the user moves the mouse \\
pointer outside of the element area while the mouse button is pressed. \\
The value is TRUE again as soon as the user moves the pointer back to the \\
element area. The mouse is then captured.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Toggle" & \begin{tabular}{l} 
With the onset of a mouse click event, the variable is set; when the mouse click \\
event is completed, the variable is reset.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL). Its value toggled when the mouse click event is ended. This is \\
when the user releases the mouse button while the mouse pointer is over the \\
element area. \\
If the user releases the mouse button while the mouse pointer is outside of the \\
element area, then the mouse click event is not ended and the value is not \\
toggled. \\
Hint: The user can cancel a started toggle input by dragging the mouse pointer \\
out of the element area.
\end{tabular} \\
\hline "Toggle on up if captured" & \begin{tabular}{l} 
V: The value toggles regardless of where the mouse pointer is when the mouse \\
button is released. The mouse is then captured.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Hotkey" & \begin{tabular}{l}
Keyboard shortcut on the element for triggering specific input actions. \\
When the keyboard shortcut event occurs, the input actions in the "Events" property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.
\end{tabular} \\
\hline "Key" & \begin{tabular}{l}
Key pressed for input action. \\
Example: [T] \\
Note: The following properties appear when a key is selected.
\end{tabular} \\
\hline "Events" & \begin{tabular}{l}
- "None" \\
- "Mouse down": Pressing the key triggers the input actions that are configured in the "OnMouseDown" property. \\
- "Mouse up": Releasing the key triggers the input actions that are configured in the "OnMouseUp" property. \\
- "Mouse down/up": Pressing and releasing the key triggers the input actions that are configured in the "OnMouseDown" property and the "OnMouseUp" property.
\end{tabular} \\
\hline "Shift" & Combination with the Shift key Example: [Shift]+[T]. \\
\hline "Control" & Combination with the Ctrl key Example: [Ctrl]+[T]. \\
\hline "Alt" & Combination with the Alt key Example: \([A I t]+[T]\). \\
\hline
\end{tabular}

All keyboard shortcuts and their actions that are configured in the visualization are listed on the "Keyboard Configuration" tab.

\section*{See also}
- ⓨ Chapter 6.4.5.21.2.2 "Command 'Keyboard Configuration'" on page 3341
- \(\xi^{2}\) Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration'" on page 3370

Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'
\begin{tabular}{|c|c|}
\hline "Access rights" & \begin{tabular}{l}
Opens the "Access rights" dialog. There you can edit the access privileges for the element. \\
Status messages: \\
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
- "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline \multicolumn{2}{|l|}{See also} \\
\hline & 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366 \\
\hline
\end{tabular}

\section*{Visualization Element 'Image'}

Symbol:

The element adds an image to the visualization. The displayed image is managed in the image pool and referenced in the visualization element by means of a static ID. You can also change the displayed image dynamically by using a variable instead of the static ID.


Directories that contain the images for use in visualizations can be defined in the project settings (category "Visualization").

\section*{Element proper-}
ties
\begin{tabular}{|c|c|}
\hline "Element name" & \begin{tabular}{l}
Example: Status bar \\
Optional \\
Hint: Assign individual names for elements so that they are found faster in the element list.
\end{tabular} \\
\hline "Type of element" & "Image" \\
\hline "Static ID" & \begin{tabular}{l}
Identifier of the image file for a static assignment \\
ID of the image file on, as it is defined in the corresponding image pool. If the image is not included in the global image pool in the POU view, then the instance path must be specified. Then the name of the image pool is preceded to make the entry unique. Example: imagepool2.button_image. \\
When a new ID is specified, a file selection dialog opens. The selected file is saved to the "GloballmagePool". \\
See also: Help for the "Image Pool" object.
\end{tabular} \\
\hline "Show frame" & \(\square\) : The image file is displayed with a frame. \\
\hline "Clipping" & Requirement: The "Scaling type" property is "Fixed".
Only part of the visualization is displayed that fits in the element frame. \\
\hline "Transparent" & The image pixels that have the "Transparent color" are displayed as transparent. \\
\hline "Transparent color" & \begin{tabular}{l}
Effective only if the "Transparent" option is activated. \\
The... button opens the color selection dialog. This is where you select the transparent color.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Scaling type" & \begin{tabular}{l}
Definition of how an image fits in the element frame. \\
- "Isotropic": The entire image is displayed in the element frame, either larger or smaller. As a result, the proportion of height and width are retained. If the alignment of the elements to each other should also be retained within a scaled frame element, then note the following. Unwanted horizontal or vertical offsets can be prevented by setting the properties "Horizontal alignment" and "Vertical alignment" to "Centered". The alignment of the elements is retained and there are no resulting horizontal or vertical offsets. Example: A lamp is centered above a switch. The lamp should remain in the horizontally centered position, even if the frame is resized. \\
- "Anisotropic": The image resizes automatically to the dimensions of the element frame, filling the entire element frame. As a result, the proportions are not retained. \\
- "Fixed": The image retains its original size, even if the element frame is resized. Note also that the "Clipping" option is selected. \\
For each reassignment of an image ID, the element size is adapted automatically to the image size.
\end{tabular} \\
\hline "Horizontal alignment" & \begin{tabular}{l}
Horizontal alignment of the element within the element frame: \\
- "Left" \\
- "Centered" \\
- "Right" \\
Requirement: The scaling type of the image is "Isotropic" or "Fixed". \\
Note: If the visualization is referenced, then the horizontal alignment takes effect within the frame position. \\
*: The "Variable" property is shown below this.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Enumeration variable (ENUM \\
VisuElemBase.VisuEnumVerticalAlignment). Contains the horizontal alignment. \\
Example: PLC_PRG.eHorizontalAlignment
\end{tabular} \\
\hline "Vertical alignment" & \begin{tabular}{l}
Vertical alignment of the element within the element frame: \\
- "Top" \\
- "Centered" \\
- "Bottom" \\
Requirement: The scaling type of the image is "Isotropic" or "Fixed". \\
Note: If the visualization is referenced, then the horizontal alignment takes effect within the frame position. \\
中: The "Variable" property is shown below this.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Enumeration variable (ENUM \\
VisuElemBase.VisuEnumVerticalAlignment). Contains the vertical alignment. \\
Example: PLC_PRG.eVerticalAlignment
\end{tabular} \\
\hline
\end{tabular}


See also
- Object 'Image Pool'

Element prop- The position defines the location and size of the element in the visualization window. This is erty 'Position' based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X "\) & \begin{tabular}{l} 
The x-coordinate of the upper left corner of the element \\
Specified in pixels \\
Example: 10
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
The y-coordinate of the upper left corner of the element \\
Specified in pixels \\
Example: 10
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels \\
Example: 30
\end{tabular} \\
\hline & \begin{tabular}{l} 
Tip: You can change the values in " \(X ", " Y ", ~ " W i d t h ", ~ a n d ~ " H e i g h t " ~ b y ~ d r a g g i n g ~ t h e ~\) \\
corresponding symbols థi- to another position in the editor.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Angle" & \begin{tabular}{l} 
Static angle of rotation (in degrees). \\
Example: 35 \\
The element is displayed rotated in the editor. The point of rotation is the center \\
of the element. A positive value rotates clockwise. \\
Tip: You can change the value in the editor by focusing the element to the \\
handle. When the cursor is displayed as a rotating arrow A, you can rotate the \\
element about its center as a handle.
\end{tabular} \\
& \begin{tabular}{l} 
"Absolute movement \(\rightarrow\) Internal rotation", then the static and dynamic angles \\
of rotation are added in runtime mode. The static angle of rotation acts as an \\
offset.
\end{tabular} \\
(1) Handle
\end{tabular}

See also
- \(\boldsymbol{y}^{\boldsymbol{y}}\) Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center'
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline " \(Y\) " & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus}\) ) to other positions in the editor.

Element prop- The properties contain fixed values for setting colors. erty 'Colors'
\begin{tabular}{|l|l|}
\hline "Color" & \begin{tabular}{l} 
Color for the frame \\
Requirement: "Show frame" property is activated. \\
\\
\\
\hline Please note that the normal state is in effect if the expression in the \\
"Color variables \(\rightarrow\) Toggle color" property is not defined or it has the value \\
FALSE.
\end{tabular} \\
\hline & \begin{tabular}{l} 
Color for the frame in alarm state \\
Requirement: "Show frame" property is activated. \\
Please note that the alarm state is in effect if the expression in the \\
"Color variables \(\rightarrow\) Toggle color" property has the value TRUE.
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l} 
Value (0 to 255) for defining the transparency of the selected color. \\
Example 255: The color is opaque. 0: The color is completely transparent.
\end{tabular} \\
\hline
\end{tabular}

See also
- \& Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

Element prop- The properties contain fixed values for setting the look of the element. erty 'Appearance'
\begin{tabular}{|l|l|}
\hline "Line width" & Value in pixels \\
& Example: 2 \\
& \begin{tabular}{l} 
Note: The values 0 and 1 both result in a line weight of one pixel. If no line \\
should be displayed, then the "Line style" property must be set to the option \\
"Invisible".
\end{tabular} \\
\hline "Line style" & \begin{tabular}{l} 
Type of line representation \\
\\
\\
\\
\\
\\
\\
\\
\\
\\
• "Solid" "Dashes" \\
• "Dots" \\
• "Dash Dot" \\
• "not visible"
\end{tabular} \\
\hline
\end{tabular}

You can assign variables in the "Appearance variables" property for controlling the appearance dynamically. The fixed values are defined here.

See also
- 芴 "Element property 'Appearance variables'" on page 3476

Element prop- The properties contains character strings for labeling the element. The character string can also erty 'Texts' contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.
CODESYS accepts the specified texts automatically into the "GlobalTextList" text list. Therefore, these texts can be localized.
\begin{tabular}{|l|l|}
\hline "Text" & \begin{tabular}{l} 
Character string (without single straight quotation marks) for the labeling the \\
element. Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter]. \\
Example: Accesses: \%i \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Text".
\end{tabular} \\
\hline "Tooltip" & \begin{tabular}{l} 
Character string (without single straight quotation marks) that is displayed as the \\
tooltip of an element. \\
Example: Number of valid accesses. \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Tooltip".
\end{tabular} \\
\hline
\end{tabular}

See also
- «4 "Element property 'Text variables'" on page 3471
- « Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872
- \(\Longleftrightarrow\) Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329

Element prop- The properties contain fixed values for the text properties.
erty 'Text prop-
erties'
\begin{tabular}{|c|c|}
\hline "Horizontal alignment" & Horizontal alignment of the text within the element. \\
\hline "Vertical alignment" & Vertical alignment of the text within the element. \\
\hline "Text format" & \begin{tabular}{l}
Definition for displaying texts that are too long \\
- "Default": The long text is truncated. \\
- "Line break": The text is split into parts. \\
- "Ellipsis": The visible text ends with "..." indicating that it is not complete.
\end{tabular} \\
\hline "Font" & \begin{tabular}{l}
Example: "Default"
The "Font" dialog box opens. \\
- : Drop-down list with style fonts.
\end{tabular} \\
\hline "Font color" & Example: "Black"
: The "Color" dialog box opens.
Drop-down list with style colors. \\
\hline "Transparency" & \begin{tabular}{l}
Whole number (value range from 0 to 255). This determines the transparency of the respective color. \\
Example: 255: The color is opaque. \\
0 : The color is completely transparent. \\
Please note: If the color is a style color and already has a transparency value, then this property is write-protected.
\end{tabular} \\
\hline
\end{tabular}

\section*{Element prop- \\ erty 'Image ID \\ variable'}
\begin{tabular}{|l|l|}
\hline "Image ID" & \begin{tabular}{l} 
Variable (STRING). Contains the image ID. The contents of the string corre- \\
sponds to the description of the "Static ID" property. \\
Example: PLC_PRG.stImageID \(:=\) 'ImagePool_A.Image3'; \\
\hline
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{*}>\) Chapter 6.4.5.20.1.5 "Visualization Element 'Image'" on page 3038
- \({ }^{\bullet}\) Chapter 6.4.1.21.2.15 "Object 'Image Pool"' on page 2468

Element prop- You can use this element property for animating a series of image files. erty 'Dynamic image'
\begin{tabular}{|l|l|}
\hline "Bitmap version" & \begin{tabular}{l} 
Variable (integer data type). Contains the version of the image. \\
If the variable changes, then the visualization re-reads the image referenced in \\
the "Image ID" property and displays it. \\
The visualization displays animations when the image file on the controller is \\
updated continuously, thus incrementing the version variable. The application \\
must be programmed for this. \\
Possible applications \\
- Displaying graphics that are generated by the application \\
- Displaying images that are refreshed by a camera
\end{tabular} \\
\hline
\end{tabular}

\section*{Element property 'Absolute movement'}

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|c|c|c|}
\hline \multicolumn{3}{|l|}{"Movement"} \\
\hline " \(X\) " & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular}} \\
\hline "Y" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular}} \\
\hline "Rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle1. \\
The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol. \\
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
\end{tabular} &  \\
\hline "Scaling" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (integer data type). Causes centric stretching. \\
Example: PLC_PRG.iScaling. \\
The reference point is the "Center" property. \\
The value 1 shrinks the element by a factor of 0.001 . The value 1000 returns the element to its original size.
\end{tabular}} \\
\hline "Interior rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle2. \\
In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. \\
The rotation point is shown as the \(\boldsymbol{\oplus}\) symbol. \\
Note: If a static angle of rotation is specified in the property "Position \(\rightarrow\) Angle", then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.
\end{tabular} &  \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

See also

Element property 'Relative movement'

The properties contains variables for moving the element. The reference point is the position of the element ("Position" property). The shape of the element can change.
\begin{tabular}{|l|l|}
\hline "Movement top-left" & \multicolumn{1}{|c|}{\begin{tabular}{|l|l|}
\hline " & \\
\hline "Y" & \begin{tabular}{l} 
Variable (integer data type). It contains the number (in pixels) that the left edge \\
is moved horizontally. Incrementing the value moves the element to the right. \\
Example: PLC__PRG. iDeltax
\end{tabular} \\
\hline "Movement bottom-right" & \begin{tabular}{l} 
Variable (integer data type). It contains the number (in pixels) that the top edge \\
is moved vertically. Incrementing the value moves the element to the down. \\
Example: PLC_PRRG. iDeltaY
\end{tabular} \\
\hline "X" & \begin{tabular}{l} 
Variable (integer data type). It contains the number (in pixels) that the right edge \\
is moved horizontally. Incrementing the value moves the element to the right. \\
Example: PLC_PRG. iDeltaWidth
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Variable (integer data type). It contains the number (in pixels) that the bottom \\
edge is moved vertically. Incrementing the value moves the element to the down. \\
Example: PLC_PRG. iDeltaHeight
\end{tabular} \\
\hline
\end{tabular}} \\
\hline
\end{tabular}

See also
- «» "Element property 'Absolute movement"' on page 3441

Element prop- These properties are variables with contents that replace a format definition. erty 'Text variables'
\begin{tabular}{|l|l|}
\hline "Text variable" & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses
\end{tabular} \\
& \begin{tabular}{l} 
Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Text". \\
Note: If you specify a variable of type enumeration with text list support, then \\
the name of the enumeration data type is added automatically in angle brackets \\
after the variable name. Example: PLC_PRG.enVar <enume ration name>. \\
Then the symbolic value of the enumeration component is printed instead of the \\
numeric value when text is printed. Refer to the help page for the enumerations.
\end{tabular} \\
\hline & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses InTool tip \\
Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Tooltip".
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{\text {• }}\) Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329
- \(\quad\) - "Element property 'Texts'" on page 3468
- \(\quad\) • 2 Chapter 6.4.1.20.5.18 "Enumerations" on page 2263

Element property 'Dynamic texts'

Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is currently indexed in the variable.
\(\left.\begin{array}{|l|l|}\hline \text { "Text list" } & \begin{array}{l}\text { Variable (string) or name of the text list as a fixed string in single straight quota- } \\ \text { tion marks. } \\ \text { Example: 'Errorlist' }\end{array} \\ \text { v: Drop-down list with the dialogs available in the text lists. }\end{array}\right\}\)

See also
- \({ }^{\text {• }}\) Chapter 6.4.1.21.2.28 "Object 'Text List"' on page 2532

Element prop- The variables allow for dynamic control of the text display.
erty 'Font varia-
bles'
\begin{tabular}{|c|c|}
\hline "Font name" & \begin{tabular}{l}
Variable (STRING). Includes the font of the text. \\
Example: PLC_PRG.stFontVar := 'Arial'; \\
The selection of fonts corresponds to the default "Font" dialog.
\end{tabular} \\
\hline "Size" & \begin{tabular}{l}
Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name. \\
- <pt>: Points (default) \\
Example: PLC_PRG.iFontHeight <pt> \\
Code: iFontHeight : INT := 12; \\
- <px>: Pixels \\
Example: PLC_PRG.iFontHeight <px> \\
Code: iFontHeight : INT := 19; \\
If you click in the value field, a drop-down list opens on the right for setting the unit. \\
Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property "Text property \(\rightarrow\) Font".
\end{tabular} \\
\hline "Flags" & \begin{tabular}{l}
Variable (DWORD). Contains the flags for displaying fonts. \\
Flags: \\
- 1: Italics \\
- 2: Bold \\
- 4: Underline \\
- 8: Strikethrough \\
Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: PLC_PRG.dwFontType \(:=6\);
\end{tabular} \\
\hline "Character set" & \begin{tabular}{l}
Variable (DWORD). Contains a character set number for the font. \\
The selection of character set numbers corresponds to the "Script" setting of the standard "Font" dialog.
\end{tabular} \\
\hline "Color" & \begin{tabular}{l}
Variable (DWORD). Includes the color of the text. \\
Example: PLC_PRG.dwColorFont:= 16\#FF000000;
\end{tabular} \\
\hline "Flags for text alignment" & \begin{tabular}{l}
Variable (integer data type). Contains the coding for text alignment. \\
Example: PLC_PRG.dwTextAlignment. \\
Coding: \\
- 0 : Top left \\
- 1: Horizontal center \\
- 2: Right \\
- 4: Vertical center \\
- 8: Bottom \\
Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: PLC_PRG.dwFontType := 5;
\end{tabular} \\
\hline
\end{tabular}

\section*{Fixed values for displaying texts are set in "Text properties".}

See also
- 会 "Element property 'Text properties'" on page 3469

Element prop- The Element property is used as an interface for project variables to dynamically control colors erty 'Color varia- at runtime. bles'
\begin{tabular}{|c|c|}
\hline "Toggle color" & \begin{tabular}{l}
The property controls the toggled color at runtime. \\
Value assignment: \\
- FALSE: The element is displayed with the color specified in the "Color" property. \\
- TRUE: The element is displayed with the color specified in the "Alarm color" property. \\
Assigning the property: \\
- Placeholder for the user input variable \\
- "<toggle/tap variable>" \\
- "<NOT toggle/tap variable>" \\
The color change is not controlled by its own variable, but by a user input variable. \\
Note: Specify a variable for the mouse events "Tap" or "Toggle" in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both "Toggle" and "Tap", then the variable specified in "Tap" is used. \\
Hint: Click the symbol to insert the placeholder "<toggle/tap variable>". When you activate the "Inputconfiguration", "Tap FALSE" property, then the "<NOT toggle/tap variable>" placeholder is displayed. \\
- Instance path of a project variable (BOOL) \\
Example: PLC_PRG.xColorIsToggeled \\
Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.
\end{tabular} \\
\hline "Color" & \begin{tabular}{l}
Color variable for the frame \\
- Variable (DWORD) for the color \\
Example: PLC_PRG.dwColor \\
- Color literal \\
Example of gray and opaque: 16\#FF888888 \\
Requirement: "Show frame" property is activated. \\
Please note that the normal state is in effect if the expression in the "Colorvariables \(\rightarrow\) Toggle color" property is not defined or it has the value FALSE.
\end{tabular} \\
\hline "Alarm color" & \begin{tabular}{l}
Color variable for the frame in alarm state \\
- Variable (DWORD) for the alarm color \\
Example: PLC_PRG.dwAlarmColor \\
- Color literal \\
Example of red and opaque: 16\#FFFF0000 \\
Please note that the alarm state is in effect if the expression in the "Colorvariables \(\rightarrow\) Toggle color" property has the value TRUE.
\end{tabular} \\
\hline
\end{tabular}

The transparency part of the color value is evaluated only if the "Activate semitransparent drawing" option of the visualization manager is selected.

Select the "Advanced" option in the toolbar of the properties view. Then all element properties are visible.

See also
- « Chapter 6.4.5.10.4 "Animating a color display" on page 2914
- © Chapter 6.4.5.21.4.2 "Object 'Visualization manager'" on page 3398

Element prop- The properties contain variables for controlling the appearance of the element dynamically. erty 'Appearance variables'
\begin{tabular}{|c|c|}
\hline "Line width" & \begin{tabular}{l}
Variable (integer data type). Contains the line weight (in pixels). \\
Note: The values 0 and 1 both result in a line weight of one pixel. If no line should be displayed, then the "Line style" property must be set to the option "Invisible".
\end{tabular} \\
\hline "Line style" & \begin{tabular}{l}
Variable (DWORD). Controls the line style. Coding: \\
- 0 : Solid line \\
- 1: Dashed line \\
- 2: Dotted line \\
- 3: Line type "Dash Dot" \\
- 3: Line type "Dash Dot Dot" \\
- 8: Invisible: The line is not drawn.
\end{tabular} \\
\hline
\end{tabular}

Fixed values can be set in the "Appearance" property. These values can be overwritten by dynamic variables at runtime.

See also
- « "Element property 'Appearance"' on page 3468

Element prop- The variables control the element behavior dynamically. erty 'State varia-
bles'
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \(=\) FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", "X", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain the configurations for the user input when using the mouse or keyboard. erty 'Input con- User input is a user event from the perspective of the element. figuration'

The "Configure" button opens the "Input configuration" dialog box for creating or modifying a user input configuration.
A configuration contains one or more input actions for the respective input event. Existing input actions are displayed below it.
Example: "Execute ST code": З PLC_PRG.i_x := 0;
\begin{tabular}{|l|l|}
\hline "OnDialogClosed" & Input event: The user closes the dialog box. \\
\hline "OnMouseClick" & \begin{tabular}{l} 
Input event: A user clicks the element completely. The mouse button is clicked \\
and released.
\end{tabular} \\
\hline "OnMouseDown" & Input event: A user clicks down on the element only. \\
\hline "OnMouseEnter" & Input event: A user drags the mouse pointer to the element. \\
\hline "OnMouseLeave" & Input event: A user drags the mouse pointer away from the element. \\
\hline "OnMouseMove" & Input event: A user moves the mouse pointer over the element area. \\
\hline "OnMouseUp" & Input event: The user releases the mouse button over the element area. \\
\hline
\end{tabular}

See also
- \({ }^{\circ}{ }^{2}\) Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration'" on page 3370
\begin{tabular}{|l|l|}
\hline "Hotkeys" & \begin{tabular}{l} 
Keyboard shortcut on the element for triggering specific input actions. \\
When the keyboard shortcut event occurs, the input actions in the "Event(s)" \\
property are triggered.
\end{tabular} \\
\hline "Key" & \begin{tabular}{l} 
Key pressed for input action. \\
Example: [T]
\end{tabular} \\
\hline "Event(s)" & \begin{tabular}{l} 
- "None" \\
\(\bullet\) "Mouse down": Pressing the key triggers the input actions that are configured \\
in the "OnMouseDown" property. \\
"Mouse up": Releasing the key triggers the input actions that are configured \\
in the "OnMouseUp" property.
\end{tabular} \\
\hline "Shift" \\
"Mouse down/up": Pressing and releasing the key triggers the input actions \\
that are configured in the "OnMouseDown" property and the "OnMouseUp" \\
property.
\end{tabular}

All keyboard shortcuts and their actions that are configured in the visualization are listed in the "Keyboard configuration" tab.

See also
- « Chapter 6.4.5.21.2.2 "Command 'Keyboard Configuration"" on page 3341

Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages: \\
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
- "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{〔}\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights"' on page 3366

See also
- Project Settings - Visualization
- \& Chapter 6.4.5.21.2.10 "Command 'Background"" on page 3349
- \& Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Frame'}

Symbol:

\section*{Category: "Basic"}

The element serves as a frame in which to display one or more already existing visualizations. You get a structured user interface. The size of the frame can be fixed or scaled. The display area of the referenced visualization then adapts itself to the frame size.

\section*{Element proper- \\ ties}
\begin{tabular}{|l|l|}
\hline "Element name" & Example: refVisUserInfo \\
\hline "Type of element" & "Frame" \\
\hline "Clipping" & \begin{tabular}{l} 
冋: Fixed size. Only that part of the referenced visualization that fits inside the \\
frame is displayed. \\
Requirement: The "Scaling type" property is "Fixed".
\end{tabular} \\
\hline "Show frame" & \begin{tabular}{l} 
Displays the frame \\
- "No frame": The displayed area of the frame does not have borders. \\
- "Frame": The displayed area of the frame has borders. \\
- "No frame with offset": The displayed area of the frame does not have a \\
border and the displayed area of the referenced visualization is reduced \\
inwards by one pixel as compared to the frame area. The gap prevents the \\
referenced visualization from touching any adjacent elements.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Scaling type" & \begin{tabular}{l} 
The method with which the height and width of the referenced visualization are \\
scaled. \\
- "Isotropic": The visualization is scaled to the size of the element. The visuali- \\
- zation retains its proportions with a fixed height/width ratio. \\
- "Anisotropic": The visualization is scaled to the size of the element. The \\
height and width are adapted to the element independently of each other. \\
- "Fixed": the visualization is displayed in its original size without taking into \\
account the size of the element. \\
- "Fixed and scrollable": The visualization is displayed fixed in the element. If it \\
is larger than the element, the element will be provided with scrollbars. \\
Please note: assign variales to the properties "Scroll position variable \\
horizontal" or "Scroll position variable vertical". You can then edit the data \\
of the scrollbar position in the application.
\end{tabular} \\
\hline
\end{tabular}

Element proper- The properties contain variables for the position of the scrollboxes in the scrollbars. You can ties 'Scrollbar then edit the data of the scrollbox position in the application. settings'
\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|l|}{Requirement: the property "Scaling type" is "fixed and scrollable".} \\
\hline "Scroll position variable horizontal" & \multirow[t]{2}{*}{Variable (integer data type, also as array). Contains the position of the horizontal or vertical scrollbox. The array contains the position for every display variant. If the visualization runs on several display variants, then the position changes are decoupled from each other.} \\
\hline "Scroll position variable & \\
\hline & Example: \\
\hline & PLC_PRG.iScrollhor [CURRENTCLIENTID] \\
\hline & PLC_PRG.iScrollVer[CURRENTCLIENTID] \\
\hline & The variable is declared as an array in the example. \\
\hline & iScrollHor: ARRAY[0..20] OF INT; \\
\hline & iScrollVer: ARRAY[0..20] OF INT; \\
\hline & CURRENTCLIENTID indexes the current display variant. \\
\hline
\end{tabular}


See also
- Unit conversion
\begin{tabular}{|l|l|}
\hline "Deactivation of the \\
background character" & \begin{tabular}{l} 
alization is drawn as a background bitmap in order to optimize the performance \\
af the visualization. \\
Consequence: Elements can be displayed in an unexpected order at runtime.
\end{tabular} \\
& \begin{tabular}{l} 
For example, an animated element can push itself behind the Frame at runtime. \\
鸟: Background character is deactivated in order to avoid the behavior described \\
above.
\end{tabular} \\
\hline
\end{tabular}

Element prop- Contains the currently configured visualization references as a subnode
erty 'Referen-
ces'
\begin{tabular}{|l|l|}
\hline "References" & \begin{tabular}{l} 
Clicking "Configure" opens the "Frame Configuration" dialog. This is used to \\
manage the referenced visualizations. \\
Caution: Visualizations can be nested at any depth by means of Frame ele- \\
ments. In order to use the "Switch to any visualization" Frame selection type \\
without any problems, a Frame must not contain more than 21 referenced \\
visualizations. For more information, see also the description for the "Input \\
configuration" of an element: Action "Switch Frame visualization".
\end{tabular} \\
\hline \begin{tabular}{ll} 
List of the currently referenced \\
visualizations
\end{tabular} & \begin{tabular}{l} 
Visualizations that have a button also have this displayed as a subnode. Each \\
interface variable is listed with the currently assigned transfer parameters. \\
Example: \\
vis_FormA
\end{tabular} \\
& \begin{tabular}{l} 
- iDataToDisplay_1 : PLC_PRG. iVar1 \\
iDataToDisplay_2 : PLC_PRG. iVar2 \\
Hint: You can change the assignment of the variables to an interface variable \\
here and edit the value field. Or click the "Configure" button instead.
\end{tabular} \\
\hline
\end{tabular}

See also
- « Chapter 6.4.5.21.2.1 "Command 'Interface Editor"' on page 3340
- EChapter 6.4.5.17 "Creating a structured user interface" on page 2940
- « "Input action 'Switch Frame Visualization"' on page 3377

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X "\) & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols ( \(\left(\Phi^{-}\right)\)) to other positions in the editor.

See also
- © Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X "\) & X-coordinate of the point of rotation \\
\hline " \(Y\) " & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\mathbf{\oplus})\) to other positions in the editor.

Element prop- The properties contain fixed values for the colors. erty 'Colors'
\begin{tabular}{|l|l|}
\hline "Color"" & \begin{tabular}{l} 
Color of the frame \\
v: Selection list with style colors appears \\
\(\ldots . .: S t a n d a r d ~ d i a l o g ~ " C o l o r " ~ o p e n s ~ f o r ~ s e l e c t i n g ~ a ~ c o l o r . ~\)
\end{tabular} \\
& \begin{tabular}{l} 
Please note: the normal state is when the boolean variable in the property \\
"Color variables \(\rightarrow\) Toggle color" is not defined or its value is FALSE.
\end{tabular} \\
\hline "Alarm color" & \begin{tabular}{l} 
Color with which the element is filled during the alarm state. \\
Please note: Alarm state is when the value of the boolean variable in the prop- \\
erty "Color variables \(\rightarrow\) Toggle color" is FALSE.
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l} 
Integer number (value range from 255 to 0). Specifies the transparency of the \\
associated color. \\
\(255: ~ T h e ~ c o l o r ~ i s ~ o p a q u e . ~\)
\end{tabular} \\
\(0:\) The color is fully transparent. \\
Please note: If the color is a style color and already contains a transparency \\
value, then this property is write-protected.
\end{tabular}

See also
- ② Chapter 6.4.5.5.4 "Assigning a color" on page 2876

Element prop- The properties contain fixed values for setting the look of the element. erty 'Appearance'
\begin{tabular}{|c|c|}
\hline "Line width" & \begin{tabular}{l}
Value in pixels \\
Example: 2 \\
Note: The values 0 and 1 both result in a line weight of one pixel. If no line should be displayed, then the "Line style" property must be set to the option "Invisible".
\end{tabular} \\
\hline "Line style" & \begin{tabular}{l}
Type of line representation \\
- "Solid" \\
- "Dashes" \\
- "Dots" \\
- "Dash Dot" \\
- "Dash Dot Dot" \\
- "not visible"
\end{tabular} \\
\hline
\end{tabular}

You can assign variables in the "Appearance variables" property for controlling the appearance dynamically. The fixed values are defined here.

See also
- « "Element property 'Appearance variables"' on page 3489

Element prop- The properties contains character strings for labeling the element. The character string can also erty 'Texts' contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.

CODESYS accepts the specified texts automatically into the "GlobalTextList" text list. Therefore, these texts can be localized.
\begin{tabular}{|l|l|}
\hline "Text" & \begin{tabular}{l} 
Character string (without single straight quotation marks) for the labeling the \\
element. Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter]. \\
Example: Accesses : \%i \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Text".
\end{tabular} \\
\hline "Tooltip" & \begin{tabular}{l} 
Character string (without single straight quotation marks) that is displayed as the \\
tooltip of an element. \\
Example: Number of valid accesses. \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Tooltip".
\end{tabular} \\
\hline
\end{tabular}

See also
- 気 "Element property 'Text variables'" on page 3485
- \(\xi^{4}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872
- \({ }^{4}\) Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329

Element prop- The properties contain fixed values for the text properties.
erty 'Text prop-
erties'
\begin{tabular}{|c|c|}
\hline "Horizontal alignment" & Horizontal alignment of the text within the element. \\
\hline "Vertical alignment" & Vertical alignment of the text within the element. \\
\hline "Text format" & \begin{tabular}{l}
Definition for displaying texts that are too long \\
- "Default": The long text is truncated. \\
- "Line break": The text is split into parts. \\
- "Ellipsis": The visible text ends with "..." indicating that it is not complete.
\end{tabular} \\
\hline "Font" & Example: "Default"
: The "Font" dialog box opens.
Drop-down list with style fonts. \\
\hline "Font color" & Example: "Black"
: The "Color" dialog box opens.
: Drop-down list with style colors. \\
\hline "Transparency" & \begin{tabular}{l}
Whole number (value range from 0 to 255). This determines the transparency of the respective color. \\
Example: 255: The color is opaque. \\
0 : The color is completely transparent. \\
Please note: If the color is a style color and already has a transparency value, then this property is write-protected.
\end{tabular} \\
\hline
\end{tabular}

\section*{Element property 'Absolute movement'}

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" \\
\hline " \(X "\) & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG. iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Y" & \begin{tabular}{l} 
Variable (numeric data type). Defines the \(Y\) position (in pixels). \\
Example: PLC_PRG. iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\boldsymbol{\oplus}\) \\
symbol. \\
In runtime mode, the alignnent of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
\hline "Scaling" & \begin{tabular}{l} 
Variable (integer data type). Causes centric stretching. \\
Example: PLC_PRG. iScaling. \\
The reference point is the "Center" property. \\
The value 1 shrinks the element by a factor of 0.001. \\
to its original size.
\end{tabular} \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

Element property 'Relative movement'

The properties "X", "Y", "Rotation", and "Interior rotation" are supported by the "Client Animation" functionality.

\section*{See also}
\begin{tabular}{|l|l|}
\hline "Movement top-left" & \\
\hline " \(X\) " & \begin{tabular}{l} 
Variable (integer data type). It contains the number (in pixels) that the left edge \\
is moved horizontally. Incrementing the value moves the element to the right. \\
Example: pLC_PRG.iDeltax
\end{tabular} \\
\hline " \(Y\) " & \begin{tabular}{l} 
Variable (integer data type). It contains the number (in pixels) that the top edge \\
is moved vertically. Incrementing the value moves the element to the down. \\
Example: PLC_PRG.iDeltaY
\end{tabular} \\
\hline "Movement bottom-right" & \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline " \(X\) " & \begin{tabular}{l} 
Variable (integer data type). It contains the number (in pixels) that the right edge \\
is moved horizontally. Incrementing the value moves the element to the right. \\
Example: PLC_PRG.iDeltaWidth
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Variable (integer data type). It contains the number (in pixels) that the bottom \\
edge is moved vertically. Incrementing the value moves the element to the down. \\
Example: PLC_PRG.iDeltaHeight
\end{tabular} \\
\hline
\end{tabular}

See also
- 乡 "Element property 'Absolute movement"' on page 3441

Element prop- These properties are variables with contents that replace a format definition. erty 'Text variables'
\begin{tabular}{|l|l|}
\hline "Text variable" & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses
\end{tabular} \\
& \begin{tabular}{l} 
Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Text". \\
Note: If you specify a variable of type enumeration with text list support, then \\
the name of the enumeration data type is added automatically in angle brackets \\
after the variable name. Example: PLC_PRG.enVar <enumeration name>. \\
Then the symbolic value of the enumeration component is printed instead of the \\
numeric value when text is printed. Refer to the help page for the enumerations.
\end{tabular} \\
\hline "Tooltip variable" & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses InTooltip \\
Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Tooltip".
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{4}\) Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329
- ⓨ "Element property 'Texts'" on page 3482
- \(y^{2}\) Chapter 6.4.1.20.5.18 "Enumerations" on page 2263

Element prop- Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is erty 'Dynamic texts' currently indexed in the variable.
\begin{tabular}{|c|c|}
\hline "Text list" & \begin{tabular}{l}
Variable (string) or name of the text list as a fixed string in single straight quotation marks. \\
Example: 'Errorlist' \\
: Drop-down list with the dialogs available in the text lists.
\end{tabular} \\
\hline "Text index" & \begin{tabular}{l}
Text list ID. This refers to the desired output text. \\
- As fixed string with the ID in single straight quotation marks. Example: '1' \\
- As a variable (STRING) for dynamically controlling the text output. \\
Example: strTextID \\
Sample assignment: PLC_PRG.strTextID := '1';
\end{tabular} \\
\hline "Tooltip index" & \begin{tabular}{l}
Text list ID. This refers to the desired output text. \\
- As fixed string with the ID in single straight quotation marks. \\
Example: '2' \\
- As a variable (STRING) for dynamically controlling the text output. \\
Example: strToolTipID \\
Sample assignment: PLC_PRG.strToolTipID := '2';
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{\wedge} \boldsymbol{y}\) Chapter 6.4.1.21.2.28 "Object 'Text List"' on page 2532

Element prop- The variables allow for dynamic control of the text display. erty 'Font variables'
\begin{tabular}{|c|c|}
\hline "Font name" & \begin{tabular}{l}
Variable (STRING). Includes the font of the text. \\
Example: PLC_PRG.stFontVar := 'Arial'; \\
The selection of fonts corresponds to the default "Font" dialog.
\end{tabular} \\
\hline "Size" & \begin{tabular}{l}
Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name. \\
- <pt>: Points (default) \\
Example: PLC_PRG.iFontHeight <pt> \\
Code: iFontHeight : INT := 12; \\
- <px> : Pixels \\
Example: PLC_PRG.iFontHeight <px> \\
Code: iFontHeight : INT := 19; \\
If you click in the value field, a drop-down list opens on the right for setting the unit. \\
Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property "Text property \(\rightarrow\) Font".
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Flags" & \begin{tabular}{l}
Variable (DWORD). Contains the flags for displaying fonts. \\
Flags: \\
- 1: Italics \\
- 2: Bold \\
- 4: Underline \\
- 8: Strikethrough \\
Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: PLC_PRG.dwFontType :=6;
\end{tabular} \\
\hline "Character set" & \begin{tabular}{l}
Variable (DWORD). Contains a character set number for the font. \\
The selection of character set numbers corresponds to the "Script" setting of the standard "Font" dialog.
\end{tabular} \\
\hline "Color" & \begin{tabular}{l}
Variable (DWORD). Includes the color of the text. \\
Example: PLC_PRG.dwColorFont:= 16\#FF000000;
\end{tabular} \\
\hline "Flags for text alignment" & \begin{tabular}{l}
Variable (integer data type). Contains the coding for text alignment. \\
Example: PLC_PRG.dwTextAlignment. \\
Coding: \\
- \(0:\) Top left \\
- 1: Horizontal center \\
- 2: Right \\
- 4: Vertical center \\
- 8: Bottom \\
Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: PLC_PRG.dwFontType := 5;
\end{tabular} \\
\hline
\end{tabular}

Fixed values for displaying texts are set in "Text properties".

Element prop- The Element property is used as an interface for project variables to dynamically control colors erty 'Color varia- at runtime. bles'
\begin{tabular}{|c|c|}
\hline "Toggle color" & \begin{tabular}{l}
The property controls the toggled color at runtime. \\
Value assignment: \\
- FALSE: The element is displayed with the color specified in the "Color" property. \\
- TRUE: The element is displayed with the color specified in the "Alarm color" property. \\
Assigning the property: \\
- Placeholder for the user input variable \\
- "<toggle/tap variable>" \\
- "<NOT toggle/tap variable>" \\
The color change is not controlled by its own variable, but by a user input variable. \\
Note: Specify a variable for the mouse events "Tap" or "Toggle" in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both "Toggle" and "Tap", then the variable specified in "Tap" is used. \\
Hint: Click the symbol to insert the placeholder "<toggle/tap variable>". When you activate the "Inputconfiguration", "Tap FALSE" property, then the "<NOT toggle/tap variable>" placeholder is displayed. \\
- Instance path of a project variable (BOOL) \\
Example: PLC_PRG.xColorIsToggeled \\
Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.
\end{tabular} \\
\hline "Color" & \begin{tabular}{l}
Color variable for the Frame \\
- Variable (DWORD) for the color \\
Example: PLC_PRG.dwColor \\
- Color literal \\
Example of gray and opaque: 16\#FF888888 \\
Requirement: "Show Frame" property is activated. \\
Please note that the normal state is in effect if the expression in the "Colorvariables \(\boldsymbol{\rightarrow}\) Toggle color" property is not defined or it has the value FALSE.
\end{tabular} \\
\hline "Alarm color" & \begin{tabular}{l}
Color variable for the Frame in alarm state \\
- Variable (DWORD) for the alarm color \\
Example: PLC_PRG.dwAlarmColor \\
- Color literal \\
Example of red and opaque: 16\#FFFF0000 \\
Please note that the alarm state is in effect if the expression in the "Colorvariables \(\boldsymbol{\rightarrow}\) Toggle color" property has the value TRUE.
\end{tabular} \\
\hline
\end{tabular}

The transparency part of the color value is evaluated only if the "Activate semitransparent drawing" option of the visualization manager is selected.

Select the "Advanced" option in the toolbar of the properties view. Then all element properties are visible.

See also
- « Chapter 6.4.5.10.4 "Animating a color display" on page 2914
- \& Chapter 6.4.5.21.4.2 "Object 'Visualization manager'" on page 3398

Element property 'Appearance variables'
\begin{tabular}{|c|c|}
\hline \multirow[t]{2}{*}{"Line width"} & Variable (integer data type). Contains the line weight (in pixels). \\
\hline & Note: The values 0 and 1 both result in a line weight of one pixel. If no line should be displayed, then the "Line style" property must be set to the option "Invisible". \\
\hline "Line style" & \begin{tabular}{l}
Variable (DWORD). Controls the line style. \\
Coding: \\
- \(0:\) Solid line \\
- 1: Dashed line \\
- 2: Dotted line \\
- 3: Line type "Dash Dot" \\
- 3: Line type "Dash Dot Dot" \\
- 8: Invisible: The line is not drawn.
\end{tabular} \\
\hline
\end{tabular}


Fixed values can be set in the "Appearance" property. These values can be overwritten by dynamic variables at runtime.

See also
- © "Element property 'Appearance"' on page 3482

Element property 'Switch frame variable'

The variable controls the switching of the referenced visualizations. This variable indexes one of the referenced frame visualizations and this is displayed in the frame. When the value of the variable changes, it switches to the recently indexed visualization.
\begin{tabular}{|c|c|}
\hline "Variable" & \begin{tabular}{l}
- Variable (integer data type) that contains the index of the active visualization Example: PLC_PRG.uiIndexVisu \\
Hint: The "Frame Configuration" dialog includes a list of referenced visualizations. The visualizations are automatically numerically indexed via the order in the list. \\
Note: This variant of switching usually affects all connected display variants. \\
- Array element (integer data type) for index access via CURRENTCLIENTID Example: PLC_PRG.aIndexVisu[CURRENTCLIENTID] \\
Note: This variant of switching applies to the current client only, and therefore only on one display variant. That is the display variant where the value change was triggered (for example, by means of user input).
\end{tabular} \\
\hline
\end{tabular}

See also
- \& Chapter 6.4.5.21.2.9 "Command 'Frame Selection"' on page 3348

Element prop- The variables control the element behavior dynamically.
erty 'State varia-
bles'
\begin{tabular}{|l|l|}
\hline "Invisible" & Variable (BOOL). Toggles the visibility of the element. \\
& \begin{tabular}{l} 
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \(=\) FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
\\
\\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", "X", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain the configurations for the user input when using the mouse or keyboard. erty 'Input con- A user input defines an event and one or more actions that are executed when an event occurs. figuration'

The "Configure" button opens the "Input Configuration" dialog. There you can create or edit user inputs.
Configured user inputs are listed below the events. They each include the action that is triggered and the setting in short form.
Example: "Execute ST Code": 3 PLC_PRG.i_x := 0;
\begin{tabular}{|l|l|}
\hline "OnDialogClosed" & Input event: The user closes the dialog. \\
\hline "OnMouseClick" & \begin{tabular}{l} 
Input event: The user clicks the mouse button completely in the element area. \\
The mouse button is clicked and released.
\end{tabular} \\
\hline "OnMouseDown" & Input event: The user clicks down on the mouse button. \\
\hline "OnMouseEnter" & Input event: The user drags the mouse pointer to the element. \\
\hline "OnMouseLeave" & Input event: The user drags the mouse pointer away from the element. \\
\hline "OnMouseMove" & Input event: The user moves the mouse pointer over the element area. \\
\hline "OnMouseUp" & \begin{tabular}{l} 
Input events: \\
- The user releases the mouse button within the element area. It is irrelevant
\end{tabular} \\
\hline
\end{tabular}
- The user releases the mouse button within the element area. It is irrelevant whether the user has previously pressed the mouse button inside or outside the element area.
- The user presses the mouse button within the element area, leaves the element area, and then releases the mouse button.

Note: This CODESYS-specific triggering behavior guarantees that actions for key elements are completed. A key element starts an action for "OnMouseDown" and ends the action for "OnMouseUp".

Example: A visualization user presses the mouse button within the element area of the key element and then moves the cursor position so that it lies outside the element area. The action is ended anyway because "OnMouseUp" is triggered.
\begin{tabular}{|c|c|}
\hline "Tap" & When a mouse click event occurs, the variable defined in "Variable" is described in the application. The coding depends on the "Tap FALSE" and "Tap on enter if captured" options. \\
\hline "Variable" & \begin{tabular}{l}
Variable (BOOL) that is set on mouse click event. \\
Example: PLC_PRG.bIsTapped \\
TRUE: A mouse click event exists. It lasts as long as the user presses the mouse button over the element. It ends when the button is released. \\
FALSE: A mouse click event does not exist. \\
Requirement: The "Tap FALSE" option is not activated.
\end{tabular} \\
\hline "Tap FALSE" & \begin{tabular}{l}
The mouse click event leads to a complementary value in "Variable". TRUE: A mouse click event does not exist. \\
FALSE: While the mouse click event exists.
\end{tabular} \\
\hline "Tap on enter if captured" & \begin{tabular}{l}
During user input, it is also taken into consideration whether the mouse pointer is dragged within the element area or not while the mouse button is pressed. \\
TRUE: While the mouse click event exists and the mouse pointer is moved over the element area. \\
FALSE: A mouse click event does not exist. Or the user moves the mouse pointer outside of the element area while the mouse button is pressed. \\
The value is TRUE again as soon as the user moves the pointer back to the element area. The mouse is then captured.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Toggle" & \begin{tabular}{l} 
With the onset of a mouse click event, the variable is set; when the mouse click \\
event is completed, the variable is reset.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (BOoL). Its value toggled when the mouse click event is ended. This is \\
when the user releases the mouse button while the mouse pointer is over the \\
element area. \\
If the user releases the mouse button while the mouse pointer is outside of the \\
element area, then the mouse click event is not ended and the value is not \\
toggled. \\
Hint: The user can cancel a started toggle input by dragging the mouse pointer \\
out of the element area.
\end{tabular} \\
\hline "Toggle on up if captured" & \begin{tabular}{l} 
V: The value toggles regardless of where the mouse pointer is when the mouse \\
button is released. The mouse is then captured.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Hotkey" & \begin{tabular}{l}
Keyboard shortcut on the element for triggering specific input actions. \\
When the keyboard shortcut event occurs, the input actions in the "Events" property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.
\end{tabular} \\
\hline "Key" & \begin{tabular}{l}
Key pressed for input action. \\
Example: [T] \\
Note: The following properties appear when a key is selected.
\end{tabular} \\
\hline "Events" & \begin{tabular}{l}
- "None" \\
- "Mouse down": Pressing the key triggers the input actions that are configured in the "OnMouseDown" property. \\
- "Mouse up": Releasing the key triggers the input actions that are configured in the "OnMouseUp" property. \\
- "Mouse down/up": Pressing and releasing the key triggers the input actions that are configured in the "OnMouseDown" property and the "OnMouseUp" property.
\end{tabular} \\
\hline "Shift" & Combination with the Shift key Example: [Shift]+[T]. \\
\hline "Control" & Combination with the Ctrl key Example: [Ctrl]+[T]. \\
\hline "Alt" & Combination with the Alt key Example: [AIt]+[T]. \\
\hline
\end{tabular}


All keyboard shortcuts and their actions that are configured in the visualization are listed on the "Keyboard Configuration" tab.

See also
- « Chapter 6.4.5.21.2.2 "Command 'Keyboard Configuration"' on page 3341
- © Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration"' on page 3370

Element property 'Access
rights'

Requirement: User management is set up for the visualization.
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages: \\
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
\\
- "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\boldsymbol{y}^{2}\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

See also
- \({ }^{\mu}\) Chapter 6.4.5.17 "Creating a structured user interface" on page 2940
- \(\boldsymbol{y y}^{2}\) Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration"" on page 3370

\section*{Visualization Element 'Label'}

Symbol:

Category: "Common Controls"
The element is used to label visualizations.

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Optional \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list. \\
Example: Header_Parameter
\end{tabular} \\
\hline "Type of element" & "Label" \\
\hline
\end{tabular}

Element prop- The property requires a character string.
erty 'Texts'
This text is entered automatically into the GlobalTextList text list and can be localized there.
\begin{tabular}{|l|l|}
\hline "Text" & \begin{tabular}{l} 
Character string (without single straight quotation marks) \\
Example: Main page
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{2}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872
- \({ }^{*}\) Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329

Element prop-
erty 'Position'
The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X "\) & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline " \(\mathrm{Y} "\) & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (■-) to other positions in the editor.

See also
- Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center' rotation is shown as the \(\oplus\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline " \(Y\) " & \(Y\)-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols \((\boldsymbol{\oplus})\) to other positions in the editor.

Element prop- The properties contain fixed values for the text properties. erty 'Text properties'
\begin{tabular}{|c|c|}
\hline "Horizontal alignment" & Horizontal alignment of the text within the element. \\
\hline "Vertical alignment" & Vertical alignment of the text within the element. \\
\hline "Text format" & \begin{tabular}{l}
Definition for displaying texts that are too long \\
- "Default": The long text is truncated. \\
- "Line break": The text is split into parts. \\
- "Ellipsis": The visible text ends with "..." indicating that it is not complete.
\end{tabular} \\
\hline "Font" & \begin{tabular}{l}
Example: "Default"
The "Font" dialog box opens. \\
- : Drop-down list with style fonts.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline \multirow[t]{3}{*}{"Font color"} & Example: "Black" \\
\hline & ...): The "Color" dialog box opens. \\
\hline & - Drop-down list with style colors. \\
\hline \multirow[t]{4}{*}{"Transparency"} & Whole number (value range from 0 to 255). This determines the transparency of the respective color. \\
\hline & Example: 255: The color is opaque. \\
\hline & 0 : The color is completely transparent. \\
\hline & Please note: If the color is a style color and already has a transparency value, then this property is write-protected. \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\oplus\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
\hline "Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol. \\
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
\hline
\end{tabular}

\section*{You can link the variables to a unit conversion.}

The " \(X\) ", " \(Y\) ", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also
-

Element prop- The variables control the element behavior dynamically. erty 'State variables'
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- Requirement: User management is set up for the visualization. erty 'Access
rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages:
\end{tabular} \\
& \begin{tabular}{l} 
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
\(\bullet\) \\
"Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

See also
- ※y Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

See also
- \(\Leftrightarrow_{2}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Combo Box, Array'}

Symbol:


\section*{Category: "Common Controls"}

The element shows values of an array as a list box. When the visualization user clicks an entry, the array index of the entry is written to an integer variable.

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Optional \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list. \\
Example: List_Product_Number
\end{tabular} \\
\hline "Type of element" & "Combo Box, Array" \\
\hline
\end{tabular}

Element prop- The position defines the location and size of the element in the visualization window. These erty 'Position' are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X\) " & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (區) to other positions in the editor.

See also
- ② Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center' rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline " \(Y\) " & \(Y\)-coordinate of the point of rotation \\
\hline
\end{tabular}

\begin{tabular}{|c|c|}
\hline "Variable" & \begin{tabular}{l}
The array index of the list entry that the user clicks is saved at runtime. \\
Property value \\
- Variable (integer data type) \\
Example: PLC_PRG.iIndexComboboxEntry \\
- Enumeration variable with text list support \\
Example: PLC_PRG.eMyCombobox<COMBO> \\
Note: Value range of the enumeration value that lies within the DWORD or DINT value range
\end{tabular} \\
\hline "Data array" & \begin{tabular}{l}
Displayed as a combo box. Every array component becomes a combo box entry. \\
Property value \\
- Array variable (ARRAY [...] OF) \\
Example: PLC_PRG.astrCombobox \\
Declaration: astrCombobox : ARRAY[0..4] OF STRING := \\
['First', 'Second', 'Third', 'Fourth'];
\end{tabular} \\
\hline
\end{tabular}

\section*{See also}
- Enumerations
- \({ }^{4}\) Chapter 6.4.5.8 "Setting Up Multiple Languages" on page 2906

Element property 'Columns'

The "Combo box - Array" element visualizes an array variable or structure variable in a tabular view. The index of array elements or structure members is shown in a column or row. Twodimensional arrays or structure arrays are shown in several columns. You specify the visualized variable in the "Data array" property. If a variable is assigned there, then you can specify the display of the table columns where the array elements are shown. You can customize each column that is assigned to an index [<n>].
\begin{tabular}{|c|c|}
\hline \begin{tabular}{l}
"Columns" \\
- [<n>]
\end{tabular} & \begin{tabular}{l}
Due to the structure of the variable that is defined in "Data array", CODESYS determines the number of columns and defines them with the index <n>. \\
Example: StringTable : ARRAY [0..2, 0..4] OF STRING := ['BMW','Audi','Mercedes','VW','Fiat', \\
'150','150','150','150','100','blue','gray','silver','blue' \\
'red'];: three columns are formed [0], [1] and [2].
\end{tabular} \\
\hline "Max. array index" & Optional. Variable (integer data type) or value. Defines up to which array index the data is displayed. \\
\hline "Row height" & Height of the rows (in pixels). \\
\hline "Number visible rows" & Optional. If the array is larger than the number of visible rows, then a scrollbar is included. \\
\hline "Scrollbar size" & Width of the vertical scrollbar (in pixels). \\
\hline
\end{tabular}

Table 718: "Element property 'Columns: Column [<n>]""
\begin{tabular}{|l|l|}
\hline "Width" & Column width (in pixels). \\
\hline "Image column" & \begin{tabular}{l} 
局: Images can be displayed in the column. Images are used from the global \\
image pool or user-defined image pools. The image IDs are shown in the cells of \\
the table as defined in the image pool.
\end{tabular} \\
\hline \multicolumn{2}{|l|}{ "Image configuration" }
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Fill mode" & \begin{tabular}{l} 
- "Fill cell" \\
The image resizes to the dimensions of the cell without fixing the height/ \\
width ratio. \\
"Centered" \\
The image is centered in the cell and retains its proportions (height-width \\
ratio).
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l} 
When the "The color that is specified in "Transparent color" is displayed as transparent. \\
displayed. Pixels with this color are transparent.
\end{tabular} \\
\hline "Transparent color" "Text alignment in column" & \begin{tabular}{l} 
- "Left" \\
- "Centered" \\
- "Right"
\end{tabular} \\
\hline
\end{tabular}

\section*{Element prop-} erty 'Texts'
\begin{tabular}{|l|l|}
\hline "Tooltip" & \begin{tabular}{l} 
Character string (without single straight quotation marks) that is displayed as the \\
tooltip of an element in runtime mode \\
Example: Products of customer A \\
Hint: The text is accepted automatically into the "GlobalTextList" text list and can \\
be localized there.
\end{tabular} \\
\hline
\end{tabular}

See also
- 乡y Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

Element prop- The properties contain fixed values for the text properties.
erty 'Text prop-
erties'
\begin{tabular}{|c|c|}
\hline "Usage of" & \begin{tabular}{l}
- "Default style values": The values of the visualization style are used. \\
- "Individual settings": The "Individual text properties" property group is shown The values can be customized here.
\end{tabular} \\
\hline \multicolumn{2}{|l|}{\begin{tabular}{l}
"Individual text properties" \\
Requirement: The "Individual settings" text property is defined.
\end{tabular}} \\
\hline "Font" & \begin{tabular}{l}
Example: "Default"
The "Font" dialog opens. \\
- List box with style fonts
\end{tabular} \\
\hline "Font color" & \begin{tabular}{l}
Example: "Black"
The "Color" dialog opens. \\
- : List box with style colors
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l}
Integer (value range from 0 to 255). This determines the transparency of the respective color. \\
255: The color is opaque. \\
0 : The color is completely transparent. \\
Note: If the color is a style color and already has a transparency value, then this property is write-protected.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG. iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG. iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\boldsymbol{\oplus}\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
\hline "Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol. \\
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

The " \(X\) ", " \(Y\) ", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

\section*{See also}

\section*{Element prop- The variables control the element behavior dynamically. erty 'State variables'}
\begin{tabular}{|l|l|}
\hline "Invisible"" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
\\
\\
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \(=\) FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

See also
- \(y^{2}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Combo Box, Integer'}

\section*{Symbol:}


\section*{Category: "Common Controls"}

The element shows values as a list box. When the user clicks an entry, the ID of the entry is written to an integer variable. The entries in the list box can be from a list and contain images from an image pool.

\section*{Element proper- \\ ties}
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Example: List of product numbers \\
Optional \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list.
\end{tabular} \\
\hline "Type of element" & "Combo Box, Integer" \\
\hline
\end{tabular}

Element prop- The position defines the location and size of the element in the visualization window. These erty 'Position' are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline "X" & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline " \(Y\) " & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols ( ( \(^{-}\)) to other positions in the editor.

See also
- Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center'
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline " \(Y\) " & \(Y\)-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus}\) ) to other positions in the editor.
\begin{tabular}{|c|c|}
\hline "Variable" & \begin{tabular}{l}
At runtime, the text list ID of the list entry that the user clicks is saved at runtime. If only one image pool is displayed, then the image ID is saved. \\
Property value \\
- Variable (integer data type) \\
Example: PLC_PRG.iIDComboboxEntry \\
- Enumeration variable with text list support \\
Example: PLC PRG.eMyCombobox<COMBO>
\end{tabular} \\
\hline "Text List" & \begin{tabular}{l}
Displayed as a combo box. Every text list entry becomes a combo box entry. \\
Note: A maximum of 32766 entries can be displayed. \\
Transfer value \\
- Text list identifier as string \\
Example: 'TextList_A' \\
Note: The IDs of the text list have to be within the range of values of DWORD or DINT. \\
- Blank \\
- When an enumeration variable with text list support is specified in the "Variable" property \\
- When only one image pool is displayed
\end{tabular} \\
\hline "Image Pool" & \begin{tabular}{l}
Displayed as a combo box. Every image in the image pool becomes a combo box entry. \\
Example: 'ImagePool_A'
\end{tabular} \\
\hline
\end{tabular}

See also
- Enumerations
- \(\Leftrightarrow\) Chapter 6.4.5.8 "Setting Up Multiple Languages" on page 2906

Element prop- Displayed list that expands when a visualization user clicks into the element. erty 'Settings of the list'
\begin{tabular}{|c|c|}
\hline "Number of rows setting" & \begin{tabular}{l}
- "From style": \\
- "Explicit": Then the "Number of visible rows" property appears below it.
\end{tabular} \\
\hline "Number of visible rows" & \begin{tabular}{l}
Number of visible lines of the combo box drop-down list defined here \\
- Integer literal \\
Example: 5 \\
- Variable (integer data type) \\
Example: PLC_PRG.iNumberOfVisibleRows \\
Note: The property is available when the "Number of rows setting" property is set to "Explicit".
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Row height" & \begin{tabular}{l}
- "From style": \\
- Literal Example: 20
\end{tabular} \\
\hline "Height of image" & \begin{tabular}{l}
Image height (in pixels) of the image displayed in the drop-down list entry \\
- "From style": \\
- Integer literal \\
Example: 30 \\
Note: Images are displayed only when a value is specified in the "Image pool" property.
\end{tabular} \\
\hline "Width of image" & \begin{tabular}{l}
Image width (in pixels) of the image displayed in the drop-down list entry \\
- "From style": \\
- Literal \\
Example: 30 \\
Note: Images are displayed only when a value is specified in the "Image pool" property.
\end{tabular} \\
\hline "Offset of image" & \begin{tabular}{l}
Makes the images in the selection list appear offset (in pixels) from the left margin. An offset of 0 means that the images are displayed directly on the margin. \\
- "From style": \\
- Literal \\
Example: 4 \\
Note: Images are displayed only when a value is specified in the "Image pool" property.
\end{tabular} \\
\hline "Scrollbar size" & \begin{tabular}{l}
Size of the scrollbar (in pixels). The scrollbar is displayed when more entries are specified in the drop-down list than in "Number of visible rows". \\
Default: 20
\end{tabular} \\
\hline
\end{tabular}

\section*{Element prop-}
erty 'Texts'
\begin{tabular}{|l|l|}
\hline "Tooltip" & \begin{tabular}{l} 
Character string (without single straight quotation marks) that is displayed as the \\
tooltip of an element in runtime mode \\
Example: Products of customer A \\
Hint: The text is accepted automatically into the "GlobalTextList" text list and can \\
be localized there.
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{〔}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Element property 'Value \\ range'}
\begin{tabular}{|l|l|}
\hline "Limit valuerange" & \begin{tabular}{l} 
Limits the text list to one subrange. This subrange is displayed by the combo \\
box. \\
Requirement: A value is specified in the "Text list" property. \\
V: Only the subrange that is defined by the "Minimum value" "Maximum value" \\
properties is displayed as a drop-down list.
\end{tabular} \\
\hline "Minimum value" & \begin{tabular}{l} 
ID of the text list entry from which a combo box entry is displayed \\
- Literal (ANY_NUM) \\
Example: 5
\end{tabular} \\
& \begin{tabular}{l} 
Variable (integer data type) \\
Example: PLC_PRG. iFirstEntry
\end{tabular} \\
\hline "Maximum value" of the text list entry up to which combo box entries are displayed \\
- Literal (ANY_NUM) \\
Example: 10
\end{tabular}

Element prop- The properties contain fixed values for the text properties. erty 'Text properties'
\begin{tabular}{|l|l|}
\hline "Usage of" & \begin{tabular}{l} 
- "Default style values": The values of the visualization style are used. \\
- \\
"Individual settings": The "Individual text properties" property group is shown \\
The values can be customized here.
\end{tabular} \\
\hline \begin{tabular}{l} 
"Individual text properties" \\
Requirement: The "Individual settings" text property is defined.
\end{tabular} \\
\hline "Horizontal alignment" & Horizontal alignment of the text within the element. \\
\hline "Vertical alignment" & Vertical alignment of the text within the element. \\
\hline "Font" & \begin{tabular}{l} 
Example: "Default" \\
\(\ldots . .:\) :The "Font" dialog box opens. \\
v: Drop-down list with style fonts.
\end{tabular} \\
\hline "Font color" & \begin{tabular}{l} 
Example: "Black" \\
"..: The "Color" dialog box opens. \\
v: Drop-down list with style colors.
\end{tabular} \\
\hline & \begin{tabular}{l} 
Whole number (value range from 0 to 255). This determines the transparency of \\
the respective color. \\
Example: 255 : The color is opaque. \\
\(0:\) The color is completely transparent.
\end{tabular} \\
\hline
\end{tabular}

\footnotetext{
Element property 'Absolute movement'
}

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|c|c|c|}
\hline \multicolumn{3}{|l|}{"Movement"} \\
\hline " \(\times\) " & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular}} \\
\hline " \(Y\) " & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the \(Y\) position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular}} \\
\hline "Rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle1. \\
The midpoint of the element rotates at the "Center" point. This rotation point is shown as the \(\boldsymbol{\oplus}\) symbol. \\
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
\end{tabular} &  \\
\hline "Interior rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle2. \\
In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. \\
The rotation point is shown as the \(\boldsymbol{\oplus}\) symbol. \\
Note: If a static angle of rotation is specified in the "Position \(\rightarrow\) Angle" property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.
\end{tabular} &  \\
\hline
\end{tabular}

You can link the variables to a unit conversion.


See also

Element prop- The variables control the element behavior dynamically. erty 'State variables'
\begin{tabular}{|l|l|}
\hline "Invisible" & Variable (BOOL). Toggles the visibility of the element. \\
& \begin{tabular}{l} 
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \(=\) FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
\\
\\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
\\
Status messages: \\
\\
\\
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
\(\bullet \quad\) "Rights are set: Limited rights": Access is restricted for at least one group. \\
\hline
\end{tabular} \\
\hline
\end{tabular}

See also
- \& Chapter 6.4.5.21.3.1 "Dialog 'Access Rights"' on page 3366

See also
- \& Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Tabs'}

Symbol:


\section*{Category: "Common Controls"}

The element displays selected visualizations in tabs. The tabs can be used by means of the tab header without having to configure an input configuration. A visualization user switches between visualizations by clicking the tab header.

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Example: Assembly A \\
Optional \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list.
\end{tabular} \\
\hline "Type of element" & "Tabs" \\
\hline "Tab width" & \begin{tabular}{l} 
Width of the tab (in pixels). If there is not space for all tab headers, then a scroll \\
bar is added. \\
Example: 30
\end{tabular} \\
\hline "Tab height" & \begin{tabular}{l} 
Height of the tab (in pixels) \\
- Integer literal \\
Example: 15
\end{tabular} \\
- "From style"
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Scaling type" & \begin{tabular}{l} 
The method with which the height and width of the referenced visualization are \\
scaled. \\
- "Isotropic": The visualization is scaled to the size of the element. The visuali- \\
- zation retains its proportions with a fixed height/width ratio. \\
- "Anisotropic": The visualization is scaled to the size of the element. The \\
height and width are adapted to the element independently of each other. \\
- "Fixed": the visualization is displayed in its original size without taking into \\
account the size of the element. \\
- "Fixed and scrollable": The visualization is displayed fixed in the element. If it \\
is larger than the element, the element will be provided with scrollbars. \\
Please note: assign variables to the properties "ccroll position variable \\
horizontal" or "Scroll position variable vertical". You can then edit the data \\
of the scrollbar position in the application.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Deactivate background \\
drawing" & \begin{tabular}{l}
\(\square:\) The non-animated elements of the referenced visualization are displayed as \\
background images in order to optimize the performance of the visualization. \\
Result: At runtime, the elements can be displayed in any order, for example \\
when an element moves behind the frame at runtime. \\
目 Deactivates the background display in order to prevent the behavior \\
described above \\
The property is not available for the following settings: \\
- The "Scaling type" property is set to "Fixed and scrollable" \\
- The client animation functionality is enabled.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Scroll bar settings'

The properties include variables for the position of the scroll boxes in the scroll bars. You can process the data for the scroll box position in the application.
\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|l|}{Requirement: The "Scaling type" property is "Fixed and scrollable".} \\
\hline "Scroll position variable horizontal" & \multirow[t]{2}{*}{Variable (integer data type, also array). Includes the position of the horizontal or vertical scroll box. The array contains the position for each display variant. If the visualization is running on multiple display variants, then the position changes} \\
\hline "Scroll position variable, vertical" & \\
\hline & Example: \\
\hline & PLC_PRG.iScrollhor [CURRENTCLIENTID] \\
\hline & PLC_PRG.iScrollVer [CURRENTCLIENTID] \\
\hline & In this example, the variable is declared as an array: \\
\hline & iScrollHor: ARRAY[0..20] OF INT; \\
\hline & iScrollVer: ARRAY[0..20] OF INT; \\
\hline & CURRENTCLIENTID indicates the current display variant. \\
\hline
\end{tabular}

See also
- Unit conversion

\section*{Element property 'References'}
\begin{tabular}{|l|l|}
\hline "References" & \begin{tabular}{l} 
Clicking "Configure" opens the "Frame Configuration" dialog. You can select an \\
existing visualization there. \\
Selected visualization references are shown in the properties. \\
Selected visualization references are listed here as subordinate properties.
\end{tabular} \\
\hline \begin{tabular}{l} 
Name pf the visualization refer- \\
ence (example: PLC_PRG.S1)
\end{tabular} & \\
\hline "Heading" & Tab caption (example: Panel) \\
\hline "Image ID" & \begin{tabular}{l} 
Image ID in the theme <image pool name>. <ID> \\
Example: Imagepool_A. 1 for the image with ID 1 in Imagepool_A
\end{tabular} \\
\hline \begin{tabular}{l} 
Interface parameter of the visu- \\
alization reference \\
Example: iX
\end{tabular} & \begin{tabular}{l} 
If the visualization has an interface, then their parameters are displayed here as \\
subordinate properties. \\
Variable (data type conforms to data type of the interface parameter). Includes \\
the initialization value for the instantiation of the visualization.
\end{tabular} \\
\hline
\end{tabular}

See also
- * Chapter 6.4.5.17 "Creating a structured user interface" on page 2940
- « Chapter 6.4.5.21.2.1 "Command 'Interface Editor"' on page 3340
- \& Chapter 6.4.5.21.2.9 "Command 'Frame Selection"' on page 3348

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X "\) & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline " \(Y\) " & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols ( \(\left(\Phi^{-}\right)\)) to other positions in the editor.

See also
- \# Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X "\) & X-coordinate of the point of rotation \\
\hline " \(Y\) " & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\oplus\) ) to other positions in the editor.

Element property 'Switch frame variable'
\begin{tabular}{|l|l|}
\hline "Variable"" & \begin{tabular}{l} 
Variable (integer data type). Specifies the index of the active visualization. \\
Example: PLC_PRG. uiActiveVisuID. \\
\\
\\
Tip: The "Frame Configuration" dialog box includes a list of selected visualiza- \\
tions. The visualizations are ordered automatically in numeric order in the list.
\end{tabular} \\
\hline
\end{tabular}
- ② Chapter 6.4.5.21.2.9 "Command 'Frame Selection'" on page 3348

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|c|c|c|}
\hline \multicolumn{3}{|l|}{"Movement"} \\
\hline " \(X\) " & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the \(X\) position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular}} \\
\hline "Y" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular}} \\
\hline "Rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle1. \\
The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol. \\
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
\end{tabular} &  \\
\hline "Interior rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle2. \\
In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol. \\
Note: If a static angle of rotation is specified in the "Position \(\rightarrow\) Angle" property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.
\end{tabular} &  \\
\hline
\end{tabular}

\section*{You can link the variables to a unit conversion.}

X, "Y", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also

Element prop- The variables control the element behavior dynamically. erty 'State variables'
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \(=\) FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\Leftrightarrow\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Button'}

Symbol:


\section*{Category: "Common Controls"}

The element triggers an action, such as setting a variable.

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Optional \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list. \\
Example: Voltage_on
\end{tabular} \\
\hline "Type of element" & "Button" \\
\hline
\end{tabular}

\footnotetext{
Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
}
\begin{tabular}{|l|l|}
\hline "X" & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (■-) to other positions in the editor.

See also
- Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center'
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline " \(Y\) " & \(Y\)-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus})\) to other positions in the editor.

Element prop- The properties contain fixed values for setting colors. erty 'Colors'
\begin{tabular}{|l|l|}
\hline "Color" & \begin{tabular}{l} 
Color for the element in its normal state. \\
Ploase note that the normal state is in effect if the expression in the \\
"Color variables \(\rightarrow\) Toggle color" property is not defined or it has the value \\
FALSE.
\end{tabular} \\
\hline "Alarm color" & \begin{tabular}{l} 
Color for the element in alarm state. \\
Please note that the alarm state is in effect if the expression in the \\
"Color variables \(\rightarrow\) Toggle color" property has the value TRUE.
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l} 
Value (0 to 255) for defining the transparency of the selected color. \\
Example 255: The color is opaque. \(0:\) The color is completely transparent.
\end{tabular} \\
\hline "Use gradient color" & 鸟: The element is displayed with a color gradient. \\
\hline "Gradient setting" & The "Color gradient editor" dialog box opens. \\
\hline
\end{tabular}

See also
- « Chapter 6.4.5.21.3.5 "Dialog 'Gradient Editor"' on page 3369
- Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

Element property 'Image'
\begin{tabular}{|c|c|}
\hline "Static ID" & \begin{tabular}{l}
Reference to an image in an image pool of the format <name of image pool>.<image ID> (example: image_pool. GreenButton). \\
If the image is from the "GloballmagePool", then you can omit the name of the image pool because CODESYS always searches this pool first.
\(\qquad\) The "Input Assistant" dialog box opens and lists all available image pools and images in the entire project.
\end{tabular} \\
\hline "Scale type" & \begin{tabular}{l}
Behavior of the image when resizing the button. \\
- "Isotropic": The image retains its proportions. The ratio of height to width is retained, even if you change the height or width of the button separately. \\
- "Anisotropic": The image resizes to the dimensions of the button. \\
- "Fixed": The image retains its original size, even if you change the size of the button.
\end{tabular} \\
\hline "Transparency" & The visualization displays the image with the transparency color that is selected in "Transparency color". \\
\hline "Transparency color" & \begin{tabular}{l}
Color that is transparent in the image (example: "White"). if the image background that is reference by "Static ID" is white, then this background is displayed transparent. Clicking ... opens a color selection dialog. \\
Requirement: The "Transparency" option is activated.
\end{tabular} \\
\hline "Horizontal alignment" & \begin{tabular}{l}
Horizontal alignment of the image \\
- "Left" \\
- "Centered" \\
- "Right"
\end{tabular} \\
\hline "Vertical alignment" & \begin{tabular}{l}
Vertical alignment of the image \\
- "Top" \\
- "Centered" \\
- "Bottom"
\end{tabular} \\
\hline
\end{tabular}

Element property 'Texts'

The properties contains character strings for labeling the element. The character string can also contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.
CODESYS accepts the specified texts automatically into the "GlobalTextList" text list. Therefore, these texts can be localized.
\begin{tabular}{|l|l|}
\hline "Text" & \begin{tabular}{l} 
Character string (without single straight quotation marks) for the labeling the \\
element. Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter]. \\
Example: Accesses : \%i \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Text".
\end{tabular} \\
\hline "Tooltip" & \begin{tabular}{l} 
Character string (without single straight quotation marks) that is displayed as the \\
tooltip of an element. \\
Example: Number of valid accesses. \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Tooltip".
\end{tabular} \\
\hline
\end{tabular}

See also
- *y "Element property 'Text variables'" on page 3519
- \(\xi^{4}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872
- \({ }^{4}\) Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329

Element prop- The properties contain fixed values for the text properties.
erty 'Text prop-
erties'
\begin{tabular}{|c|c|}
\hline "Horizontal alignment" & Horizontal alignment of the text within the element. \\
\hline "Vertical alignment" & Vertical alignment of the text within the element. \\
\hline "Text format" & \begin{tabular}{l}
Definition for displaying texts that are too long \\
- "Default": The long text is truncated. \\
- "Line break": The text is split into parts. \\
- "Ellipsis": The visible text ends with "..." indicating that it is not complete.
\end{tabular} \\
\hline "Font" & Example: "Default"
: The "Font" dialog box opens.
Drop-down list with style fonts. \\
\hline "Font color" & Example: "Black"
: The "Color" dialog box opens.
: Drop-down list with style colors. \\
\hline "Transparency" & \begin{tabular}{l}
Whole number (value range from 0 to 255). This determines the transparency of the respective color. \\
Example: 255: The color is opaque. \\
0 : The color is completely transparent. \\
Please note: If the color is a style color and already has a transparency value, then this property is write-protected.
\end{tabular} \\
\hline
\end{tabular}

\section*{Element property 'Absolute movement'}

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" \\
\hline " \(X "\) & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG. iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Y" & \begin{tabular}{l} 
Variable (numeric data type). Defines the \(Y\) position (in pixels). \\
Example: PLC_PRG. iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\boldsymbol{\oplus}\) \\
symbol. \\
In runtime mode, the alignnent of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
\hline "Scaling" & \begin{tabular}{l} 
Variable (integer data type). Causes centric stretching. \\
Example: PLC_PRG. iScaling. \\
The reference point is the "Center" property. \\
The value 1 shrinks the element by a factor of 0.001. \\
to its original size.
\end{tabular} \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

Element property 'Relative movement'

The properties "X", "Y", "Rotation", and "Interior rotation" are supported by the "Client Animation" functionality.

\section*{See also}
\begin{tabular}{|l|l|}
\hline "Movement top-left" & \\
\hline " \(X\) " & \begin{tabular}{l} 
Variable (integer data type). It contains the number (in pixels) that the left edge \\
is moved horizontally. Incrementing the value moves the element to the right. \\
Example: pLC_PRG.iDeltax
\end{tabular} \\
\hline " \(Y\) " & \begin{tabular}{l} 
Variable (integer data type). It contains the number (in pixels) that the top edge \\
is moved vertically. Incrementing the value moves the element to the down. \\
Example: PLC_PRG.iDeltaY
\end{tabular} \\
\hline "Movement bottom-right" & \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline " \(X\) " & \begin{tabular}{l} 
Variable (integer data type). It contains the number (in pixels) that the right edge \\
is moved horizontally. Incrementing the value moves the element to the right. \\
Example: PLC_PRG.iDeltaWidth
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Variable (integer data type). It contains the number (in pixels) that the bottom \\
edge is moved vertically. Incrementing the value moves the element to the down. \\
Example: PLC_PRG.iDeltaHeight
\end{tabular} \\
\hline
\end{tabular}

See also
- 乡 "Element property 'Absolute movement"' on page 3441

Element prop- These properties are variables with contents that replace a format definition. erty 'Text variables'
\begin{tabular}{|l|l|}
\hline "Text variable" & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses
\end{tabular} \\
& \begin{tabular}{l} 
Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Text". \\
Note: If you specify a variable of type enumeration with text list support, then \\
the name of the enumeration data type is added automatically in angle brackets \\
after the variable name. Example: PLC_PRG.enVar <enumeration name>. \\
Then the symbolic value of the enumeration component is printed instead of the \\
numeric value when text is printed. Refer to the help page for the enumerations.
\end{tabular} \\
\hline "Tooltip variable" & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses InTooltip \\
Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Tooltip".
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{4}\) Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329
- ⓨ "Element property 'Texts'" on page 3516
- \(\quad\) - \(y^{2}\) Chapter 6.4.1.20.5.18 "Enumerations" on page 2263

Element prop- Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is erty 'Dynamic texts' currently indexed in the variable.
\begin{tabular}{|c|c|}
\hline "Text list" & \begin{tabular}{l}
Variable (string) or name of the text list as a fixed string in single straight quotation marks. \\
Example: 'Errorlist' \\
: Drop-down list with the dialogs available in the text lists.
\end{tabular} \\
\hline "Text index" & \begin{tabular}{l}
Text list ID. This refers to the desired output text. \\
- As fixed string with the ID in single straight quotation marks. Example: '1' \\
- As a variable (STRING) for dynamically controlling the text output. \\
Example: strTextID \\
Sample assignment: PLC_PRG.strTextID := '1';
\end{tabular} \\
\hline "Tooltip index" & \begin{tabular}{l}
Text list ID. This refers to the desired output text. \\
- As fixed string with the ID in single straight quotation marks. \\
Example: '2' \\
- As a variable (STRING) for dynamically controlling the text output. \\
Example: strToolTipID \\
Sample assignment: PLC_PRG.strToolTipID := '2';
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{\wedge} \boldsymbol{y}\) Chapter 6.4.1.21.2.28 "Object 'Text List"' on page 2532

Element prop- The variables allow for dynamic control of the text display. erty 'Font variables'
\begin{tabular}{|c|c|}
\hline "Font name" & \begin{tabular}{l}
Variable (STRING). Includes the font of the text. \\
Example: PLC_PRG.stFontVar := 'Arial'; \\
The selection of fonts corresponds to the default "Font" dialog.
\end{tabular} \\
\hline "Size" & \begin{tabular}{l}
Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name. \\
- <pt>: Points (default) \\
Example: PLC_PRG.iFontHeight <pt> \\
Code: iFontHeight : INT := 12; \\
- <px> : Pixels \\
Example: PLC_PRG.iFontHeight <px> \\
Code: iFontHeight : INT := 19; \\
If you click in the value field, a drop-down list opens on the right for setting the unit. \\
Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property "Text property \(\rightarrow\) Font".
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Flags" & \begin{tabular}{l}
Variable (DWORD). Contains the flags for displaying fonts. \\
Flags: \\
- 1: Italics \\
- 2: Bold \\
- 4: Underline \\
- 8: Strikethrough \\
Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: PLC_PRG.dwFontType :=6;
\end{tabular} \\
\hline "Character set" & \begin{tabular}{l}
Variable (DWORD). Contains a character set number for the font. \\
The selection of character set numbers corresponds to the "Script" setting of the standard "Font" dialog.
\end{tabular} \\
\hline "Color" & \begin{tabular}{l}
Variable (DWORD). Includes the color of the text. \\
Example: PLC_PRG.dwColorFont:= 16\#FF000000;
\end{tabular} \\
\hline "Flags for text alignment" & \begin{tabular}{l}
Variable (integer data type). Contains the coding for text alignment. \\
Example: PLC_PRG.dwTextAlignment. \\
Coding: \\
- \(0:\) Top left \\
- 1: Horizontal center \\
- 2: Right \\
- 4: Vertical center \\
- 8: Bottom \\
Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: PLC_PRG.dwFontType := 5;
\end{tabular} \\
\hline
\end{tabular}

Fixed values for displaying texts are set in "Text properties".

See also
- « "Element property 'Text properties'" on page 3517

Element prop- The Element property is used as an interface for project variables to dynamically control colors erty 'Color varia- at runtime. bles'
\begin{tabular}{|c|c|}
\hline "Toggle color" & \begin{tabular}{l}
The property controls the toggled color at runtime. \\
Value assignment: \\
- FALSE: The element is displayed with the color specified in the "Color" property. \\
- TRUE: The element is displayed with the color specified in the "Alarm color" property. \\
Assigning the property: \\
- Placeholder for the user input variable \\
- "<toggle/tap variable>" \\
- "<NOT toggle/tap variable>" \\
The color change is not controlled by its own variable, but by a user input variable. \\
Note: Specify a variable for the mouse events "Tap" or "Toggle" in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both "Toggle" and "Tap", then the variable specified in "Tap" is used. \\
Hint: Click the symbol to insert the placeholder "<toggle/tap variable>". When you activate the "Inputconfiguration", "Tap FALSE" property, then the "<NOT toggle/tap variable>" placeholder is displayed. \\
- Instance path of a project variable (BOOL) \\
Example: PLC_PRG.xColorIsToggeled \\
Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.
\end{tabular} \\
\hline "Color" & \begin{tabular}{l}
Color variable for the Frame \\
- Variable (DWORD) for the color \\
Example: PLC_PRG.dwColor \\
- Color literal \\
Example of gray and opaque: 16\#FF888888 \\
Requirement: "Show Frame" property is activated. \\
Please note that the normal state is in effect if the expression in the "Colorvariables \(\boldsymbol{\rightarrow}\) Toggle color" property is not defined or it has the value FALSE.
\end{tabular} \\
\hline "Alarm color" & \begin{tabular}{l}
Color variable for the Frame in alarm state \\
- Variable (DWORD) for the alarm color \\
Example: PLC_PRG.dwAlarmColor \\
- Color literal \\
Example of red and opaque: 16\#FFFF0000 \\
Please note that the alarm state is in effect if the expression in the "Colorvariables \(\boldsymbol{\rightarrow}\) Toggle color" property has the value TRUE.
\end{tabular} \\
\hline
\end{tabular}

The transparency part of the color value is evaluated only if the "Activate semitransparent drawing" option of the visualization manager is selected.

See also
- « Chapter 6.4.5.10.4 "Animating a color display" on page 2914
- © Chapter 6.4.5.21.4.2 "Object 'Visualization manager'" on page 3398

Element prop- The variables control the element behavior dynamically. erty 'State variables'
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \(=\) FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

> The "Invisible" property is supported by the "Client Animation" functionality.

\section*{Element prop- \\ erty 'Button \\ state variable'}
\begin{tabular}{|c|c|}
\hline "Digital variable" & \begin{tabular}{l}
At runtime, the property controls whether the Button is displayed as pressed or not. \\
Values: \\
- FALSE: The Button is displayed as not pressed. \\
- tRUE: The Button is displayed as pressed. \\
Argument passed to the property: \\
- Placeholder for the user input variable to couple the representation of the Button with the input variable. \\
- "<toggle/tap variable>" \\
- "<NOT toggle/tap variable>" \\
Note: Specify a variable for the mouse events "Tap" or "Toggle" in the input configuration of the Button. Only then is the placeholder set. If you configure a variable in both "Toggle" and "Tap", then the variable specified in "Tap" is used. \\
Hint: Click the symbol to insert the placeholder "<toggle/tap variable>". When you activate the "Inputconfiguration", "Tap FALSE" property, then the "<NOT toggle/tap variable>" placeholder is displayed. \\
- Instance path of a project variable (BOOL) \\
Example: prgA.xButtonState \\
Note: Implement a value assignment in the code for the variable specified here.
\end{tabular} \\
\hline
\end{tabular}

\section*{Element property 'Image ID variable'}
\begin{tabular}{|l|l|}
\hline "Image ID" & \begin{tabular}{l} 
Variable (STRING). Contains the image ID. The contents of the string corre- \\
sponds to the description of the "Static ID" property. \\
Example: PLC_PRG.stImageID \(:=\) 'ImagePool_A.Image3'; \\
\hline
\end{tabular} \\
\hline
\end{tabular}

See also
- " \({ }^{2}\) Chapter 6.4.5.21.5.5 "Visualization Element 'Image'" on page 3463
- \({ }^{\bullet}\), Chapter 6.4.1.21.2.15 "Object 'Image Pool'" on page 2468

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain the configurations for the user input when using the mouse or keyboard. erty 'Input con- A user input defines an event and one or more actions that are executed when an event occurs. figuration'
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
The "Configure" button opens the "Input Configuration" dialog. There you can create or edit user inputs. \\
Configured user inputs are listed below the events. They each include the action that is triggered and the setting \\
in short form. \\
Example: "Execute ST Code":
\end{tabular} \\
\hline "OnDialogClosed" PRG.i_x \(:=0 ;\) \\
\hline "OnMouseClick" & Input event: The user closes the dialog. \\
\hline "OnMouseDown" & \begin{tabular}{l} 
Input event: The user clicks the mouse button completely in the element area. \\
The mouse button is clicked and released.
\end{tabular} \\
\hline "OnMouseEnter" & Input event: The user clicks down on the mouse button. \\
\hline "OnMouseLeave" & Input event: The user drags the mouse pointer to the element. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "OnMouseMove" & Input event: The user moves the mouse pointer over the element area. \\
\hline "OnMouseUp" & \begin{tabular}{l} 
Input events: \\
- \begin{tabular}{l} 
The user releases the mouse button within the element area. It is irrelevant \\
whether the user has previously pressed the mouse button inside or outside \\
the element area.
\end{tabular} \\
- The user presses the mouse button within the element area, leaves the \\
element area, and then releases the mouse button. \\
Note: This CODESYS-specific triggering behavior guarantees that actions for \\
key elements are completed. A key element starts an action for "OnMouseDown"" \\
and ends the action for "OnMouseUp". \\
Example: A visualization user presses the mouse button within the element area \\
of the key element and then moves the cursor position so that it lies outside the \\
element area. The action is ended anyway because "OnMouseUp" is triggered.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Tap" & \begin{tabular}{l} 
When a mouse click event occurs, the variable defined in "Variable" is described \\
in the application. The coding depends on the "Tap FALSE" and "Tap on enter if \\
captured" options.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL) that is set on mouse click event. \\
Example: PLC_PRG. bIsTapped \\
TRUE: A mouse click event exists. It lasts as long as the user presses the mouse \\
button over the element. It ends when the button is released. \\
FALSE: A mouse click event does not exist. \\
Requirement: The "Tap FALSE" option is not activated.
\end{tabular} \\
\hline "Tap FALSE" & \begin{tabular}{l} 
V: The mouse click event leads to a complementary value in "Variable". \\
TRUE: A mouse click event does not exist. \\
FALSE: While the mouse click event exists.
\end{tabular} \\
\hline "Tap on enter if captured" & \begin{tabular}{l} 
FA: During user input, it is also taken into consideration whether the mouse \\
pointer is dragged within the element area or not while the mouse button is \\
pressed. \\
TRUE: While the mouse click event exists and the mouse pointer is moved over \\
the element area. \\
FALSE: A mouse click event does not exist. Or the user moves the mouse \\
pointer outside of the element area while the mouse button is pressed. \\
The value is TRUE again as soon as the user moves the pointer back to the \\
element area. The mouse is then captured.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Toggle" & \begin{tabular}{l} 
With the onset of a mouse click event, the variable is set; when the mouse click \\
event is completed, the variable is reset.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL). Its value toggled when the mouse click event is ended. This is \\
when the user releases the mouse button while the mouse pointer is over the \\
element area. \\
If the user releases the mouse button while the mouse pointer is outside of the \\
element area, then the mouse click event is not ended and the value is not \\
toggled. \\
Hint: The user can cancel a started toggle input by dragging the mouse pointer \\
out of the element area.
\end{tabular} \\
\hline "Toggle on up if captured" & \begin{tabular}{l} 
V: The value toggles regardless of where the mouse pointer is when the mouse \\
button is released. The mouse is then captured.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Hotkey" & \begin{tabular}{l}
Keyboard shortcut on the element for triggering specific input actions. \\
When the keyboard shortcut event occurs, the input actions in the "Events" property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.
\end{tabular} \\
\hline "Key" & \begin{tabular}{l}
Key pressed for input action. \\
Example: [T] \\
Note: The following properties appear when a key is selected.
\end{tabular} \\
\hline "Events" & \begin{tabular}{l}
- "None" \\
- "Mouse down": Pressing the key triggers the input actions that are configured in the "OnMouseDown" property. \\
- "Mouse up": Releasing the key triggers the input actions that are configured in the "OnMouseUp" property. \\
- "Mouse down/up": Pressing and releasing the key triggers the input actions that are configured in the "OnMouseDown" property and the "OnMouseUp" property.
\end{tabular} \\
\hline "Shift" & Combination with the Shift key Example: [Shift]+[T]. \\
\hline "Control" & Combination with the Ctrl key Example: [Ctrl]+[T]. \\
\hline "Alt" & Combination with the Alt key Example: [AIt]+[T]. \\
\hline
\end{tabular}

All keyboard shortcuts and their actions that are configured in the visualization are listed on the "Keyboard Configuration" tab.

\section*{See also}
- ⓨ Chapter 6.4.5.21.2.2 "Command 'Keyboard Configuration'" on page 3341
- \(\xi^{2}\) Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration'" on page 3370

Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'
\begin{tabular}{|c|c|}
\hline "Access rights" & \begin{tabular}{l}
Opens the "Access rights" dialog. There you can edit the access privileges for the element. \\
Status messages: \\
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
- "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline \multicolumn{2}{|l|}{See also} \\
\hline & 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366 \\
\hline
\end{tabular}

\section*{Visualization Element 'Group Box'}

Symbol:


\section*{Category: "Common Controls"}

The element provides a visual grouping of visualization elements. The group box can have multiple levels of nesting.

You can also use drag\&drop to add elements to a "Group Box". To do this, drag the element to the window area of the "Group Box". The appearance of the cursor changes (a small plus sign is displayed). When you click the [Shift] key at the same time, the element is not added.

You can remove elements from the "Group Box" by dragging them out of the window area.

\section*{Element proper- \\ ties}
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Example: Parameter axis 1 \\
Optional \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list.
\end{tabular} \\
\hline "Type of element" & "Group Box" \\
\hline "Clipping" & : Elements that protrude beyond the size of the group box are clipped. \\
\hline
\end{tabular}

Element prop- The position defines the location and size of the element in the visualization window. These erty 'Position' are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline "X" & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (吊) to other positions in the editor.

See also
- Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center'
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline "Y" & \(Y\)-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus}\) ) to other positions in the editor.

Element prop- The properties contains character strings for labeling the element.
CODESYS accepts the specified texts automatically into the "GlobalTextList" text list. Therefore, these texts can be localized.
\begin{tabular}{|l|l|}
\hline "Text" & \begin{tabular}{l} 
Character string (without single straight quotation marks) for the labeling the \\
element. \\
Example: Axis 1.
\end{tabular} \\
\hline "Tooltip" & \begin{tabular}{l} 
Character string (without single straight quotation marks) that is displayed as the \\
tooltip of an element. \\
Example: Parameters of Axis 1.
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\xi^{4}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

Element prop- The properties contain fixed values for the text properties. erty 'Text properties'
\begin{tabular}{|l|l|}
\hline "Font" & \begin{tabular}{l} 
Example: "Default" \\
\(\ldots . .:\) The "Font" dialog box opens. \\
v: Drop-down list with style fonts.
\end{tabular} \\
\hline "Font color" & \begin{tabular}{l} 
Example: "Black" \\
\(\ldots . .:\) The "Color" dialog box opens. \\
v: Drop-down list with style colors.
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l} 
Whole number (value range from 0 to 255). This determines the transparency of \\
the respective color. \\
Example: \(255:\) The color is opaque. \\
\(0:\) The color is completely transparent. \\
Please note: If the color is a style color and already has a transparency value, \\
then this property is write-protected.
\end{tabular} \\
\hline
\end{tabular}

\section*{Element property 'Absolute movement'}

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|c|c|c|}
\hline \multicolumn{3}{|l|}{"Movement"} \\
\hline " \(\times\) " & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular}} \\
\hline " \(Y\) " & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular}} \\
\hline "Rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle1. \\
The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol. \\
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
\end{tabular} &  \\
\hline "Scaling" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (integer data type). Causes centric stretching. \\
Example: PLC_PRG.iScaling. \\
The reference point is the "Center" property. \\
The value 1 shrinks the element by a factor of 0.001 . The value 1000 returns the element to its original size.
\end{tabular}} \\
\hline "Interior rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle2. \\
In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. \\
The rotation point is shown as the \(\boldsymbol{\oplus}\) symbol. \\
Note: If a static angle of rotation is specified in the property "Position \(\rightarrow\) Angle", then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.
\end{tabular} &  \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

\section*{See also}

Element prop- The variables control the element behavior dynamically. erty 'State variables'
\begin{tabular}{|l|l|}
\hline "Invisible" & Variable (BOOL). Toggles the visibility of the element. \\
& \begin{tabular}{l} 
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{\boldsymbol{y}} \boldsymbol{y}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Table'}

\section*{Symbol:}


\section*{Category: "Common Controls"}

The element displays data that can be represented as an array in a table. Therefore, the data type of the visualizing variable can be 1) a one-dimensional array, 2) a maximum two-dimensional array, 3) an array of an array, 4) an array of structures, or 5) an array of a function block.

\section*{Element proper-}
ties
\begin{tabular}{|c|c|}
\hline "Element name" & \begin{tabular}{l}
Example: Data set component 1 \\
Optional \\
Hint: Assign individual names for elements so that they are found faster in the element list.
\end{tabular} \\
\hline "Type of element" & Table \\
\hline "Data array" & \begin{tabular}{l}
Array whose data is visualized as a table \\
Variable (ARRAY) whose data type determines the number of columns and rows in the table \\
Array types \\
- One-dimensional array: The table has one column. \\
- Two-dimensional array: The second dimension determines the number of columns. \\
- Array of an array: The number of array elements of the back array determines the number of columns. \\
- Array of a structure: The number of structure members determines the number of columns. \\
- Array of a function block: The number of local variables determines the number of columns. \\
Example: PLC_PRG.aiTable \\
Declaration: aiTable : ARRAY[0..3, 0..4] OF INT := [4(1, 2, 3, 4, 5) ]; \\
Hint: If the declaration of the array changes, then the table can be refreshed by placing the cursor in the data array value field and pressing the [Enter] key.
\end{tabular} \\
\hline "Max. array index" & \begin{tabular}{l}
Top index limit for the displayed table. Limits the number of displayed rows. The index begins at 0 . \\
- Variable (integer data type) \\
Example: PLC_PRG.iUpperIndexBoundToDisplay \\
- Integer literal \\
Example: 4 is displayed as 5 in the row of the table.
\end{tabular} \\
\hline
\end{tabular}

See also
- Data Type 'ARRAY'

Element property 'Columns'

The "Table" element shows the values of a variable in a tabular view. The array elements of structure members are shown in a column or in a row. Two-dimensional arrays or arrays of a structure are shown in multiple columns. The visualized variable is defined in the "Data array" property. When a variable is assigned there, you can specify the display of the Table columns where the array elements are shown. An individual configuration is possible for each column that is assigned to an index [ \(<n>]\).
\begin{tabular}{|l|l|}
\hline "Show row header" & \(\boxed{\checkmark}\) : The row header is visible. \\
& Example: For an array, the index of the array element is displayed in the header. \\
\hline "Show column header" & \(\boxed{\text { : }}\) : The column label is visible. \\
\hline "Row height" & Height of the rows (in pixels) \\
\hline "Row header width" & Width of the row label \\
\hline "Scroll bar size" " & Size of the scroll bar (in pixels) \\
\hline
\end{tabular}

Table 719: "Element property 'Columns: Column [<n>]""
\begin{tabular}{|c|c|}
\hline "Column header" & \begin{tabular}{l}
By default, the name of the array or structure is applied as the heading with the index or structure member for the column. If an array of a function block has been selected for "Data array", then the name of the array is applied to the column header with the local variables of the function block that belong to the column. \\
The column label can be changed here by specifying a new title.
\end{tabular} \\
\hline "Width" & Column width (in pixels) \\
\hline "Image column" & Images can be displayed in the column. Images are used from the global image pool or custom image pools. The image IDs are shown in the cells of the Table as they are defined in the image pool. \\
\hline \multicolumn{2}{|l|}{"Image configuration"} \\
\hline "Fill mode" & \begin{tabular}{l}
- Fill cell: The image resizes to the dimensions of the cell without fixing the height/width ratio. \\
- Centered: The image is centered in the cell and retains its proportions (height/width ratio).
\end{tabular} \\
\hline "Transparency" & The color which is specified in "Transparent color" is displayed as transparent. \\
\hline "Transparent color" & \begin{tabular}{l}
This color is displayed as transparent. \\
Requirement: The "Transparency" property is activated.
\end{tabular} \\
\hline "Text alignment of header" & \begin{tabular}{l}
Alignment of the column header: \\
- Left \\
- Centered \\
- Right
\end{tabular} \\
\hline "Use template" & Another visualization element (type "Rectangle", "Rounded Rectangle", or "Ellipse") is inserted into each line of this Table column. The properties list is extended automatically with the properties of this element in "Template". \\
\hline "Text alignment of the headline from the template" & \begin{tabular}{l}
Requirement: The "Use template" property is activated. \\
When activated, the settings for font (size) and alignment in the inserted template are also applied to the column header.
\end{tabular} \\
\hline "Template" & \begin{tabular}{l}
Requirement: The "Use template" property is activated. \\
The properties of all elements assigned to the column are listed in "Template". They can be modified there as described in "Rectangle", "Rounded Rectangle", and "Ellipse".
\end{tabular} \\
\hline
\end{tabular}

See also
- \#Chapter 6.4.5.21.5.1 "Visualization Element 'Rectangle', 'Rounded Rectangle', 'Ellipse'" on page 3413

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline "X" & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline " \(Y\) " & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols ( ( - ) to other positions in the editor.

See also
- Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center' rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline "X" & X-coordinate of the point of rotation \\
\hline " \(Y\) " & \(Y\)-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus})\) to other positions in the editor.

Element prop- The properties contain fixed values for the text properties.
erty 'Text prop-
erties'
\begin{tabular}{|l|l|}
\hline "Horizontal alignment" & Horizontal alignment of the text within the element. \\
\hline "Vertical alignment" & Vertical alignment of the text within the element. \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Font" & Example: "Default"
The "Font" dialog box opens.
Drop-down list with style fonts. \\
\hline "Font color" & \begin{tabular}{l}
Example: "Black"
The "Color" dialog box opens. \\
- : Drop-down list with style colors.
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l}
Whole number (value range from 0 to 255). This determines the transparency of the respective color. \\
Example: 255: The color is opaque. \\
0 : The color is completely transparent. \\
Please note: If the color is a style color and already has a transparency value, then this property is write-protected.
\end{tabular} \\
\hline
\end{tabular}

Element prop- Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is erty 'Dynamic currently indexed in the variable. texts'
\begin{tabular}{|c|c|}
\hline "Text list" & \begin{tabular}{l}
Variable (string) or name of the text list as a fixed string in single straight quotation marks. \\
Example: 'Errorlist' \\
- : Drop-down list with the dialogs available in the text lists.
\end{tabular} \\
\hline "Text index" & \begin{tabular}{l}
Text list ID. This refers to the desired output text. \\
- As fixed string with the ID in single straight quotation marks. Example: '1' \\
- As a variable (STRING) for dynamically controlling the text output. \\
Example: strTextID \\
Sample assignment: PLC_PRG.strTextID := '1';
\end{tabular} \\
\hline "Tooltip index" & \begin{tabular}{l}
Text list ID. This refers to the desired output text. \\
- As fixed string with the ID in single straight quotation marks. Example: '2' \\
- As a variable (STRING) for dynamically controlling the text output. \\
Example: strToolTipID \\
Sample assignment: PLC_PRG.strToolTipID := '2';
\end{tabular} \\
\hline
\end{tabular}

\section*{See also}
- \& Chapter 6.4.1.21.2.28 "Object 'Text List"' on page 2532

Element prop- The variables enable dynamic control of the text display. erty 'Font variables'
\begin{tabular}{|l|l|}
\hline "Font name" & \begin{tabular}{l} 
Variable (STRING). Includes the font of the text. \\
Example: PLC_PRG.stFontVar \(:=~ ' A r i a l ' ~ ; ~\)
\end{tabular} \\
The selection of fonts corresponds to the default "Font" dialog box.
\end{tabular}\(|\)\begin{tabular}{ll} 
VSize" & \begin{tabular}{l} 
Variable (integer data type). Contains the font size (in pixels). \\
The selection of font sizes corresponds to the default "Font" dialog box.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Flags" & \begin{tabular}{l}
Variable (DWORD). Contains the flags for displaying fonts. \\
Flags: \\
- 1: Italics \\
- 2: Bold \\
- 4: Underline \\
- 8: Strikethrough \\
Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: PLC_PRG.dwFontType :=6;
\end{tabular} \\
\hline "Charset" & \begin{tabular}{l}
Variable (DWORD). Contains a character set number for the font. \\
The selection of character set numbers corresponds to the "Script" setting of the standard "Font" dialog box.
\end{tabular} \\
\hline "Color" & \begin{tabular}{l}
Variable (DWORD). Includes the color of the text. \\
Example: PLC_PRG.dwColorFont:= 16\#FF000000;
\end{tabular} \\
\hline "Flags for text alignment" & \begin{tabular}{l}
Variable (integer data type). Contains the coding for text alignment. \\
Example: PLC_PRG.dwTextAlignment. \\
Coding: \\
- 0 : Top left \\
- 1: Horizontal center \\
- 2: Right \\
- 4: Vertical center \\
- 8: Bottom \\
Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: PLC_PRG.dwFontType := 5;
\end{tabular} \\
\hline
\end{tabular}

Fixed values for displaying texts are set in "Text properties".

\section*{See also}
- E"Element property 'Text properties"' on page 3541

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" \\
\hline "X" & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\oplus\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
"Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\boldsymbol{\oplus}\) symbol.
\end{tabular} \\
\begin{tabular}{l} 
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular}
\end{tabular}

You can link the variables to a unit conversion.
"Client Animation" functionality.

Element prop- The variables control the element behavior dynamically. erty 'State varia-
bles'
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \(=\) FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

Element prop-
erty 'Selection'
\begin{tabular}{|c|c|}
\hline "Background color on selection" & Fill color of the selected row. \\
\hline "Selection font color" & Font color of the selected row. \\
\hline "Selection type" & \begin{tabular}{l}
Selection when clicking the table row. \\
- No selection: No selection \\
- Cell selection: The clicked cell only. \\
- Row selection: Row of the clicked cell. \\
- Column selection: Column of the clicked cell. \\
- Row and column selection: Row and column of the clicked cell.
\end{tabular} \\
\hline "Frame around selected cells" & \(\checkmark\) : A frame is drawn around the selected cells. \\
\hline "Variable for selected column" & \begin{tabular}{l}
Variable (INT). Contains the array index of the "Column" of the selected cell. If the data array points to a structure, then the structure components are indexed, starting at 0 . \\
Warning: This index represents the correct position in the array only if no columns have been removed from the table in the display.
\end{tabular} \\
\hline "Variable for selected row" & Variable (INT). Contains the array index of the "Row" of the selected cell. \\
\hline "Variable for valid column selection" & \begin{tabular}{l}
Variable (BOOL). \\
TRUE: The "Variable for selected column" variable contains a valid value.
\end{tabular} \\
\hline "Variable for valid row selection" & \begin{tabular}{l}
Variable (BOOL). \\
TRUE: The "Variable for selected row" variable contains a valid value.
\end{tabular} \\
\hline
\end{tabular}

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- Requirement: User management is set up for the visualization. erty 'Access
rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages:
\end{tabular} \\
& \begin{tabular}{l} 
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
\(\bullet\) \\
"Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{\wedge} \Rightarrow\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

See also

- 乡y Chapter 6.4.5.11.2 "Displaying Array Variables in Tables" on page 2917
- Data Type 'ARRAY'

\section*{Visualization Element 'Text Field'}

Symbol:

Category: "Common Controls"

The element is used for the following purposes:
- Static output of text. The contents of a variable can be part of the text.
- Showing a tooltip. The text is managed as static text and can also be defined so that the contents of a variable are also displayed.
- Dynamic output of text. Texts of a text list are displayed dynamically.
- Input of text. For example, a user can input a number or a text literal.

See also
- \({ }^{4}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Element proper- \\ ties}
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Optional \\
Example: FileName_A \\
\\
Hint: Assign individual names for elements so that they are found faster in the \\
element list.
\end{tabular} \\
\hline "Type of element" & "Text Field" \\
\hline
\end{tabular}

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X "\) & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (ए-) to other positions in the editor.

See also
- \({ }^{\circ}\) ) Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

\section*{Element property 'Colors'}
\begin{tabular}{|l|l|}
\hline "Normal state" & \begin{tabular}{l} 
The normal state is in effect if the variable in "Color variables \(\rightarrow\) Toggle color" is \\
not defined or it has the value FALSE.
\end{tabular} \\
\hline "Frame color" & Frame and fill color for the corresponding state of the variable. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Fill color" & \\
\hline "Transparency" & \begin{tabular}{l} 
Transparency value (0 to 255) for defining the transparency of the selected color. \\
Example: 255: The color is opaque. \(0:\) The color is completely transparent.
\end{tabular} \\
\hline "Alarm state" & \begin{tabular}{l} 
The alarm state is in effect if the variable in "Color variables \(\rightarrow\) Toggle color" has \\
the value TRUE.
\end{tabular} \\
\hline
\end{tabular}

See also
- « Chapter 6.4.5.21.3.5 "Dialog 'Gradient Editor'" on page 3369

Element prop- The properties contain fixed values for setting the look of the element. erty 'Appearance'
\begin{tabular}{|c|c|}
\hline "Line width" & \begin{tabular}{l}
Value in pixels \\
Example: 2 \\
Note: The values 0 and 1 both result in a line weight of 1 pixel. If no line should be displayed, then the "Line style" property must be set to the option "Invisible".
\end{tabular} \\
\hline "Fill attributes" & \begin{tabular}{l}
The way in which the element is filled. \\
- "Filled":The element is filled with the color from property "Colors \(\rightarrow\) Fill color". \\
- "Invisible": The fill color is invisible.
\end{tabular} \\
\hline "Line style" & \begin{tabular}{l}
Type of line representation \\
- "Solid" \\
- "Dashes" \\
- "Dots" \\
- "Dash Dot" \\
- "Dash Dot Dot" \\
- "not visible"
\end{tabular} \\
\hline
\end{tabular}

You can assign variables in the "Appearance variables" property for controlling the appearance dynamically. The fixed values here are overwritten.

\section*{See also}
- "Element property 'Appearance variables'" on page 3489

Element property 'Texts'

The properties contains character strings for labeling the element. The character string can also contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.
CODESYS accepts the specified texts automatically into the "GlobalTextList" text list. Therefore, these texts can be localized.
\begin{tabular}{|l|l|}
\hline "Text" & \begin{tabular}{l} 
Character string (without single straight quotation marks) for the labeling the \\
element. Add a line break by pressing the keyboard shortcut [Ctrl] + [Enter]. \\
Example: Accesses : \%i \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Text".
\end{tabular} \\
\hline "Tooltip" & \begin{tabular}{l} 
Character string (without single straight quotation marks) that is displayed as the \\
tooltip of an element. \\
Example: Number of valid accesses. \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Tooltip".
\end{tabular} \\
\hline
\end{tabular}

See also
- をy "Element property 'Text variables'" on page 3541
- \(\xi^{\star}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872
- \({ }^{4}\) Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329

Element prop- The properties contain fixed values for the text properties.
erty 'Text prop-
erties'
\begin{tabular}{|l|l|}
\hline "Horizontal alignment" & Horizontal alignment of the text within the element. \\
\hline "Vertical alignment" & Vertical alignment of the text within the element. \\
\hline "Text format" & \begin{tabular}{l} 
Definition for displaying texts that are too long \\
- "Default": The long text is truncated. \\
- "Line break": The text is split into parts. \\
- "Ellipsis": The visible text ends with "..." indicating that it is not complete.
\end{tabular} \\
\hline "Font" & \begin{tabular}{l} 
Example: "Default" \\
...: The "Font" dialog box opens. \\
-: Drop-down list with style fonts.
\end{tabular} \\
\hline "Font color" & \begin{tabular}{l} 
Example: "Black" \\
(..: The "Color" dialog box opens. \\
v: Drop-down list with style colors.
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l} 
Whole number (value range from 0 to 255). This determines the transparency of \\
the respective color. \\
Example: \(255:\) The color is opaque. \\
\(0:\) The color is completely transparent.
\end{tabular} \\
\hline
\end{tabular}

\footnotetext{
Element prop- These properties are variables with contents that replace a format definition. erty 'Text variables'
}
\begin{tabular}{|l|l|}
\hline "Text variable" & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses
\end{tabular} \\
& \begin{tabular}{l} 
Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Text". \\
Note: If you specify a variable of type enumeration with text list support, then \\
the name of the enumeration data type is added automatically in angle brackets \\
after the variable name. Example: PLC_PRG.enVar <enumeration name>. \\
Then the symbolic value of the enumeration component is printed instead of the \\
numeric value when text is printed. Refer to the help page for the enumerations.
\end{tabular} \\
\hline "Tooltip variable" & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses InTooltip \\
Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Tooltip".
\end{tabular} \\
\hline
\end{tabular}

See also
- Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329
- © "Element property 'Texts"' on page 3540
- E Chapter 6.4.1.20.5.18 "Enumerations" on page 2263

Element prop- Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is erty 'Dynamic currently indexed in the variable. texts'
\begin{tabular}{|c|c|}
\hline "Text list" & \begin{tabular}{l}
Variable (string) or name of the text list as a fixed string in single straight quotation marks. \\
Example: 'Errorlist' \\
: Drop-down list with the dialogs available in the text lists.
\end{tabular} \\
\hline "Text index" & \begin{tabular}{l}
Text list ID. This refers to the desired output text. \\
- As fixed string with the ID in single straight quotation marks. Example: '1' \\
- As a variable (STRING) for dynamically controlling the text output. Example: strTextID \\
Sample assignment: PLC_PRG.strTextID := '1';
\end{tabular} \\
\hline "Tooltip index" & \begin{tabular}{l}
Text list ID. This refers to the desired output text. \\
- As fixed string with the ID in single straight quotation marks. \\
Example: ' 2 ' \\
- As a variable (STRING) for dynamically controlling the text output. \\
Example: strToolTipID \\
Sample assignment: PLC_PRG.strToolTipID := '2';
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\xi^{*}\) Chapter 6.4.1.21.2.28 "Object 'Text List"" on page 2532

\footnotetext{
Element prop- The variables allow for dynamic control of the text display. erty 'Font variables'
}
\begin{tabular}{|c|c|}
\hline "Font name" & \begin{tabular}{l}
Variable (STRING). Includes the font of the text. \\
Example: PLC_PRG.stFontVar := 'Arial'; \\
The selection of fonts corresponds to the default "Font" dialog.
\end{tabular} \\
\hline "Size" & \begin{tabular}{l}
Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name. \\
- <pt>: Points (default) \\
Example: PLC_PRG.iFontHeight <pt> \\
Code: iFontHeight : INT := 12; \\
- <px>: Pixels \\
Example: PLC_PRG.iFontHeight <px> \\
Code: iFontHeight : INT := 19; \\
If you click in the value field, a drop-down list opens on the right for setting the unit. \\
Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property "Text property \(\rightarrow\) Font".
\end{tabular} \\
\hline "Flags" & \begin{tabular}{l}
Variable (DWORD). Contains the flags for displaying fonts. \\
Flags: \\
- 1: Italics \\
- 2: Bold \\
- 4: Underline \\
- 8: Strikethrough \\
Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: PLC_PRG.dwFontType \(:=6\);
\end{tabular} \\
\hline "Character set" & \begin{tabular}{l}
Variable (DWORD). Contains a character set number for the font. \\
The selection of character set numbers corresponds to the "Script" setting of the standard "Font" dialog.
\end{tabular} \\
\hline "Color" & \begin{tabular}{l}
Variable (DWORD). Includes the color of the text. \\
Example: PLC_PRG.dwColorFont:= 16\#FF000000;
\end{tabular} \\
\hline "Flags for text alignment" & \begin{tabular}{l}
Variable (integer data type). Contains the coding for text alignment. \\
Example: PLC_PRG.dwTextAlignment. \\
Coding: \\
- 0 : Top left \\
- 1: Horizontal center \\
- 2: Right \\
- 4: Vertical center \\
- 8: Bottom \\
Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: PLC_PRG.dwFontType := 5;
\end{tabular} \\
\hline
\end{tabular}

Fixed values for displaying texts are set in "Text properties".

See also
- * "Element property 'Text properties"' on page 3541

Element prop- The Element property is used as an interface for project variables to dynamically control colors erty 'Color varia- at runtime. bles'
\begin{tabular}{|c|c|}
\hline "Toggle color" & \begin{tabular}{l}
The property controls the toggled color at runtime. \\
Value assignment: \\
- FALSE: The element is displayed with the color specified in the "Color" property. \\
- TRUE: The element is displayed with the color specified in the "Alarm color" property. \\
Assignment options: \\
- Placeholder for the user input variable \\
- "<toggle/tap variable>" \\
- "<NOT toggle/tap variable>" \\
The color change is not controlled by its own variable, but by a user input variable. \\
Note: Specify a variable for the mouse events "Tap" or "Toggle" in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both "Toggle" and "Tap", then the variable specified in "Tap" is used. \\
Hint: Click the symbol to insert the placeholder "<toggle/tap variable>". When you activate the "Inputconfiguration", "Tap FALSE" property, then the "<NOT toggle/tap variable>" placeholder is displayed. \\
- Instance path of a project variable (BOOL) \\
Example: PLC_PRG.xColorIsToggeled \\
Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.
\end{tabular} \\
\hline \begin{tabular}{l}
"Normal state" \\
"Alarm state"
\end{tabular} & The properties listed below control the color depending on the state. The normal state is in effect if the variable in "Color variables", "Toggle color" is not defined or it has the value FALSE. The alarm state is in effect if the variable in "Colorvariables", "Toggle color" has the value TRUE. \\
\hline "Frame color" & \begin{tabular}{l}
Assignment options: \\
- Variable (DWORD) for the frame color Example: PLC_PRG.dwBorderColor \\
- Color literal Example of green and opaque: 16\#FF00FF00
\end{tabular} \\
\hline "Filling color" & \begin{tabular}{l}
Assignment options: \\
- Variable (DWORD) for the fill color \\
Example: PLC_PRG.dwFillColor \\
- Color literal \\
Example of gray and opaque: \(16 \#\) FF888888
\end{tabular} \\
\hline
\end{tabular}

The transparency part of the color value is evaluated only if the "Activate semitransparent drawing" option of the visualization manager is selected.

Select the "Advanced" option in the toolbar of the properties view. Then all element properties are visible.

See also
- \({ }^{4}\) Chapter 6.4.5.10.4 "Animating a color display" on page 2914

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|c|c|c|}
\hline \multicolumn{3}{|l|}{"Movement"} \\
\hline " \(X\) " & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the \(X\) position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular}} \\
\hline "Y" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular}} \\
\hline "Rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle1. \\
The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol. \\
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
\end{tabular} &  \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\boldsymbol{\oplus}\) symbol.
\end{tabular} \\
\begin{tabular}{l} 
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

The "X", " \(Y\) ", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also
-

Element prop- The variables control the element behavior dynamically.
erty 'State varia-
bles'
\begin{tabular}{|l|l|}
\hline "Invisible" & Variable (BOOL). Toggles the visibility of the element. \\
& \begin{tabular}{l} 
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \(=\) FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Selection and caret configuration'

The "Invisible" property is supported by the "Client Animation" functionality.


The variables allow for controlling the caret position and the selection of the text.
\begin{tabular}{|l|l|}
\hline "Caret position" & Variable (integer data type). Contains the position of the cursor. \\
\hline "Selection start" & \begin{tabular}{l} 
Variable (integer data type). Contains the position of the first selected character. \\
Example: PLC_PRG. iselStart
\end{tabular} \\
\hline "Selection end" & \begin{tabular}{l} 
Variable (integer data type). Contains the position of the last selected character. \\
Example: PLC_PRG. iSelEnd
\end{tabular} \\
\hline "All selected" & \begin{tabular}{l} 
Variable (BOOL). Toggles the selection of the entered text. \\
TRUE: The text in the text field is selected. \\
FALSE: The selection starts with the value in "Selection start" and ends with \\
"Selection end".
\end{tabular} \\
\hline
\end{tabular}

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain the configurations for the user input when using the mouse or keyboard. erty 'Input con- A user input defines an event and one or more actions that are executed when an event occurs. figuration'

The "Configure" button opens the "Input Configuration" dialog. There you can create or edit user inputs.
Configured user inputs are listed below the events. They each include the action that is triggered and the setting in short form.
Example: "Execute ST Code": ЗPLC_PRG.i_x := 0;
\begin{tabular}{|l|l} 
"OnDialogClosed" & Input event: The user closes the dialog.
\end{tabular}
\begin{tabular}{|l|l|}
\hline "OnMouseClick" & \begin{tabular}{l} 
Input event: The user clicks the mouse button completely in the element area. \\
The mouse button is clicked and released.
\end{tabular} \\
\hline "OnMouseDown" & Input event: The user clicks down on the mouse button. \\
\hline "OnMouseEnter" & Input event: The user drags the mouse pointer to the element. \\
\hline "OnMouseLeave" & Input event: The user drags the mouse pointer away from the element. \\
\hline "OnMouseMove" & \begin{tabular}{l} 
Input event: The user moves the mouse pointer over the element area.
\end{tabular} \\
\hline "OnMouseUp" & \begin{tabular}{l} 
Input events: \\
The user releases the mouse button within the element area. It is irrelevant \\
whether the user has previously pressed the mouse button inside or outside \\
the element area. \\
- The user presses the mouse button within the element area, leaves the \\
element area, and then releases the mouse button. \\
Note: This CODESYS-specific triggering behavior guarantees that actions for \\
key elements are completed. A key element starts an action for "OnMouseDown" \\
and ends the action for "OnMouseUp". \\
Example: A visualization user presses the mouse button within the element area \\
of the key element and then moves the cursor position so that it lies outside the \\
element area. The action is ended anyway because "OnMouseUp" is triggered.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Tap" & \begin{tabular}{l} 
When a mouse click event occurs, the variable defined in "Variable" is described \\
in the application. The coding depends on the "Tap FALSE" and "Tap on enter if \\
captured" options.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL) that is set on mouse click event. \\
Example: PLC_PRG. bIsTapped \\
TRUE: A mouse click event exists. It lasts as long as the user presses the mouse \\
button over the element. It ends when the button is released. \\
FALSE: A mouse click event does not exist. \\
Requirement: The "Tap FALSE" option is not activated.
\end{tabular} \\
\hline "Tap FALSE" & \begin{tabular}{l} 
Q: The mouse click event leads to a complementary value in "Variable". \\
TRUE: A mouse click event does not exist. \\
FALSE: While the mouse click event exists.
\end{tabular} \\
\hline "Tap on enter if captured" & \begin{tabular}{l} 
V: During user input, it is also taken into consideration whether the mouse \\
pointer is dragged within the element area or not while the mouse button is \\
pressed.
\end{tabular} \\
\begin{tabular}{l} 
TRUE: While the mouse click event exists and the mouse pointer is moved over \\
the element area. \\
FALSE: A mouse click event does not exist. Or the user moves the mouse \\
pointer outside of the element area while the mouse button is pressed. \\
The value is TRUE again as soon as the user moves the pointer back to the \\
element area. The mouse is then captured.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Toggle" & \begin{tabular}{l} 
With the onset of a mouse click event, the variable is set; when the mouse click \\
event is completed, the variable is reset.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL). Its value toggled when the mouse click event is ended. This is \\
when the user releases the mouse button while the mouse pointer is over the \\
element area. \\
If the user releases the mouse button while the mouse pointer is outside of the \\
element area, then the mouse click event is not ended and the value is not \\
toggled. \\
Hint: The user can cancel a started toggle input by dragging the mouse pointer \\
out of the element area.
\end{tabular} \\
\hline "Toggle on up if captured" & \begin{tabular}{l} 
V: The value toggles regardless of where the mouse pointer is when the mouse \\
button is released. The mouse is then captured.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Hotkey" & \begin{tabular}{l}
Keyboard shortcut on the element for triggering specific input actions. \\
When the keyboard shortcut event occurs, the input actions in the "Events" property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.
\end{tabular} \\
\hline "Key" & \begin{tabular}{l}
Key pressed for input action. \\
Example: [T] \\
Note: The following properties appear when a key is selected.
\end{tabular} \\
\hline "Events" & \begin{tabular}{l}
- "None" \\
- "Mouse down": Pressing the key triggers the input actions that are configured in the "OnMouseDown" property. \\
- "Mouse up": Releasing the key triggers the input actions that are configured in the "OnMouseUp" property. \\
- "Mouse down/up": Pressing and releasing the key triggers the input actions that are configured in the "OnMouseDown" property and the "OnMouseUp" property.
\end{tabular} \\
\hline "Shift" & Combination with the Shift key Example: [Shift]+[T]. \\
\hline "Control" & Combination with the Ctrl key Example: [Ctrl]+[T]. \\
\hline "Alt" & Combination with the Alt key Example: [AIt]+[T]. \\
\hline
\end{tabular}


All keyboard shortcuts and their actions that are configured in the visualization are listed on the "Keyboard Configuration" tab.

See also
- « Chapter 6.4.5.21.2.2 "Command 'Keyboard Configuration"' on page 3341
- \& Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration"' on page 3370

Element property 'Access
rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages: \\
\\
\\
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
- "Rights are set: Limited rights": Access is restricted for at least one group. \\
\hline
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\sum^{*}\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

\section*{Visualization Element 'Scroll Bar'}

Symbol:


Category: "Common Controls"
The element sets the value of a variable, depending on the position of the scroll bar.

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & Example: Speed Conveyor Belt 1 \\
Optional \\
& \begin{tabular}{l} 
Hint: Assign individual names for elements so that they are found faster in the \\
element list.
\end{tabular} \\
\hline "Type of element" & "Scroll Bar" \\
\hline
\end{tabular}

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center' rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline "Y" & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus})\) to other positions in the editor.
\begin{tabular}{|l|l|}
\hline "Value" & Variable as type integer that includes the position of the scroll bar. \\
\hline "Minimum value" & Smallest value of the scroll bar (fixed value or variable). \\
\hline "Maximum value" & Largest value of the scroll bar (fixed value or variable). \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Page size" & Page size \\
& - As a fixed value, for example 10 \\
& - As a variable of data type integer \\
Requirement: Visible when the "Move to click" property is not selected. \\
\hline "Move to click" & \begin{tabular}{l} 
Behavior of the scroll bar at visualization runtime when it is clicked: \\
\\
\\
\\
\\
\\
: The scrollbar moves to the clicked position. \\
\(\square\)
\end{tabular} \\
\hline
\end{tabular} erty 'Position'

Element prop- The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x -axis runs to the right. The positive vertical y -axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X\) " & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (ए) (\%) other positions in the editor.

\section*{See also}
- \({ }^{*}\) ) Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The property defines the representation of scaling and direction of travel. erty 'Bar'
\begin{tabular}{|l|l|}
\hline "Orientation" & \begin{tabular}{l} 
Alignment of the slider; defined by the ratio of width to height. \\
- "Horizontal" \\
- "Vertical" \\
You can modify the alignment in the visualization editor by using the pointing \\
device to adjust the width and height of the Scroll Bar.
\end{tabular} \\
\hline "Running direction" & \begin{tabular}{l} 
The drop-down list varies depending on the alignment of the slider. \\
Horizontal \\
- "Left to right": Scale starts at the left. \\
- "Right to left": Scale starts at the right. \\
Vertical \\
- "Bottom to top": Scale starts at the bottom. \\
- "Top to bottom": Scale starts at the top.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain fixed values for setting colors. erty 'Colors'
\begin{tabular}{|l|l|}
\hline "Color" & \begin{tabular}{l} 
Color for the element in its normal state. \\
Please note that the normal state is in effect if the expression in the \\
"Color variables \(\rightarrow\) Toggle color" property is not defined or it has the value \\
FALSE.
\end{tabular} \\
\hline "Alarm color" & \begin{tabular}{l} 
Color for the element in alarm state. \\
Please note that the alarm state is in effect if the expression in the \\
"Color variables \(\rightarrow\) Toggle color" property has the value TRUE.
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l} 
Value (0 to 255) for defining the transparency of the selected color. \\
Example 255: The color is opaque. 0 : The color is completely transparent.
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{\text {a }}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

Element prop- The properties contains character strings for labeling the element. The character string can also erty 'Texts' contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.
CODESYS accepts the specified texts automatically into the "GlobalTextList" text list. Therefore, these texts can be localized.
\begin{tabular}{|l|l|}
\hline "Text" & \begin{tabular}{l} 
Character string (without single straight quotation marks) for the labeling the \\
element. Add a line break by pressing the keyboard shortcut [CtrI] + [Enter]. \\
Example: Accesses: \%i \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Text".
\end{tabular} \\
\hline "Tooltip" & \begin{tabular}{l} 
Character string (without single straight quotation marks) that is displayed as the \\
tooltip of an element. \\
Example: Number of valid accesses. \\
The variable that contains the current value for the placeholder is specified in the \\
property "Text variable \(\rightarrow\) Tooltip".
\end{tabular} \\
\hline
\end{tabular}

See also
- \(乡\) "Element property 'Text variables'" on page 3553
- « Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872
- \({ }^{\text {• }}\) Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329

Element prop- The properties contain fixed values for the text properties. erty 'Text properties'
\begin{tabular}{|l|l|}
\hline "Horizontal alignment" & Horizontal alignment of the text within the element. \\
\hline "Vertical alignment" & Vertical alignment of the text within the element. \\
\hline "Font" & \begin{tabular}{l} 
Example: "Default" \\
\(\ldots .\). \\
\hline v: The "Font" dialog box opens. \\
vrop-down list with style fonts.
\end{tabular} \\
\hline "Font color" & \begin{tabular}{l} 
Example: "Black" \\
\hline\(\ldots .:\) The "Color" dialog box opens. \\
v: Drop-down list with style colors.
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l} 
Whole number (value range from 0 to 255). This determines the transparency of \\
the respective color. \\
Example: \(255:\) The color is opaque. \\
\(0:\) The color is completely transparent. \\
Please note: If the color is a style color and already has a transparency value, \\
then this property is write-protected.
\end{tabular} \\
\hline
\end{tabular}

Element prop- These properties are variables with contents that replace a format definition. erty 'Text variables'
\begin{tabular}{|l|l|}
\hline "Text variable" & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses
\end{tabular} \\
& Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Text". \\
& \begin{tabular}{l} 
Note: If you specify a variable of type enumeration with text list support, then \\
the name of the enumeration data type is added automatically in angle brackets \\
after the variable name. Example: PLC_PRG. enVar <enumeration name>. \\
Then the symbolic value of the enumeration component is printed instead of the \\
numeric value when text is printed. Refer to the help page for the enumerations.
\end{tabular} \\
\hline "Tooltip variable" & \begin{tabular}{l} 
Variable (data type compliant with the format definition). It contains what is \\
printed instead of the format definition. \\
Example: PLC_PRG. iAccesses InTooltip \\
Note: The format definition is part of the text in the property "Texts \(\rightarrow\) Tooltip".
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\quad \stackrel{y}{c}\) Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329
- \(\quad\) " "Element property 'Texts'" on page 3552
- \({ }^{\bullet}>\) Chapter 6.4.1.20.5.18 "Enumerations" on page 2263

Element prop- Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is erty 'Dynamic texts' currently indexed in the variable.
\begin{tabular}{|l|l|}
\hline "Text list" & \begin{tabular}{l} 
Variable (string) or name of the text list as a fixed string in single straight quota- \\
tion marks. \\
Example: ' Errorlist'
\end{tabular} \\
& : Drop-down list with the dialogs available in the text lists.
\end{tabular}

See also
- \({ }^{\text {• }}\) Chapter 6.4.1.21.2.28 "Object 'Text List"' on page 2532

Element prop- The variables allow for dynamic control of the text display. erty 'Font variables'
\begin{tabular}{|c|c|}
\hline "Font name" & \begin{tabular}{l}
Variable (STRING). Includes the font of the text. \\
Example: PLC_PRG.stFontVar := 'Arial'; \\
The selection of fonts corresponds to the default "Font" dialog.
\end{tabular} \\
\hline "Size" & \begin{tabular}{l}
Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name. \\
- <pt>: Points (default) \\
Example: PLC_PRG.iFontHeight <pt> \\
Code: iFontHeight : INT := 12; \\
- <px> : Pixels \\
Example: PLC_PRG.iFontHeight <px> \\
Code: iFontHeight : INT := 19; \\
If you click in the value field, a drop-down list opens on the right for setting the unit. \\
Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property "Text property \(\rightarrow\) Font".
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Flags" & \begin{tabular}{l}
Variable (DWORD). Contains the flags for displaying fonts. \\
Flags: \\
- 1: Italics \\
- 2: Bold \\
- 4: Underline \\
- 8: Strikethrough \\
Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: PLC_PRG.dwFontType :=6;
\end{tabular} \\
\hline "Character set" & \begin{tabular}{l}
Variable (DWORD). Contains a character set number for the font. \\
The selection of character set numbers corresponds to the "Script" setting of the standard "Font" dialog.
\end{tabular} \\
\hline "Color" & \begin{tabular}{l}
Variable (DWORD). Includes the color of the text. \\
Example: PLC_PRG.dwColorFont:= 16\#FF000000;
\end{tabular} \\
\hline "Flags for text alignment" & \begin{tabular}{l}
Variable (integer data type). Contains the coding for text alignment. \\
Example: PLC_PRG.dwTextAlignment. \\
Coding: \\
- \(0:\) Top left \\
- 1: Horizontal center \\
- 2: Right \\
- 4: Vertical center \\
- 8: Bottom \\
Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: PLC_PRG.dwFontType := 5;
\end{tabular} \\
\hline
\end{tabular}

Fixed values for displaying texts are set in "Text properties".

See also
- « "Element property 'Text properties'" on page 3541

Element prop- The Element property is used as an interface for project variables to dynamically control colors erty 'Color varia- at runtime. bles'
\begin{tabular}{|c|c|}
\hline "Toggle color" & \begin{tabular}{l}
The property controls the toggled color at runtime. \\
Value assignment: \\
- FALSE: The element is displayed with the color specified in the "Color" property. \\
- TRUE: The element is displayed with the color specified in the "Alarm color" property. \\
Assignment options: \\
- Placeholder for the user input variable \\
- "<toggle/tap variable>" \\
- "<NOT toggle/tap variable>" \\
The color change is not controlled by its own variable, but by a user input variable. \\
Note: Specify a variable for the mouse events "Tap" or "Toggle" in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both "Toggle" and "Tap", then the variable specified in "Tap" is used. \\
Hint: Click the symbol to insert the placeholder "<toggle/tap variable>". When you activate the "Inputconfiguration", "Tap FALSE" property, then the "<NOT toggle/tap variable>" placeholder is displayed. \\
- Instance path of a project variable (BOOL) \\
Example: PLC_PRG.xColorIsToggeled \\
Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.
\end{tabular} \\
\hline \begin{tabular}{l}
"Normal state" \\
"Alarm state"
\end{tabular} & The properties listed below control the color depending on the state. The normal state is in effect if the variable in "Color variables", "Toggle color" is not defined or it has the value FALSE. The alarm state is in effect if the variable in "Colorvariables", "Toggle color" has the value TRUE. \\
\hline "Frame color" & \begin{tabular}{l}
Assignment options: \\
- Variable (DWORD) for the frame color Example: PLC_PRG.dwBorderColor \\
- Color literal Example of green and opaque: 16\#FF00FF00
\end{tabular} \\
\hline "Filling color" & \begin{tabular}{l}
Assignment options: \\
- Variable (DWORD) for the fill color Example: PLC_PRG.dwFillColor \\
- Color literal Example of gray and opaque: \(16 \#\) FF888888
\end{tabular} \\
\hline
\end{tabular}

The transparency part of the color value is evaluated only if the "Activate semitransparent drawing" option of the visualization manager is selected.

Select the "Advanced" option in the toolbar of the properties view. Then all element properties are visible.

\section*{See also}
- \& Chapter 6.4.5.10.4 "Animating a color display" on page 2914

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG. iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG. iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\boldsymbol{\oplus}\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
\hline "Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol. \\
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

The " \(X\) ", " \(Y\) ", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

\section*{See also}

\section*{Element prop- The variables control the element behavior dynamically. erty 'State variables'}
\begin{tabular}{|l|l|}
\hline "Invisible"" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
\\
\\
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \(=\) FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

\footnotetext{
Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'
}
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages: \\
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
- "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

\section*{See also}
- \(\#\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights"" on page 3366

See also
- \(\Leftrightarrow\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Slider'}

\section*{Symbol:}

\section*{Category: "Common Controls"}

The element changes the value of a variable, depending on the position of the slider within the slider bar. You define the value range of the slider bar by means of the scale start and scale end.

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Example: Speed controller conveyor belt 1 \\
Optional \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list.
\end{tabular} \\
\hline "Type of element" & "Slider" \\
\hline
\end{tabular}

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline "X" & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (ए) to other positions in the editor.

See also
- ๕ Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center' rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline " \(Y\) " & \(Y\)-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus}\) ) to other positions in the editor.
\begin{tabular}{|l|l|}
\hline "Variable" & \begin{tabular}{l} 
Variable (numeric data type) \\
Example: PLC_PRG.rSlider \\
When executed, the variable assigns a value that corresponds to the position of \\
the slider in the bar.
\end{tabular} \\
\hline "Page size" & \begin{tabular}{l} 
Page size \\
\(\bullet \quad\) As a fixed value, for example 10 \\
\(\bullet \quad\) As an IEC variable of data type integer \\
Requirement: The "Move to click" element property is not selected.
\end{tabular} \\
\hline "Move to click" & \begin{tabular}{l} 
Behavior of the slider at visualization runtime when it is clicked: \\
■: The slider moves to the clicked position.
\end{tabular} \\
& \begin{tabular}{l}
\(\square:\) The slider moves to the value (defined in the "Page size" element property) in \\
the direction of the click.
\end{tabular} \\
\hline
\end{tabular}

\section*{Element prop- \\ erty 'Scale'}
\begin{tabular}{|l|l|}
\hline "Show scale" & \begin{tabular}{l}
\(\boxed{\square}\) \\
Note: This option is available for the "Slider" only.
\end{tabular} \\
\hline "Scale start" & \begin{tabular}{l} 
Least value of the scale and the lower limit of the value range for the element. \\
Example: 0 \\
\(0:\) The property "Variable" is shown below.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the scale start. \\
Example: PLC_PRG.iScaleStart \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iScaleStart : INT := 0;
END VAR

```
\end{tabular} \\
\hline "Scale end" & \begin{tabular}{l}
Greatest value of the scale and the upper limit of the value range for the element. \\
Example: 100 \\
p: The property "Variable" is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the scale end. \\
Example: PLC_PRG.iScaleEnd \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iScaleEnd : INT := 120;
END_VAR

```
\end{tabular} \\
\hline "Main scale" & \begin{tabular}{l}
Distance between two tick marks on the rough scale. \\
Example: 10 \\
p: The property "Variable" is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the distance. \\
Example: PLC_PRG.iMainScale \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iMainScale : INT := 20;
END_VAR

```
\end{tabular} \\
\hline "Subscale" & \begin{tabular}{l}
Distance between two dashes on the fine scale. You can hide the fine scale by setting the value to 0 . \\
Example: 2 \\
*: The property "Variable" is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the distance. \\
Example: PLC_PRG.iSubScale \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iMainScale : INT := 5;
END_VAR

```
\end{tabular} \\
\hline "Scale format (C Syntax)" & \begin{tabular}{l}
Formatting of the scale label (example: \%d \%s) \\
Note: This property is available for the Slider only.
\end{tabular} \\
\hline "Scale proportion" & Size of the scale (in \%) of the total size \\
\hline
\end{tabular}

Element prop- The property defines the representation of scaling and direction of travel. erty 'Bar'
\begin{tabular}{|l|l|}
\hline "Diagram type" & \begin{tabular}{l} 
The drop-down list varies depending on the alignment of the diagram. \\
Horizontal \\
- "Top": Scale is above the slider. \\
- "Bottom": Scale is below the slider. \\
- "Top and bottom": Two scales frame the slider above and below. \\
Vertical \\
- Left: Scale is left of the slider. \\
- Right: Scale is right of the slider. \\
- Left and right: Two scales frame the slider on the left and the right.
\end{tabular} \\
\hline "Orientation" \begin{tabular}{l} 
Alignment of the slider; defined by the ratio of width to height. \\
- "Horizontal" \\
- "Vertical" \\
You can modify the alignment in the visualization editor by using the pointing \\
device to adjust the width and height of the scrollbar.
\end{tabular} \\
\hline \begin{tabular}{ll} 
The drop-down list varies depending on the alignment of the slider. \\
Horizontal \\
- "Left to right": Scale starts at the left. \\
- "Right to left": Scale starts at the right. \\
Vertical \\
- "Bottom to top": Scale starts at the bottom. \\
- "Top to bottom": Scale starts at the top.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline \multicolumn{1}{|l|}{ "Movement" } & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\oplus\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
"Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PPRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\boldsymbol{\oplus}\) symbol.
\end{tabular} \\
\begin{tabular}{l} 
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular}
\end{tabular}

You can link the variables to a unit conversion.

The " \(X\) ", " "", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

\section*{See also}

\section*{Element prop- The variables control the element behavior dynamically. erty 'State varia- \\ bles'}
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL := FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

\section*{The "Invisible" property is supported by the "Client Animation" functionality.}

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- Requirement: User management is set up for the visualization. erty 'Access
rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages: \\
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
- "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\(\qquad\)\begin{tabular}{l} 
See also \\
- \& Chapter 6.4.5.21.3.1 "Dialog 'Access Rights"" on page 3366
\end{tabular}
\end{tabular}

\section*{See also}
- © Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Spin Box'}

Symbol:


\section*{Category: "Common Controls"}

The element increments or decrements the value of a variable in defined intervals.

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Example: Speed controller conveyor belt \\
Optional \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list.
\end{tabular} \\
\hline "Type of element" "Spin Box" \\
\hline
\end{tabular}

Element prop- The position defines the location and size of the element in the visualization window. These erty 'Position' are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal x -axis runs to the right. The positive vertical y -axis runs downwards.
\begin{tabular}{|l|l|}
\hline "X" & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols ( ( - ) to other positions in the editor.

See also
- © Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X "\) & X-coordinate of the point of rotation \\
\hline\(" Y "\) & \(Y\)-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\mathbf{\oplus})\) to other positions in the editor.
\begin{tabular}{|l|l|}
\hline "Variable" & \begin{tabular}{l} 
Variable (numeric data type) \\
Example: PLC_PRG.iTemp
\end{tabular} \\
\hline "Number format" & \begin{tabular}{l} 
Format of the value in printf syntax \\
Example: \%d, \%5.2f
\end{tabular} \\
\hline "Interval" & Interval used for modification of the value \\
\hline
\end{tabular}

Element property 'Value
range'
\begin{tabular}{|l|l|}
\hline "Minimum value" & \begin{tabular}{l} 
Lower limit of the output value \\
- fixed value \\
- Variable (INT)
\end{tabular} \\
\hline "Maximum value" & \begin{tabular}{l} 
Upper limit of the output value \\
- fixed value \\
- Variable (INT)
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain fixed values for the text properties. erty 'Text properties'
\begin{tabular}{|c|c|}
\hline "Usage of" & \begin{tabular}{l}
- "Default style values": The values of the visualization style are used. \\
- "Individual settings": The "Individual text properties" property group is shown The values can be customized here.
\end{tabular} \\
\hline \multicolumn{2}{|l|}{\multirow[t]{2}{*}{\begin{tabular}{l}
"Individual text properties" \\
Requirement: The "Individual settings" text property is defined.
\end{tabular}}} \\
\hline & \\
\hline "Horizontal alignment" & Horizontal alignment of the text within the element. \\
\hline "Vertical alignment" & Vertical alignment of the text within the element. \\
\hline "Font" & \begin{tabular}{l}
Example: "Default"
The "Font" dialog box opens. \\
*: Drop-down list with style fonts.
\end{tabular} \\
\hline "Font color" & \begin{tabular}{l}
Example: "Black"
The "Color" dialog box opens. \\
- : Drop-down list with style colors.
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l}
Whole number (value range from 0 to 255). This determines the transparency of the respective color. \\
Example: 255: The color is opaque. \\
0 : The color is completely transparent. \\
Please note: If the color is a style color and already has a transparency value, then this property is write-protected.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The Element property is used as an interface for project variables to dynamically control colors erty 'Color varia- at runtime. bles'
\begin{tabular}{|c|c|}
\hline "Toggle color" & \begin{tabular}{l}
The property controls the toggled color at runtime. \\
Value assignment: \\
- FALSE: The element is displayed with the color specified in the "Color" property. \\
- TRUE: The element is displayed with the color specified in the "Alarm color" property. \\
Assigning the property: \\
- Placeholder for the user input variable \\
- "<toggle/tap variable>" \\
- "<NOT toggle/tap variable>" \\
The color change is not controlled by its own variable, but by a user input variable. \\
Note: Specify a variable for the mouse events "Tap" or "Toggle" in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both "Toggle" and "Tap", then the variable specified in "Tap" is used. \\
Hint: Click the symbol to insert the placeholder "<toggle/tap variable>". When you activate the "Inputconfiguration"," "Tap FALSE" property, then the "<NOT toggle/tap variable>" placeholder is displayed. \\
- Instance path of a project variable (BOOL) \\
Example: PLC_PRG.xColorIsToggeled \\
Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes.
\end{tabular} \\
\hline
\end{tabular}

The transparency part of the color value is evaluated only if the "Activate semitransparent drawing" option of the visualization manager is selected.

Select the "Advanced" option in the toolbar of the properties view. Then all element properties are visible.

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline \multicolumn{1}{|l|}{ "Movement" } \\
\hline " \(X\) " & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline " \(\mathrm{Y} "\) & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG. iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\oplus\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
"Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol.
\end{tabular} \\
\begin{tabular}{l} 
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

The " \(X\) ", " \(Y\) ", "Rotation", and "Interior rotation" properties are supported by the
"Client Animation" functionality.

See also

Element prop- The variables control the element behavior dynamically.
erty 'State varia-
bles'
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \(:=\) FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
\\
\\
\begin{tabular}{l} 
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain the configurations for the user input when using the mouse or keyboard. erty 'Input con- User input is a user event from the perspective of the element. figuration'


The "Configure" button opens the "Input configuration" dialog box for creating or modifying a user input configuration.

A configuration contains one or more input actions for the respective input event. Existing input actions are displayed below it.
Example: "Execute ST code": З PLC_PRG.i_x := 0;
\begin{tabular}{|l|l|}
\hline "OnDialogClosed" & Input event: The user closes the dialog box. \\
\hline "OnMouseClick" & \begin{tabular}{l} 
Input event: The user clicks the element completely. The mouse button is clicked \\
and released.
\end{tabular} \\
\hline "OnMouseDown" & Input event: The user clicks down on the element only. \\
\hline "OnMouseEnter" & Input event: The user drags the mouse pointer to the element. \\
\hline "OnMouseLeave" & Input event: The user drags the mouse pointer away from the element. \\
\hline "OnMouseMove" & Input event: The user moves the mouse pointer over the element area. \\
\hline "OnMouseUp" & Input event: The user releases the mouse button over the element area. \\
\hline
\end{tabular}

See also
- \({ }^{\wedge}>\) Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration'" on page 3370
\begin{tabular}{|c|c|}
\hline "Tap" & When a mouse click event occurs, the variable defined in "Variable" is described in the application. The coding depends on the options "Tap FALSE" and "Tap on enter if captured". \\
\hline "Variable" & \begin{tabular}{l}
Variable (BOOL). Contains the information whether a mouse click event exists. \\
Example: PLC_PRG.bIsTapped \\
TRUE: A mouse click event exists. It lasts while the user presses the mouse button over the element. It ends when the button is released. \\
FALSE: A mouse click event does not exist. \\
Requirement: The "Tap FALSE" option is not activated.
\end{tabular} \\
\hline "Tap FALSE" & \begin{tabular}{l}
The mouse click event leads to a complementary value in "Variable". TRUE: A mouse click event does not exist. \\
FALSE: While the mouse click event exists.
\end{tabular} \\
\hline "Tap on enter if captured" & \begin{tabular}{l}
During user input, it is also taken into consideration whether the mouse pointer is dragged within the element area or not while the mouse button is pressed. \\
TRUE: While the mouse click event exists and the mouse pointer is moved over the element area. \\
FALSE: A mouse click event does not exist. Or the user moves the mouse pointer outside of the element area while the mouse button is pressed. \\
The value is TRUE again as soon as the user moves the pointer back to the element area. The mouse is then captured.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Shift" & \begin{tabular}{l} 
When a mouse click event occurs, the variable here is described in the applica- \\
tion. When the mouse click event ends, its value is toggled with the "Toggle on \\
up if captured" option.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL). Its value toggled when the mouse click event is ended. This is \\
when the user releases the mouse button while the mouse pointer is over the \\
element area. \\
If the user releases the mouse button while the mouse pointer is outside of the \\
element area, then the mouse click event is not ended and the value is not \\
toggled. \\
Tip: The user can cancel a started toggle input by dragging the mouse pointer \\
out of the element area.
\end{tabular} \\
\hline "Toggle on up if captured" & \begin{tabular}{l}
\(\boxed{\square}\) \\
button is released. The mouse is then captured.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Hotkeys" & \begin{tabular}{l}
Keyboard shortcut on the element for triggering specific input actions. \\
When the keyboard shortcut event occurs, the input actions in the "Event(s)" property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.
\end{tabular} \\
\hline "Key" & \begin{tabular}{l}
Key pressed for input action. \\
Example: [T] \\
Note: The following properties appear when a key is selected.
\end{tabular} \\
\hline "Event(s)" & \begin{tabular}{l}
- "None" \\
- "Mouse down": Pressing the key triggers the input actions that are configured in the "OnMouseDown" property. \\
- "Mouse up": Releasing the key triggers the input actions that are configured in the "OnMouseUp" property. \\
- "Mouse down/up": Pressing and releasing the key triggers the input actions that are configured in the "OnMouseDown" property and the "OnMouseUp" property.
\end{tabular} \\
\hline "Shift" & Combination with the Shift key Example: [Shift]+[T]. \\
\hline "Control" & Combination with the Ctrl key Example: [Ctrl]+[T]. \\
\hline "Alt" & Combination with the Alt key Example: \([\) Alt \(]+[T]\). \\
\hline
\end{tabular}


See also
- \({ }^{\Perp}\) Chapter 6.4.5.21.2.2 "Command 'Keyboard Configuration"" on page 3341
- \(\quad\) Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration'" on page 3370

Element prop- Requirement: User management is set up for the visualization.
erty 'Access
rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages: \\
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
\\
- "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

\section*{See also}
- \({ }^{\#}\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights"" on page 3366

\section*{Visualization Element 'Invisible Input'}

\section*{Symbol:}


\section*{Category: "Common Controls"}

This element is displayed in the editor with a dashed line which is not visible in online mode. You define the behavior of the el in the input configuration.

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name"" & \begin{tabular}{l} 
Optional \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list. \\
Example: Unsichtbare_Eingabe_1
\end{tabular} \\
\hline "Type of element" & "Invisible Input" \\
\hline
\end{tabular}

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline "X" & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (ए) to other positions in the editor.

See also
- « Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center'
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline " \(Y\) " & \(Y\)-coordinate of the point of rotation \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|c|c|c|}
\hline "Movement" & & \\
\hline "X" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the \(X\) position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular}} \\
\hline "Y" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular}} \\
\hline "Rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle1. \\
The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol. \\
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
\end{tabular} &  \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center"according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol.
\end{tabular} \\
\begin{tabular}{l} 
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

The "X", " \(Y\) ", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also

Element prop- The variables control the element behavior dynamically. erty 'State varia-
bles'
\begin{tabular}{|l|l|}
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", "X", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain the configurations for the user input when using the mouse or keyboard. erty 'Input con- A user input defines an event and one or more actions that are executed when an event occurs. figuration'
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
The "Configure" button opens the "Input Configuration" dialog. There you can create or edit user inputs. \\
Configured user inputs are listed below the events. They each include the action that is triggered and the setting \\
in short form. \\
Example: "Execute ST Code":
\end{tabular} \\
\hline "Onc__PRG.i_x \(:=0 ;\) \\
\hline "OnDialogClosed" & Input event: The user closes the dialog. \\
\hline "OnMouseDown" & \begin{tabular}{l} 
Input event: The user clicks the mouse button completely in the element area. \\
The mouse button is clicked and released.
\end{tabular} \\
\hline "OnMouseEnter" & Input event: The user clicks down on the mouse button. \\
\hline "OnMouseLeave" & Input event: The user drags the mouse pointer to the element. \\
\hline
\end{tabular}
\(\left.\begin{array}{|l|l|}\hline \text { "OnMouseMove" } & \text { Input event: The user moves the mouse pointer over the element area. } \\ \hline \text { "OnMouseUp" } & \begin{array}{l}\text { Input events: } \\ \text { - } \\ \text { The user releases the mouse button within the element area. It is irrelevant } \\ \text { whether the user has previously pressed the mouse button inside or outside } \\ \text { the element area. }\end{array} \\ \text { - The user presses the mouse button within the element area, leaves the } \\ \text { element area, and then releases the mouse button. } \\ \text { Note: This CODESYS-specific triggering behavior guarantees that actions for } \\ \text { key elements are completed. A key element starts an action for "OnMouseDown" } \\ \text { and ends the action for "OnMouseUp". } \\ \text { Example: A visualization user presses the mouse button within the element area } \\ \text { of the key element and then moves the cursor position so that it lies outside the } \\ \text { element area. The action is ended anyway because "OnMouseUp" is triggered. }\end{array}\right\}\)
\begin{tabular}{|l|l|}
\hline "Tap" & \begin{tabular}{l} 
When a mouse click event occurs, the variable defined in "Variable" is described \\
in the application. The coding depends on the "Tap FALSE" and "Tap on enter if \\
captured" options.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL) that is set on mouse click event. \\
Example: PLC_PRG . bI sTapped \\
TRUE: A mouse click event exists. It lasts as long as the user presses the mouse \\
button over the element. It ends when the button is released.
\end{tabular} \\
& \begin{tabular}{l} 
FALSE: A mouse click event does not exist. \\
Requirement: The "Tap FALSE" option is not activated.
\end{tabular} \\
\hline "Tap FALSE" & \begin{tabular}{l} 
V: The mouse click event leads to a complementary value in "Variable". \\
TRUE: A mouse click event does not exist.
\end{tabular} \\
\hline "Tap on enter if captured" & \begin{tabular}{l} 
FALSE: While the mouse click event exists.
\end{tabular} \\
\hline \begin{tabular}{l} 
pointer is dragged within the element area or not while the mouse button is \\
pressed. \\
TRUE: While the mouse click event exists and the mouse pointer is moved over \\
the element area. \\
FALSE: A mouse click event does not exist. Or the user moves the mouse \\
pointer outside of the element area while the mouse button is pressed. \\
The value is TRUE again as soon as the user moves the pointer back to the \\
element area. The mouse is then captured.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Toggle" & \begin{tabular}{l} 
With the onset of a mouse click event, the variable is set; when the mouse click \\
event is completed, the variable is reset.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (BOOL). Its value toggled when the mouse click event is ended. This is \\
when the user releases the mouse button while the mouse pointer is over the \\
element area. \\
If the user releases the mouse button while the mouse pointer is outside of the \\
element area, then the mouse click event is not ended and the value is not \\
toggled. \\
Hint: The user can cancel a started toggle input by dragging the mouse pointer \\
out of the element area.
\end{tabular} \\
\hline "Toggle on up if captured" & \begin{tabular}{l} 
V: The value toggles regardless of where the mouse pointer is when the mouse \\
button is released. The mouse is then captured.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Hotkey" & \begin{tabular}{l}
Keyboard shortcut on the element for triggering specific input actions. \\
When the keyboard shortcut event occurs, the input actions in the "Events" property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action.
\end{tabular} \\
\hline "Key" & \begin{tabular}{l}
Key pressed for input action. \\
Example: [T] \\
Note: The following properties appear when a key is selected.
\end{tabular} \\
\hline "Events" & \begin{tabular}{l}
- "None" \\
- "Mouse down": Pressing the key triggers the input actions that are configured in the "OnMouseDown" property. \\
- "Mouse up": Releasing the key triggers the input actions that are configured in the "OnMouseUp" property. \\
- "Mouse down/up": Pressing and releasing the key triggers the input actions that are configured in the "OnMouseDown" property and the "OnMouseUp" property.
\end{tabular} \\
\hline "Shift" & Combination with the Shift key Example: [Shift]+[T]. \\
\hline "Control" & Combination with the Ctrl key Example: [Ctrl]+[T]. \\
\hline "Alt" & Combination with the Alt key Example: \([A I t]+[T]\). \\
\hline
\end{tabular}

All keyboard shortcuts and their actions that are configured in the visualization are listed on the "Keyboard Configuration" tab.

\section*{See also}
- ⓨ Chapter 6.4.5.21.2.2 "Command 'Keyboard Configuration'" on page 3341
- \(\xi^{2}\) Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration'" on page 3370

Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages:
\end{tabular} \\
& \begin{tabular}{ll}
\(\bullet\) "Not set. Full rights.": Access rights for all user groups : "operable" \\
\(\bullet \quad\) "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\xi^{2}\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

See also
- \(\xi^{4}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Check Box'}

Symbol:

\section*{Category: "Common Controls"}

The element is used for setting and resetting a Boolean variable. The set state is identified by a check mark.

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Example: signal_tone_for_parts_deficit \\
Optional \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list.
\end{tabular} \\
\hline "Type of element" & "Check Box" \\
\hline "Text ID" & \begin{tabular}{l} 
ID for the text in the "GlobalTextList" \\
Example: 22 \\
The text I I cannot be changed. As soon as you specify and save a text in \\
"Texts" - "ext", CODESYS automatically creates an entry in the "GlobalTextList" \\
and displays the ID here.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X "\) & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (匝) to other positions in the editor.

\section*{See also}
- \({ }^{\circ} \boldsymbol{y}\) Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center'
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline " \(Y\) " & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus}\) ) to other positions in the editor.
\begin{tabular}{|l|l|}
\hline "Variable" & \begin{tabular}{l} 
Variable of type BOOL \\
Example: "PLC_PRG.xIsTrue"
\end{tabular} \\
\hline "Frame size" & \begin{tabular}{l} 
Distance of the element to the edge \\
Example: "From style"
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contains character strings for labeling the element. erty 'Texts'

CODESYS accepts the specified texts automatically into the "GlobalTextList" text list. Therefore, these texts can be localized.
\begin{tabular}{|l|l|}
\hline "Text" & \begin{tabular}{l} 
Character string (without single straight quotation marks) for the labeling the \\
element. \\
Example: Axis 1.
\end{tabular} \\
\hline "Tooltip" & \begin{tabular}{l} 
Character string (without single straight quotation marks) that is displayed as the \\
tooltip of an element. \\
Example: Parameters of Axis 1.
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\Leftrightarrow\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

Element prop- The properties contain fixed values for the text properties.
erty 'Text properties'
\begin{tabular}{|c|c|}
\hline "Usage of" & \begin{tabular}{l}
- "Default style values": The values of the visualization style are used. \\
- "Individual settings": The "Individual text properties" property group is shown The values can be customized here.
\end{tabular} \\
\hline \multicolumn{2}{|l|}{\multirow[t]{2}{*}{\begin{tabular}{l}
"Individual text properties" \\
Requirement: The "Individual settings" text property is defined.
\end{tabular}}} \\
\hline & \\
\hline "Horizontal alignment" & Horizontal alignment of the text within the element. \\
\hline "Vertical alignment" & Vertical alignment of the text within the element. \\
\hline "Text format" & \begin{tabular}{l}
Definition for displaying texts that are too long \\
- "Default": The long text is truncated. \\
- "Line break": The text is split into parts. \\
- "Ellipsis": The visible text ends with "..." indicating that it is not complete.
\end{tabular} \\
\hline "Font" & \begin{tabular}{l}
Example: "Default" \\
...: The "Font" dialog box opens. \\
*: Drop-down list with style fonts.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Font color" & \begin{tabular}{l} 
Example: "Black" \\
\\
\\
\hline... : The "Color" dialog box opens. \\
マ: Drop-down list with style colors.
\end{tabular} \\
\hline & \begin{tabular}{l} 
Whole number (value range from 0 to 255 ). This determines the transparency of \\
the respective color. \\
Example: \(255:\) The color is opaque. \\
\(0:\) The color is completely transparent. \\
Please note: If the color is a style color and already has a transparency value, \\
then this property is write-protected.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG. iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG. iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\oplus\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
\hline "Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol. \\
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
\hline
\end{tabular}

\section*{You can link the variables to a unit conversion.}

X, "Y", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also

Element prop- The variables control the element behavior dynamically. erty 'State variables'
\begin{tabular}{|l|l|}
\hline "Invisible" & Variable (BOOL). Toggles the visibility of the element. \\
& TRUE: The element is not visible at runtime. \\
& \begin{tabular}{l} 
Example: bIsVisible with VAR bIsVis ible : BOOL : \(=\) FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- Requirement: User management is set up for the visualization. erty 'Access
rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages:
\end{tabular} \\
& \begin{tabular}{l} 
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
\(\bullet\) \\
"Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{\wedge} \Rightarrow\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

See also
- \({ }^{\star}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Progress Bar'}

Symbol:


\section*{Category: "Common Controls"}

The element displays the value of a variable as a progress bar.

\section*{Element proper- \\ ties}
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Optional \\
Hint: Assign individual names for elements so that they are found faster in the \\
element list. \\
Example: Progress_Data_Transfer
\end{tabular} \\
\hline "Type of element" & "Progress Bar" \\
\hline "Text ID" & \begin{tabular}{l} 
ID of the global text list \\
Requirement: Text is configured in the property "Texts \(\rightarrow\) Text".
\end{tabular} \\
\hline "Variable" & Variable (numeric data type). Represents the length of the progress bar. \\
\hline "Minimum value" & Value range of the variable \\
\hline "Maximum value" & \begin{tabular}{l}
\(\bullet\) "Blocks" \\
\(\bullet\) \\
"Style" "Bar"
\end{tabular} \\
\hline
\end{tabular}

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X "\) & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (ए़+) to other positions in the editor.

See also
- \({ }^{\text {® }}\), Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline "X" & X-coordinate of the point of rotation \\
\hline "Y" & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\mathbf{\oplus})\) to other positions in the editor.

Element prop-
erty 'Texts'
\begin{tabular}{|l|l|}
\hline "Text" & \begin{tabular}{l} 
String label for the element. \\
Example: Zoom
\end{tabular} \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" & \multicolumn{1}{|l|}{\begin{tabular}{ll} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular}} \\
\hline "Y" & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG. iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\oplus\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
\hline "Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol. \\
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

The "X", " \(Y\) ", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also
-

Element prop- The variables control the element behavior dynamically. erty 'State variables'
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- Requirement: User management is set up for the visualization. erty 'Access
rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages:
\end{tabular} \\
& \begin{tabular}{l} 
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
\(\bullet\) \\
"Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{\wedge} \Rightarrow\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

See also
- *y Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Radio Buttons'}

Symbol:


\section*{Category: "Common Controls"}

The element provides a series of radios buttons with an unlimited number of options.

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & \begin{tabular}{l} 
Optional \\
\\
\\
Hint: Assign individual names for elements so that they are found faster in the \\
element list. \\
Example: Morning Shift
\end{tabular} \\
\hline "Type of element" & "Radio Buttons" \\
\hline
\end{tabular}

Element prop- The position defines the location and size of the element in the visualization window. These erty 'Position' are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X\) " & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (區) to other positions in the editor.

See also
- \({ }^{\text {• }}\) Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874
\begin{tabular}{|l|l|}
\hline "Variable" & \begin{tabular}{l} 
Variable (integer data type) that gives the index of the radio button that the \\
visualization user has activated \\
Example: PLC_PRG. iNrOfActivatedRadioButton
\end{tabular} \\
\hline "Number of columns" & \begin{tabular}{l} 
Definition of the number of list boxes displayed in a row \\
Example: 2
\end{tabular} \\
\hline "Radio button order" & \begin{tabular}{l} 
"Left to right": The radio buttons are aligned by rows until the number of columns \\
is reached. \\
"Top to bottom": The radio buttons are aligned row by columns until the number \\
of columns is reached.
\end{tabular} \\
\hline "Frame size" & Defines the distance from the list boxes to the edge (in pixels). \\
\hline "Row height" & \begin{tabular}{l} 
Height of the row (in pixels) Modifying the height of the row also changes the \\
size of the list box.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain fixed values for the text properties.
erty 'Text prop-
erties'
\begin{tabular}{|c|c|}
\hline "Usage of" & \begin{tabular}{l}
- "Default style values": The values of the visualization style are used. \\
- "Individual settings": The "Individual text properties" property group is shown The values can be customized here.
\end{tabular} \\
\hline \multicolumn{2}{|l|}{\multirow[t]{2}{*}{\begin{tabular}{l}
"Individual text properties" \\
Requirement: The "Individual settings" text property is defined.
\end{tabular}}} \\
\hline & \\
\hline "Horizontal alignment" & Horizontal alignment of the text within the element. \\
\hline "Vertical alignment" & Vertical alignment of the text within the element. \\
\hline "Text format" & \begin{tabular}{l}
Definition for displaying texts that are too long \\
- "Default": The long text is truncated. \\
- "Line break": The text is split into parts. \\
- "Ellipsis": The visible text ends with "..." indicating that it is not complete.
\end{tabular} \\
\hline "Font" & \begin{tabular}{l}
Example: "Default"
The "Font" dialog box opens. \\
*: Drop-down list with style fonts.
\end{tabular} \\
\hline "Font color" & \begin{tabular}{l}
Example: "Black"
: The "Color" dialog box opens. \\
- Drop-down list with style colors.
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l}
Whole number (value range from 0 to 255). This determines the transparency of the respective color. \\
Example: 255: The color is opaque. \\
0 : The color is completely transparent. \\
Please note: If the color is a style color and already has a transparency value, then this property is write-protected.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" & \multicolumn{1}{|l|}{\begin{tabular}{ll} 
"X" & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline " \(Y\) " & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG. iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline
\end{tabular}} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\oplus\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
"Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PPRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\boldsymbol{\oplus}\) symbol.
\end{tabular} \\
\begin{tabular}{l} 
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular}
\end{tabular}

You can link the variables to a unit conversion.

The " \(X\) ", " "", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

\section*{See also}

\section*{Element prop- The variables control the element behavior dynamically. erty 'State varia- \\ bles'}
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL := FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

\section*{The "Invisible" property is supported by the "Client Animation" functionality.}
\begin{tabular}{|c|c|c|}
\hline \multirow[t]{5}{*}{Element property 'Radio button settings'} & \begin{tabular}{l}
"Radio button" \\
- "Areas" \\
- " \([<n>]\) "
\end{tabular} & \begin{tabular}{l}
"Create new": Clicking this button creates a new selection button in the editor and lists an additional area in the properties editor. \\
For each radio button, an area is visible that records the settings. \\
- [<n>] \\
- "[<n>]": This number indicates the area. Clicking "Delete" will delete the associated radio button with its settings "Text", "Tooltip", and "Line spacing (in pixels)".
\end{tabular} \\
\hline & Areas: [<n>] & \\
\hline & "Text" & The button name is specified here. Default value: "Radio_button" \\
\hline & "Tooltip" & Text is specified here that is displayed in a tooltip. \\
\hline & "Line spacing (in pixels)" & The distance (in pixels) to the upper button can be specified here. \\
\hline
\end{tabular}

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

\footnotetext{
Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'
}
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages:
\end{tabular} \\
& - "Not set. Full rights.": Access rights for all user groups : "operable" \\
& - "Rights are set: Limited rights": Access is restricted for at least one group. \\
\hline
\end{tabular}

See also
- \(\boldsymbol{y}^{2}\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

\section*{See also}
- ② Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Alarm Table'}

Symbol:


\section*{Category: "Alarm Manager"}

The element displays alarms in a list. In the element properties, you specify which information is shown. You define the appearance of the element and the variables that control the element behavior.

In online mode, you can sort an alarm table by a specific column - even in the classic view. Click into the column header. A small triangle indicates the current sort order (ascending, descending). Clicking the symbol reverses the order.

Sorting inside the column depends on the type of the contained information. The "Priority" column is sorted numerically, and the "Message" and "Class" columns alphabetically. The "Value" and "Latch" columns may contain different value types. In this case, sorting is first by type (blank, Boolean, numeric value, character string) and then either numerically or alphabetically depending on the type.
If an alarm history has been created, then you can programmatically delete it at runtime. The recording starts again from the time of deletion. See the help page for "Visualizing Alarm Management".

\section*{Element proper- \\ ties}
\begin{tabular}{|c|c|}
\hline "Element name" & Example: GenElemInst_1 \\
\hline "Type of element" & "Alarm Table" \\
\hline "Data source" & \begin{tabular}{l}
Selection of the device and the application where the data to be visualized and the alarms are generated \\
- Remote data source which accesses a remote device, accesses a remote application, and then transfers the data to the alarm configuration \\
Example: \(\mathbb{x}^{\text {PI }}\) DataSource_A \\
Below the (now visible) "Application" property, the remote application is displayed as configured in the data source. \\
Example: \\
Note: If the data source is accessed symbolically by means of a symbol file (CODESYS symbolic), then the required symbol file and the corresponding project have to be saved in the same folder. \\
- Local application below which the alarm configuration is located Example: "<local application>"
\end{tabular} \\
\hline
\end{tabular}

See also
- Object 'Data Source'

Element property 'Alarm con-
figuration'
\begin{tabular}{|l|l|}
\hline "Alarm groups" & \begin{tabular}{l} 
Opens the "Select Alarm Group" dialog where you define the alarm groups that \\
you want to display.
\end{tabular} \\
\hline "Priority from" & Least priority for alarm display. (0 to 255). \\
\hline "Priority to" & Greatest priority for alarm display. (0 to 255). \\
\hline "Alarm classes" & \begin{tabular}{l} 
Opens the "Select Class Group" dialog where you define the alarm classes that \\
you want to display.
\end{tabular} \\
\hline "Filter criterion" & \begin{tabular}{l} 
For the "Alarm Banner" element only \\
- "Most important": The alarm with the highest priority (lowest value) is dis- \\
played. \\
- "Newest": The most recent alarm is displayed.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Filter by latch 1" & \begin{tabular}{l}
The generated alarms (previous and current) can be filtered by the contents of "Latch variable 1", which is specified in the configuration of the alarm group. In "Filter type", you define whether or not the filtering is performed by a string value or a numerical value. \\
- "Filter variable": Indicates what the alarms are filtered by. Possible entries: Application variable of data type STRING or WSTRING, or a literal value directly. Examples:PLC_PRG.strFilterVariable, 'STRING'. \\
- "Filter type": Integer value that determines by which criteria the latch variable value is used for filtering. Possible entries: Numerical variable from the application (example: PLC_PRG.diFilterType, or a value directly (example: 2). \\
Possible values: \\
- 0: No filtering \\
- 1: Filter by alarms whose latch variable 1 contains the string specified in "Filter variable". Example: The filter variable contains 'Error 1 ' which is the latch variable 1 of different alarms of type STRING and has the value 'Error 1' ->. Only these alarms are displayed. \\
- 2: Filter by alarms whose latch variable 1 contains the typed literal specified in "Filter variable" according to IEC 61131-3. Examples: T\#1h2s, DINT\#15, REAL\#1.5, FALSE \\
- 3: Filter by alarms whose latch variable 1 contains the LINT literal value specified in "Filter variable". Therefore, the value of the latch variables has to be in the range of \(9,223,372,036,854,775,808\) to 9,223,372,036,854,775,807. \\
- All other values: The behavior is not defined and can change in the future.
\end{tabular} \\
\hline "Filter by time range" & \begin{tabular}{l}
The generated alarms (remote, historical, local) can be displayed for a specified time range. You use the "Filter type" to define whether filtering by time range is enabled or disabled. \\
- "Filter variable, from": Variable of data type DT or DATE_AND_TIME (example: PLC_PRG.filterTimeFrom) for the start time that the alarms are displayed. \\
- "Filter variable, to": Variable of data type DT or DATE_AND_TIME (example: PLC_PRG.filterTimeTo) for the end time that the alarms are displayed. \\
- "Filter type": Variable of integer data type that determines whether "Filter by time range" is enabled or disabled. \\
Possible values: \\
- 1: Filtering is enabled \\
- 0 : Filtering is disabled
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{4}\), Chapter 6.4.5.21.3.17 "Dialog 'Selected Alarm Group'" on page 3390
- \(\Rightarrow\) Chapter 6.4.5.21.3.16 "Dialog 'Selected Alarm Class'" on page 3389

\section*{Element property 'General table configuration'}
\begin{tabular}{|l|l|}
\hline "Show row header" & \(\checkmark\) : Display of the row number at the beginning of the row. \\
\hline "Show column header" & \(\boxed{\square}\) : Display of the column heading as defined in "Column heading". \\
\hline "Row height" & Height of the table rows (in pixels). \\
\hline "Row header width" & Width of the line header (in pixels). \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Scrollbar size" & \begin{tabular}{l} 
Width of the scrollbar when it runs vertically. Width of the scrollbar when it runs \\
horizontally. Specified in pixels
\end{tabular} \\
\hline "Automatic line break for alarm & \(\boxed{\square}\) : The message text is truncated at the end of the line. \\
message" & \(\square\) : The message text is truncated at the end of the column, if the text is too long. \\
\hline
\end{tabular}

Element property 'Columns: Column [<n>]'

By default, columns [0] and [1] are configured: "Time stamp" and "Message text". You can create more columns by clicking the "Create new", and remove columns by clicking "Delete".

Animations (dynamic text, font variables), text, and tooltip are not supported.
\begin{tabular}{|c|c|}
\hline "Column header" & The standard header is set and changed here by specifying a new text. \\
\hline "Use text alignment in title" & The text in the column header is aligned according to the current definition in "Text alignment".
\(\square\) : The text in the column header is centered. \\
\hline "Width" & Width of the column (in pixels). \\
\hline "Data type" & \begin{tabular}{l}
Notice about time stamps: For use in a TargetVisu or WebVisu, you can control the date and time format with the help of the global string variables from the library Alarmmanager. library: AlarmGlobals.g_sDateFormat (example: AlarmGlobals.g_sDateFormat := 'MM.yyyy') and AlarmGlobals.g_sTimeFormat (example: AlarmGlobals.g_sTimeFormat := 'HH:mm'). \\
Define the information to be displayed in the column. \\
- "Symbol" \\
- "Time stamp": Date and time of the last status change of the alarm. \\
- "Time stamp active": Date and time of the last activation of the alarm. \\
- "Time stamp inactive": Date and time of the last deactivation of the alarm. \\
- "Time stamp acknowledge": Date and time of the last acknowledgment. \\
- "Value": Current value of the printout \\
- "Message text": Output of the message text \\
- "Priority": Alarm priority \\
- "Class": Alarm class \\
- "State": Alarm state \\
- "Latch Variable <n>": Value of the selected latch variables
\end{tabular} \\
\hline "Text alignment" & \begin{tabular}{l}
Alignment of the text in this column \\
- "Left" \\
- "Centered" \\
- "Right"
\end{tabular} \\
\hline "Color settings" & \begin{tabular}{l}
- "Activate color settings": Boolean variable for activating and deactivating the color settings defined here. Example: PLC_PRG.bColorSettings \\
- "Cell fill color": \\
- "Color variable": Variable for the cell fill color, example: dwCellColor (hexadecimal color definition: 16\#TTRRGGBB) \\
- "Use color also for column header": \(\square\) : The color defined via "Color variable" is used in the column header as well. \\
- "Text color": \\
- "Color variable": Variable for the definition of the text color in the column, example: dwTextColor (hexadecimal color definition: 16\#TTRRGGBB) \\
- "Use color also for column header": \(\square\) : The color defined via "Color variable" is used in the column header as well.
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{4}\) Chapter 6.4.5.10.4 "Animating a color display" on page 2914

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X "\) & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (무-) to other positions in the editor.

See also
- *a Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X "\) & X-coordinate of the point of rotation \\
\hline\(Y "\) & \(Y\)-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus})\) to other positions in the editor.

Element prop- The properties contain fixed values for the text properties. erty 'Text properties'
\begin{tabular}{|l|l|}
\hline "Font" & \begin{tabular}{l} 
Example: "Default" \\
n...: The "Font" dialog box opens. \\
-: Drop-down list with style fonts.
\end{tabular} \\
\hline "Font color" & \begin{tabular}{l} 
Example: "Black" \\
\(\ldots .\). : The "Color" dialog box opens. \\
-: Drop-down list with style colors.
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l} 
Whole number (value range from 0 to 255). This determines the transparency of \\
the respective color. \\
Example: \(255:\) The color is opaque. \\
: The color is completely transparent. \\
Please note: If the color is a style color and already has a transparency value, \\
then this property is write-protected.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Selection'
\begin{tabular}{|c|c|}
\hline "Selection color" & Fill color of the selected row \\
\hline "Selection font color" & Font color of the selected row \\
\hline "Frame around selected cells" & \(\square\) : A frame is drawn around the selected cells at runtime. \\
\hline "Variable for selected alarm group" & Name of the affected alarm group; type: STRING, WSTRING \\
\hline "Variable for selected alarm ID" & Alarm ID of the affected alarm group; type: STRING, WSTRING \\
\hline "Variable for selected line" & Index of the selected alarm line (0-based). The index can be read and written; integer data type \\
\hline "Variable for valid selection" & \begin{tabular}{l}
TRUE: An alarm line is selected. \\
FALSE: The selection is invalid. For example, for an empty alarm table or when an alarm is not selected yet.
\end{tabular} \\
\hline "Variable for selected alarm information" & \begin{tabular}{l}
Information about the selected alarm. Type AlarmSelectionInfo \\
For easy usage, the function block AlarmSelectionInfoDefault is provided. This FB fills the structure with the memory for 10 messages and 10 latch variables. \\
Example: myAlarmSelectionInfoDefault.AlarmSelectionInfo \\
The following information is available: \\
- sAlarmgroup \\
- uialarmID \\
- timeStampActive \\
- timeStampInactive \\
- timeStampAcknowledge \\
- timeStampLast \\
- paLatchVariables \\
- iLatchVariablesCount \\
- papwsAlarmMessages \\
- dwAlarmMessageTextBufferSize \\
- iAlarmMessagesCount \\
- iSelectionChangedCounter
\end{tabular} \\
\hline
\end{tabular}

Element prop-
erty 'Control Boolean variables are defined here for executing specific actions in the table can be executed at runtime.

\section*{variables'}
\begin{tabular}{|l|l|}
\hline "Acknowledge selected" & \begin{tabular}{l} 
Variable (BOOL) \\
Example: PLC_PRG. bAckSelectedAlarms \\
If the assigned variable is TRUE, then the selected alarm is acknowledged.
\end{tabular} \\
\hline "Acknowledge all visible" & \begin{tabular}{l} 
Variable (BOOL) \\
Example: PLC_PRG. bAckVisi.bleAlarms \\
If the assigned variable is TRUE, then all alarms are acknowledged that are \\
visible in the alarm table.
\end{tabular} \\
\hline "History" & \begin{tabular}{l} 
Variable (BOOL) \\
Example: PLC_PRG. bShowHistory \\
If the assigned variable is TRUE, then the history alarms are displayed in addition \\
to the active alarms. In the classic view, the same sort options apply as in normal \\
mode. \\
Note: Acknowledgment is not possible in this view.
\end{tabular} \\
\hline "Freeze scroll position" & \begin{tabular}{l} 
Variable (BOOL) \\
Example: PLC_PRG . bFreezeScrollPosition \\
"Variable for sorting direction"
\end{tabular} \\
\hline Variable (BOOL) \\
Example: PLC_PRG. xSortAscending \\
is retained, even if a new alarm is active. If not, then the scroll position jumps to \\
the first table row (the newest alarm).
\end{tabular}


You can also use the "Insert Elements for Acknowledging Alarms" command to define buttons with predefined control variables.

See also
- © Chapter 6.4.5.21.2.23 "Command 'Add Elements for Alarm Acknowledgement"' on page 3365

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|c|c|c|}
\hline \multicolumn{3}{|l|}{"Movement"} \\
\hline " \(\times\) " & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular}} \\
\hline "Y" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the \(Y\) position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular}} \\
\hline "Rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle1. \\
The midpoint of the element rotates at the "Center" point. This rotation point is shown as the \(\boldsymbol{\oplus}\) symbol. \\
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
\end{tabular} &  \\
\hline "Interior rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle2. \\
In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. \\
The rotation point is shown as the \(\boldsymbol{\oplus}\) symbol. \\
Note: If a static angle of rotation is specified in the "Position \(\rightarrow\) Angle" property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.
\end{tabular} &  \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

The " \(X\) ", " \(Y\) ", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also

Element prop- The variables control the element behavior dynamically. erty 'State variables'
\begin{tabular}{|l|l}
\hline "Invisible" & Variable (BOOL). Toggles the visibility of the element.
\end{tabular}

TRUE: The element is not visible at runtime.

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", "X", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

\footnotetext{
Element property 'Access
rights'
}
\begin{tabular}{|c|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages: \\
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
- "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\hline See also
\end{tabular}
- \(y^{3}\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

\section*{Visualization Element 'Alarm Banner'}

Symbol:


\section*{Category: "Alarm Manager"}

The element is a simplified version of the alarm table. It visualizes a single alarm only. In the element properties, you specify which information is shown. You define the appearance of the element and the variables that control the element behavior.

The alarm banner displays active alarms only. If the alarm is acknowledged, then it disappears from the alarm banner.

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & Example: GenElem Inst_1 \\
\hline "Type of element" & "Alarm Banner" \\
\hline "Data source" & \begin{tabular}{l} 
If you intend to use a remote alarm configuration, then you have to specify the \\
name of the remote application here. If you do not specify anything, the alarm \\
configuration will be located locally.
\end{tabular} \\
\hline
\end{tabular}

\section*{Element prop- \\ erty 'Alarm con- \\ figuration'}
\begin{tabular}{|l|l|}
\hline "Alarm groups" & \begin{tabular}{l} 
Opens the "Select Alarm Group" dialog where you define the alarm groups that \\
you want to display.
\end{tabular} \\
\hline "Priority from" & Least priority for alarm display. (0 to 255). \\
\hline "Priority to" & Greatest priority for alarm display. (0 to 255). \\
\hline "Alarm classes" & \begin{tabular}{l} 
Opens the "Select Class Group" dialog where you define the alarm classes that \\
you want to display.
\end{tabular} \\
\hline "Filter criterion" & \begin{tabular}{l} 
For the "Alarm Banner" element only \\
- "Most important": The alarm with the highest priority (lowest value) is dis- \\
played. \\
-"Newest": The most recent alarm is displayed.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Filter by latch 1" & \begin{tabular}{l}
The generated alarms (previous and current) can be filtered by the contents of "Latch variable 1 ", which is specified in the configuration of the alarm group. In "Filter type", you define whether or not the filtering is performed by a string value or a numerical value. \\
- "Filter variable": Indicates what the alarms are filtered by. Possible entries: Application variable of data type STRING or WSTRING, or a literal value directly. Examples:PLC_PRG.strFilterVariable, 'STRING'. \\
- "Filter type": Integer value that determines by which criteria the latch variable value is used for filtering. Possible entries: Numerical variable from the application (example: PLC_PRG.diFilterType, or a value directly (example: 2). \\
Possible values: \\
- 0: No filtering \\
- 1: Filter by alarms whose latch variable 1 contains the string specified in "Filter variable". Example: The filter variable contains 'Error 1 ' which is the latch variable 1 of different alarms of type STRING and has the value 'Error 1' ->. Only these alarms are displayed. \\
- 2: Filter by alarms whose latch variable 1 contains the typed literal specified in "Filter variable" according to IEC 61131-3. Examples: T\#1h2s, DINT\#15, REAL\#1.5, FALSE \\
- 3: Filter by alarms whose latch variable 1 contains the LINT literal value specified in "Filter variable". Therefore, the value of the latch variables has to be in the range of \(9,223,372,036,854,775,808\) to 9,223,372,036,854,775,807. \\
- All other values: The behavior is not defined and can change in the future.
\end{tabular} \\
\hline "Filter by time range" & \begin{tabular}{l}
The generated alarms (remote, historical, local) can be displayed for a specified time range. You use the "Filter type" to define whether filtering by time range is enabled or disabled. \\
- "Filter variable, from": Variable of data type DT or DATE_AND_TIME (example: PLC_PRG.filterTimeFrom) for the start time that the alarms are displayed. \\
- "Filter variable, to": Variable of data type DT or DATE_AND_TIME (example: PLC_PRG.filterTimeTo) for the end time that the alarms are displayed. \\
- "Filter type": Variable of integer data type that determines whether "Filter by time range" is enabled or disabled. \\
Possible values: \\
- 1: Filtering is enabled \\
- 0 : Filtering is disabled
\end{tabular} \\
\hline
\end{tabular}

See also
- 乡y Chapter 6.4.5.21.3.17 "Dialog 'Selected Alarm Group"' on page 3390
- \(\Rightarrow\) Chapter 6.4.5.21.3.16 "Dialog 'Selected Alarm Class'" on page 3389

Element prop- By default, columns [0] and [1] are preconfigured: "Time stamp" and "Message text". You create erty 'Columns: Column [<n>]' more columns by clicking "Create new". You remove columns by clicking "Delete".
Animations (dynamic text, font variables), texts, and tooltips are not supported.
\begin{tabular}{|c|c|}
\hline "Width" & Width of the column (in pixels) \\
\hline "Type of data" & \begin{tabular}{l}
About time stamps: When used in a TargetVisu or WebVisu, you can control the date and time format by means of the global string variables from the library Alarmmanager.library: AlarmGlobals.g_sDateFormat (example: AlarmGlobals.g_sDateFormat := 'MM. уyyy') and AlarmGlobals.g_sTimeFormat (example: AlarmGlobals.g_sTimeFormat := 'HH:mm'). \\
Here you define the information to be displayed in the column. \\
- "Bitmap" \\
- "Time stamp": Date and time of the last status change of the alarm \\
- "Time stamp active": Date and time of the last activation of the alarm \\
- "Time stamp inactive": Date and time of the last deactivation of the alarm \\
- "Time stamp acknowledge": Date and time of the last acknowledgement \\
- "Value": Actual value of the expression \\
- "Message": Output of the message text \\
- "Priority": Alarm priority \\
- "Class": Alarm class \\
- "State": Alarm state \\
- "Latch Variable <n>": Value of the selected latch variables
\end{tabular} \\
\hline "Text alignment" & \begin{tabular}{l}
Alignment of the contents in the column \\
- "Left" \\
- "Centered" \\
- "Right"
\end{tabular} \\
\hline
\end{tabular}

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline "X" & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (品) to other positions in the editor.

See also
- \({ }^{\text {® }}\) Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center'
\begin{tabular}{|l|l|}
\hline\(" X "\) & X-coordinate of the point of rotation \\
\hline\(" Y "\) & \(Y\)-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus})\) to other positions in the editor.

Element prop- The properties contain fixed values for the text properties. erty 'Text properties'
\begin{tabular}{|c|c|}
\hline "Font" & Example: "Default"
\(\qquad\) : The "Font" dialog box opens.
Drop-down list with style fonts. \\
\hline "Font color" & \begin{tabular}{l}
Example: "Black" \\
...: The "Color" dialog box opens. \\
*: Drop-down list with style colors.
\end{tabular} \\
\hline "Transparency" & \begin{tabular}{l}
Whole number (value range from 0 to 255). This determines the transparency of the respective color. \\
Example: 255: The color is opaque. \\
0 : The color is completely transparent. \\
Please note: If the color is a style color and already has a transparency value, then this property is write-protected.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|l}
\hline "Acknowledge variable" & A rising edge of this variable acknowledges the alarm.
\end{tabular}

Handling of mul-
tiple active alarms
\begin{tabular}{|l|l|}
\hline "Automatic switch" & \begin{tabular}{l} 
V: The display in the alarm banner is \\
switched automatically according to the time \\
to the next alarm as configured in "Every \(N\) \\
second".
\end{tabular} \\
\hline "Every N second" & \begin{tabular}{l} 
Time period until the next switching. Available \\
only if "Automatic switch" is selected.
\end{tabular} \\
\hline "Next alarm" & \begin{tabular}{l} 
Variable for switching to the next alarm. Avail- \\
able only if "Automatic switch" is not selected.
\end{tabular} \\
\hline "Previous alarm" & \begin{tabular}{l} 
Variable for switching to the previous alarm. \\
Available only if "Automatic switch" is not \\
selected.
\end{tabular} \\
\hline "Multiple alarms active" & \begin{tabular}{l} 
Variable that has the value TRUE if multiple \\
alarms are active.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|c|c|c|}
\hline \multicolumn{3}{|l|}{"Movement"} \\
\hline "X" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular}} \\
\hline "Y" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the \(Y\) position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular}} \\
\hline "Rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle1. \\
The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol. \\
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
\end{tabular} &  \\
\hline "Interior rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle2. \\
In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. \\
The rotation point is shown as the \(\boldsymbol{\oplus}\) symbol. \\
Note: If a static angle of rotation is specified in the "Position \(\rightarrow\) Angle" property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.
\end{tabular} &  \\
\hline
\end{tabular}

You can link the variables to a unit conversion.


See also

Element prop- The variables control the element behavior dynamically. erty 'State variables'
\begin{tabular}{|l|l}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
\\
TRUE: The element is not visible at runtime.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", "X", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- Requirement: User management is set up for the visualization.
erty 'Access
rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages: \\
\\
\\
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
- "Rights are set: Limited rights": Access is restricted for at least one group. \\
\hline
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\Rightarrow\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights"' on page 3366

\section*{Visualization Element 'Bar Display'}

Symbol:

Category: "Measurement Controls"
The element displays the value of a variable.
See also
- ⓨ Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & Example: GenElemInst_2 \\
\hline "Type of element" & "Bar Display" \\
\hline "Value" & \begin{tabular}{l} 
Variable (numeric data type) \\
The value of the variable is displayed as a bar length.
\end{tabular} \\
\hline
\end{tabular}
\(\begin{array}{ll}\text { Element prop- } & \text { The properties contain fixed values for the coordinates of the point of rotation. This point of } \\ \text { erty 'Center' } & \text { rotation is shown as the } \boldsymbol{\oplus} \text { symbol. The point is used as the center for rotating and scaling. }\end{array}\)
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline " \(Y\) " & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus}\) ) to other positions in the editor.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline "X" & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (ए-) to other positions in the editor.

\section*{See also}
- \({ }^{\wedge} \Rightarrow\) Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop-
erty 'Back-
ground'
\begin{tabular}{|l|l|}
\hline "Background color" & \begin{tabular}{l} 
Drop-down list with background colors \\
Note: This property depends on the style. For example, there are no heterochro- \\
matic background images for "FlatStyle" and "Whitestyle".
\end{tabular} \\
\hline "Own image" & \begin{tabular}{l} 
- "image": Image ID of the background image. You select the background \\
image from an image pool by clicking the \\
Info: If you specify the "<default" " salue or select the image from the \\
"Default" category in the input assistant, then the original element back- \\
ground image is used. \\
-Transparent color": Color of pixels that are displayed as transparent. \\
Selection from drop-down list or input assistant.
\end{tabular} \\
\hline "Optimized drawing" & \begin{tabular}{l}
\(\square:\) The background image is drawn one time. If there is a change in the fore- \\
ground, then only the affected part of the image is redrawn. \\
\(\square\) \\
\(\square\)
\end{tabular} \\
\hline Note: Deactivating this option is sensible only in certain exceptional cases.
\end{tabular}

\section*{Element prop- \\ erty 'Bar'}
\begin{tabular}{|c|c|}
\hline "Diagram type" & \begin{tabular}{l}
Position of the scale \\
- "Scale besides bar" \\
- "Scale in bar" \\
- "Bar in scale" \\
- "No scale"
\end{tabular} \\
\hline "Orientation" & \begin{tabular}{l}
Orientation depending on the ratio of width to height of the Bar Display: \\
- "Horizontal" \\
- "Vertical"
\end{tabular} \\
\hline "Running direction" & \begin{tabular}{l}
Direction the values are increased. \\
Drop-down list for "Orientation Horizontal": \\
- "Left to right" \\
- "Right to left" \\
Drop-down list for "Orientation Vertical": \\
- "Bottom to top" \\
- "Top to bottom"
\end{tabular} \\
\hline "Optimum size for bar" & \begin{tabular}{l}
The bar width requires the majority of the element surface. \\
Note: This property depends on the style. It is not provided for "FlatStyle" or "WhiteStyle".
\end{tabular} \\
\hline
\end{tabular}

\section*{Element prop- \\ erty 'Scale'}
\begin{tabular}{|c|c|}
\hline "Scale start" & \begin{tabular}{l}
Least value of the scale and the lower limit of the value range for the element. \\
Example: 0 \\
*: The property "Variable" is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the scale start. \\
Example: PLC_PRG.iScaleStart \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iScaleStart : INT := 0;
END_VAR

```
\end{tabular} \\
\hline "Scale end" & \begin{tabular}{l}
Greatest value of the scale and the upper limit of the value range for the element. \\
Example: 100 \\
*: The property "Variable" is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the scale end. \\
Example: PLC_PRG.iScaleEnd \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iScaleEnd : INT := 120;
END_VAR

```
\end{tabular} \\
\hline "Main scale" & \begin{tabular}{l}
Distance between 2 values on the rough scale. \\
Example: 10 \\
*: The property "Variable" is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the distance. \\
Example: PLC_PRG.iMainScale \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iMainScale : INT := 20;
END_VAR

```
\end{tabular} \\
\hline "Subscale" & \begin{tabular}{l}
Distance between 2 values on the fine scale. \\
You can hide the fine scale by setting the value to 0 . \\
Example: 2 \\
*: The property "Variable" is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the spacing. \\
Example: PLC_PRG.iSubScale \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iSubScale : INT := 5;
END_VAR

```
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Scale line width" & \begin{tabular}{l}
Specified in pixels. \\
Example: 3
\end{tabular} \\
\hline "Scale color" & \begin{tabular}{l}
Color of scale lines \\
- ...: \\
The "Color" dialog box opens. \\
- \(\quad\) : A drop-down list with color names opens.
\end{tabular} \\
\hline "Scale in 3D" & \begin{tabular}{l}
Tick marks are displayed with slight 3D shadowing. \\
Note: This property depends on the style. Not available for "FlatStyle".
\end{tabular} \\
\hline "Element frame" & \(\checkmark\) : A frame is drawn around the element. \\
\hline
\end{tabular}

Element property 'Label'
\begin{tabular}{|l|l|}
\hline "Unit" & \begin{tabular}{l} 
Text that is displayed in the element. \\
Example: Units displayed in m/s.
\end{tabular} \\
\hline "Font" & \begin{tabular}{l} 
Font for labels (example: scale numbering). \\
Selection from the drop-down list or by clicking the "" button. \\
\hline "...
\end{tabular} \\
\hline "Mcale format (C Syntax)" text width of labels" & \begin{tabular}{l} 
Values scaled in "printf" syntax \\
Examples: \%d, \%5.2f
\end{tabular} \\
\hline "Text height of labels" & \begin{tabular}{l} 
(optional) Value that redefines the maximum width of the scale label. The correct \\
value is normally set automatically. \\
Note: Change this value only if the automatic adjustment does not yield the \\
expected result.
\end{tabular} \\
\hline "Font color" & \begin{tabular}{l} 
(optional) Value that redefines the maximum height of the scale label. The cor- \\
rect value is normally set automatically. \\
Note: Change this value only if the automatic adjustment does not yield the \\
expected result.
\end{tabular} \\
\hline
\end{tabular}

\section*{Element prop-}
erty 'Positio-
ning'
\begin{tabular}{|l|l|}
\hline "Horizontal offset" & \begin{tabular}{l} 
Distance from the scale (bar) to the horizontal element frame \\
Specified in pixels. \\
Used for achieving the exact position relative to the background image.
\end{tabular} \\
\hline "Vertical offset" & \begin{tabular}{l} 
Distance from the scale (bar) to the vertical element frame \\
Specified in pixels. \\
Used for achieving the exact position relative to the background image.
\end{tabular} \\
\hline "Horizontal scaling" & \begin{tabular}{l} 
Horizontal division of the scale \\
Specified in pixels. \\
Used for achieving the exact positioning relative to the background image.
\end{tabular} \\
\hline "Vertical scaling" & \begin{tabular}{l} 
Vertical division of the scale \\
Specified in pixels. \\
Used for achieving the exact positioning relative to the background image.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Colors'
\begin{tabular}{|c|c|}
\hline "Graph color" & Color of the bar \\
\hline "Bar background" & The background of the bar is black.
The background of the bar is white. \\
\hline "Frame color" & Color that the frames are drawn. \\
\hline "Switch whole color" & The total color of the bar is switched to the color of the color area of the current value. \\
\hline "Use gradient color for bar" & - Bar is displayed with a gradient. \\
\hline "Color range markers" & \begin{tabular}{l}
The color areas can be separated from each other inside the bar with a vertical mark. \\
- "No markers": No display. \\
- "Marker forwards": The color of the vertical mark corresponds to the color of the previous color area. \\
- "Marker backwards": The color of the vertical mark corresponds to the color of the next color area.
\end{tabular} \\
\hline \multicolumn{2}{|l|}{"Color areas"} \\
\hline "Create new" & A new color area is added. \\
\hline "Delete" & The color area is removed from the list. \\
\hline "Begin of area" & Start value of the color area \\
\hline "End of area" & End value of the color area \\
\hline "Color" & Color that is used for displaying the area. \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" & \multicolumn{1}{l|}{\begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular}} \\
\hline " \(Y\) " & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG.i Pos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\boldsymbol{\rightarrow}\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\boldsymbol{\oplus}\) symbol.
\end{tabular} \\
\begin{tabular}{l} 
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
\hline
\end{tabular}

You can link the variables to a unit conversion.

The "X", " \(Y\) ", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also

Element prop- The variables control the element behavior dynamically. erty 'State varia-
bles'
\begin{tabular}{|l|l|}
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

Element prop- Requirement: User management is set up for the visualization. erty 'Access
rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages:
\end{tabular} \\
& - "Not set. Full rights.": Access rights for all user groups : "operable" \\
- "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular}

See also
- \({ }^{\#}\) Chapter 6.4.5.21.3.1 "Dialog 'Access Rights"' on page 3366

\section*{Visualization Element 'Meter 900'}

Symbol:


\section*{Category: "Measurement Controls"}

The element displays the value of a variable. The needle is positioned according to the value of the assigned variable. A meter is used to represent a tachometer, for example.

\section*{Element proper- \\ ties}
\begin{tabular}{|l|l|}
\hline "Element name" & Example: GenElemInst_1 \\
\hline "Type of element" & "Meter \(90^{\circ} "\) \\
\hline "Value" & \begin{tabular}{l} 
Variable (numeric data type) \\
The variable value determines the pointer direction of the element.
\end{tabular} \\
\hline
\end{tabular}

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline "X" & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (■-) to other positions in the editor.

See also
- Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

\section*{Element prop- \\ erty 'Back- \\ ground'}
\begin{tabular}{|l|l|}
\hline "Image color" & List box containing background colors \\
\hline "Own image" & \begin{tabular}{l} 
- "Image": ID of the background image. \\
\\
\\
\\
\\
\\
\\
\\
You select the background image from an image pool by clicking \begin{tabular}{l} 
Info: If you specify the value "<default>" or select the image from the \\
"Default" category in the Input Assistant, then the original element back- \\
ground image is used. \\
- "Transparency color": Selection from list box or Input Assistant.
\end{tabular} \\
\hline
\end{tabular} \\
\hline
\end{tabular}

\section*{Element prop- \\ erty 'Arrow'}
\begin{tabular}{|l|l|}
\hline "Hand style" & Drop-down list with different arrow types \\
\hline "Color" & -...\(:\) The "Color" dialog box opens. \\
& - : Drop-down list with color names \\
\hline "Angle range" & Drop-down list for the alignment of the element \\
\hline "Additional arrow" & : An additional arrow is shown inside the scale. \\
\hline
\end{tabular}

\section*{Element property 'Scale'}
\begin{tabular}{|l|l|}
\hline "Sub scale position" & \begin{tabular}{l} 
- "Outside": The subscale is displayed on the outer scale ring. ("Frame \\
outside") "Inside": The subscale is displayed on the inner scale ring. ("Frame inside")
\end{tabular} \\
\hline "Scale type" & \begin{tabular}{l} 
Type of scale \\
- "Lines" \\
- "Dots" \\
- "Squares"
\end{tabular} \\
\hline "Scale start" & \begin{tabular}{l} 
Least value of the scale and the lower limit of the value range for the element \\
Example: 0 \\
: The "Variable" property is displayed in the line below this.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (integer data type). Contains the scale start \\
Example: PLC_PRG. iScaleStart \\
Declaration:
\end{tabular} \\
\hline "Scale end" & \begin{tabular}{l} 
PROGRAM PLC_PRG \\
VAR \\
END_VAR iScaleStart : INT : \(\quad 0 ;\)
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Greatest value of the scale and the upper limit of the value range for the element \\
Example: 100 \\
: The "Variable" property is shown below this.
\end{tabular} \\
\hline Variable (integer data type). Contains the scale end \\
Example: PLC_PRG. iScaleEnd \\
Declaration:
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type) Contains the distance between two values on the fine scale \\
Example: PLC_PRG.iSubScale \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iSubScale : INT := 5;
END_VAR

```
\end{tabular} \\
\hline "Scale line width" & \begin{tabular}{l}
Specified in pixels \\
Example: 3
\end{tabular} \\
\hline "Scale color" & \begin{tabular}{l}
Color of scale lines The "Color" dialog opens. \\
- - : A list box with style colors opens
\end{tabular} \\
\hline "Scale in 3D" & Scale lines are displayed with soft 3D shadowing. Note: This property is not displayed in "FlatStyle". \\
\hline "Show scale" & V: The scale is displayed. \\
\hline "Frame inside" & \(\square\) : A frame is drawn at the inner end of the scale. \\
\hline "Frame outside" & V: A frame is drawn at the outer end of the scale. \\
\hline
\end{tabular}

Element prop-
erty 'Label'
\begin{tabular}{|l|l|}
\hline "Label" & \begin{tabular}{l} 
Selection list \\
- "Outside": Scale values are placed outside of the scale. \\
- "Inside": Scale values are placed inside of the scale.
\end{tabular} \\
\hline "Unit" & \begin{tabular}{l} 
Text that is displayed in the element. \\
Example: Units displayed in m/s.
\end{tabular} \\
\hline "Font" & \begin{tabular}{l} 
Font for labels (example: scale numbering). \\
Selection from the drop-down list or by clicking the "" button. ...
\end{tabular} \\
\hline "Scale format (C Syntax)" & \begin{tabular}{l} 
Values scaled in "printf" syntax \\
Examples: \%d, \%5.2f
\end{tabular} \\
\hline "Max. text width of labels" & \begin{tabular}{l} 
(optional) Value that redefines the maximum width of the scale label. The correct \\
value is normally set automatically. \\
Note: Change this value only if the automatic adjustment does not yield the \\
expected result.
\end{tabular} \\
\hline "Text height of labels" & \begin{tabular}{l} 
(optional) Value that redefines the maximum height of the scale label. The cor- \\
rect value is normally set automatically. \\
Note: Change this value only if the automatic adjustment does not yield the \\
expected result.
\end{tabular} \\
\hline "Font color" & Selection from the drop-down list or by clicking the ■... button. \\
\hline
\end{tabular}

\section*{Element property 'Positioning'}
\begin{tabular}{|c|c|}
\hline "Usage of" & \begin{tabular}{l}
- "Preset style values": Values from the current style \\
- "User-defined settings": The subnode "Positioning" appears.
\end{tabular} \\
\hline \begin{tabular}{l}
"Positioning" \\
Requirement: "User- \\
The displayed positio whether a custom ba relative to the backgrour
\end{tabular} & \begin{tabular}{l}
tings" is selected as "Usage of". \\
gs depend on the type of needle instrument and Potentiometer, and partially on mage is selected. The following settings are used for achieving the exact position
\end{tabular} \\
\hline "Needle movement" & Length of the needle (in pixels) \\
\hline "Scale movement" & \begin{tabular}{l}
Distance from the tick marks to the center (in pixels) \\
Requirement: A customer image is selected as "Background".
\end{tabular} \\
\hline "Scale length" & \begin{tabular}{l}
Length of the tick marks (in pixels) \\
Requirement: A customer image is selected as "Background".
\end{tabular} \\
\hline "Label offset": & Distance from the labels to the tick marks (in pixels) \\
\hline "Unit offset": & Distance of the unit text "Label \(\rightarrow\) Unit" from the upper scale edge (in pixels) \\
\hline "Origin offset" & \begin{tabular}{l}
Offset of the element (in pixels) \\
Requirement: For the elements "Meter \(180^{\circ}\) " and "Meter \(90^{\circ}\) ", this property is displayed only if a custom image is selected as "Background".
\end{tabular} \\
\hline
\end{tabular}

\section*{Element prop- \\ erty 'Colors'}
\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|l|}{"Color areas"} \\
\hline "Durable color areas" & All color areas are visible, regardless of the current value.
Only the color area is visible that includes the current value. \\
\hline "Use colors for scale" & 回: Colors in the color area are used only for the scale and frame. \\
\hline "Color areas" & \\
\hline "Create new" & A new color area is added to the "Elements" view. \\
\hline "Delete" & The color area is removed from the list and the list is refreshed. \\
\hline "Begin of area" & \begin{tabular}{l}
Start value of the color area \\
Example: 20 \\
*: The property "Variable" is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the start value. \\
Example: PLC_PRG.iColorAreaStart0 \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iColorAreaStart0 : INT := 80;
END_VAR

```
\end{tabular} \\
\hline "End of area" & \begin{tabular}{l}
End value of the color area \\
Example: 120 \\
: The property "Variable" is shown below.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Variable" & Variable (integer data type). Contains the end value. \\
& Example: iColorAreaEndo \\
& Declaration: \\
& \begin{tabular}{l} 
PROGRAM PLC_PRG \\
VAR iColorAreaEndo : INT \(:=100 ;\) \\
\\
\\
End_VAR
\end{tabular} \\
\hline "Color" & Color that is used for displaying the area. \\
\hline
\end{tabular} erty 'Absolute movement'

Element prop- The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG. iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\oplus\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
\hline "Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2. \\
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\oplus\) symbol. \\
Note: If a static angle of rotation is specified in the \\
"Position \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular} \\
\hline
\end{tabular}

\section*{You can link the variables to a unit conversion.}

Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages: \\
\\
\\
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
- "Rights are set: Limited rights": Access is restricted for at least one group. \\
\hline
\end{tabular} \\
\hline
\end{tabular}

See also


See also
- © Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872""

\section*{Visualization Element 'Meter \(\mathbf{1 8 0}^{\circ}\)}

Symbol:
The " \(X\) ", " \(Y\) ", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also
\(\bullet\)
- Chapter 6.4.5.21.3.1 'Dial 'Access Rign' on page 336 .
- Chapter 6.4.5.5 'Dasiging a visualization wit element"on pag 2872


\section*{Category: "Measurement Controls"}

The element displays the value of a variable. The needle is positioned according to the value of the assigned variable on a scale. A meter is used to represent a tachometer, for example.

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & Example: GenElemInst_1 \\
\hline "Type of element" & "Meter \(180^{\circ} "\) \\
\hline "Value" & \begin{tabular}{l} 
Variable (numeric data type) \\
The variable value determines the pointer direction of the element.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center' rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline " \(Y\) " & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus}\) ) to other positions in the editor.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The position defines the location and size of the element in the visualization window. These erty 'Position' are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X\) " & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (마) to other positions in the editor.

See also
- * Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop-
erty 'Back-
ground'
\begin{tabular}{|l|l|}
\hline "Image color" & List box containing background colors \\
\hline "Own image" & - "Image": ID of the background image. \\
& \begin{tabular}{l} 
You select the background image from an image pool by clicking \\
Info: If you specify the value "<default>" or select the image from the \\
"Defautt" category in the Input Assistant, then the original element back- \\
ground image is used. \\
\\
\\
\\
- "Transparency color": Selection from list box or Input Assistant.
\end{tabular} \\
\hline
\end{tabular}

\section*{Element prop-}
erty 'Arrow'
\begin{tabular}{|l|l|}
\hline "Hand style" & Drop-down list with different arrow types \\
\hline "Color" & - \(\ldots\) : The "Color" dialog box opens. \\
& - : Drop-down list with color names \\
\hline "Angle range" & Drop-down list for the alignment of the element \\
\hline "Additional arrow" & 目: An additional arrow is shown inside the scale. \\
\hline
\end{tabular}

\section*{Element prop-}
erty 'Scale'
\begin{tabular}{|l|l|}
\hline "Sub scale position" & \begin{tabular}{l} 
- "Outside": The subscale is displayed on the outer scale ring. ("Frame \\
outside") \\
- "Inside": The subscale is displayed on the inner scale ring. ("Frame inside")
\end{tabular} \\
\hline "Scale type" & \begin{tabular}{l} 
Type of scale \\
- "Lines" \\
- "Dots" \\
- "Squares"
\end{tabular} \\
\hline "Scale start" & \begin{tabular}{l} 
Least value of the scale and the lower limit of the value range for the element \\
Example: 0 \\
": The "Variable" property is displayed in the line below this.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the scale start \\
Example: PLC_PRG.iScaleStart \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iScaleStart : INT := 0;
END_VAR

```
\end{tabular} \\
\hline "Scale end" & \begin{tabular}{l}
Greatest value of the scale and the upper limit of the value range for the element Example: 100 \\
*: The "Variable" property is shown below this.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the scale end \\
Example: PLC_PRG.iScaleEnd \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iScaleEnd : INT := 120;
END_VAR

```
\end{tabular} \\
\hline "Main scale" & \begin{tabular}{l}
Distance between two values on the main scale Example: 10 \\
*: The "Variable" property is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type) Contains the distance between two values on the main scale \\
Example: PLC_PRG.iMainScale \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iMainScale : INT := 20;
END_VAR

```
\end{tabular} \\
\hline "Sub scale" & \begin{tabular}{l}
Distance between two values on the fine scale You can hide the fine scale by setting the value to 0 . Example: 2 \\
*: The "Variable" property is shown below this.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type) Contains the distance between two values on the fine scale \\
Example: PLC_PRG.iSubScale \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iSubScale : INT := 5;
END_VAR

```
\end{tabular} \\
\hline "Scale line width" & \begin{tabular}{l}
Specified in pixels \\
Example: 3
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Scale color" & \begin{tabular}{l}
Color of scale lines
: The "Color" dialog opens. \\
- - A list box with style colors opens.
\end{tabular} \\
\hline "Scale in 3D" & Scale lines are displayed with soft 3D shadowing. Note: This property is not displayed in "FlatStyle". \\
\hline "Show scale" & \(\square\) : The scale is displayed. \\
\hline "Frame inside" & \(\square\) : A frame is drawn at the inner end of the scale. \\
\hline "Frame outside" & - A frame is drawn at the outer end of the scale. \\
\hline
\end{tabular}

Element property 'Label'
\begin{tabular}{|c|c|}
\hline "Label" & \begin{tabular}{l}
Selection list \\
- "Outside": Scale values are placed outside of the scale. \\
- "Inside": Scale values are placed inside of the scale.
\end{tabular} \\
\hline "Unit" & \begin{tabular}{l}
Text that is displayed in the element. \\
Example: Units displayed in \(\mathrm{m} / \mathrm{s}\).
\end{tabular} \\
\hline "Font" & \begin{tabular}{l}
Font for labels (example: scale numbering). \\
Selection from the drop-down list or by clicking the "" button.
\end{tabular} \\
\hline "Scale format (C Syntax)" & \begin{tabular}{l}
Values scaled in "printf" syntax \\
Examples: \%d, \(\% 5.2 f\)
\end{tabular} \\
\hline "Max. text width of labels" & \begin{tabular}{l}
(optional) Value that redefines the maximum width of the scale label. The correct value is normally set automatically. \\
Note: Change this value only if the automatic adjustment does not yield the expected result.
\end{tabular} \\
\hline "Text height of labels" & \begin{tabular}{l}
(optional) Value that redefines the maximum height of the scale label. The correct value is normally set automatically. \\
Note: Change this value only if the automatic adjustment does not yield the expected result.
\end{tabular} \\
\hline "Font color" & Selection from the drop-down list or by clicking the ... button. \\
\hline
\end{tabular}

\section*{Element prop- \\ erty 'Positio- \\ ning'}
\begin{tabular}{|l|l|}
\hline "Usage of" & - "Preset style values": Values from the current style \\
- "User-defined settings": The subnode "Positioning" appears.
\end{tabular}\(|\)\begin{tabular}{l|l|}
\hline \begin{tabular}{l} 
Requirement: "User-defined settings" is selected as "Usage of". \\
The displayed positioning settings depend on the type of needle instrument and Potentiometer, and partially on \\
whether a custom background image is selected. The following settings are used for achieving the exact position \\
relative to the background image.
\end{tabular} \\
\hline "Needle movement" & Length of the needle (in pixels) \\
\hline "Scale movement" & \begin{tabular}{l} 
Distance from the tick marks to the center (in pixels) \\
Requirement: A customer image is selected as "Background".
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Scale length" & \begin{tabular}{l} 
Length of the tick marks (in pixels) \\
Requirement: A customer image is selected as "Background".
\end{tabular} \\
\hline "Label offset": & Distance from the labels to the tick marks (in pixels) \\
\hline "Unit offset": & Distance of the unit text "Label \(\rightarrow\) Unit" from the upper scale edge (in pixels) \\
\hline "Origin offset" & \begin{tabular}{l} 
Offset of the element (in pixels) \\
Requirement: For the elements "Meter \(180^{\circ} "\) and "Meter \(90{ }^{\circ} ", ~ t h i s ~ p r o p e r t y ~ i s ~\) \\
displayed only if a custom image is selected as "Background".
\end{tabular} \\
\hline
\end{tabular}

Element prop-
erty 'Colors'
\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|l|}{"Color areas"} \\
\hline "Durable color areas" & All color areas are visible, regardless of the current value.
Only the color area is visible that includes the current value. \\
\hline "Use colors for scale" & \(\checkmark\) : Colors in the color area are used only for the scale and frame. \\
\hline "Color areas" & \\
\hline "Create new" & A new color area is added to the "Elements" view. \\
\hline " Delete" & The color area is removed from the list and the list is refreshed. \\
\hline "Begin of area" & \begin{tabular}{l}
Start value of the color area \\
Example: 20 \\
*: The property "Variable" is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the start value. \\
Example: PLC_PRG.iColorAreaStart0 \\
Declaration: \\
PROGRAM PLC_PRG \\
VAR \\
iColorAreaStart0 : INT := 80;
\end{tabular} \\
\hline "End of area" & \begin{tabular}{l}
End value of the color area \\
Example: 120 \\
*: The property "Variable" is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the end value. \\
Example: iColorAreaEndo \\
Declaration: \\
PROGRAM PLC_PRG \\
VAR \\
iColorAreaEnd0 : INT := 100;
\end{tabular} \\
\hline "Color" & Color that is used for displaying the area. \\
\hline
\end{tabular}

\section*{Element property 'Absolute movement'}

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|c|c|c|}
\hline \multicolumn{3}{|l|}{"Movement"} \\
\hline " \(\times\) " & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular}} \\
\hline " \(Y\) " & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the \(Y\) position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular}} \\
\hline "Rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle1. \\
The midpoint of the element rotates at the "Center" point. This rotation point is shown as the \(\boldsymbol{\oplus}\) symbol. \\
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
\end{tabular} &  \\
\hline "Interior rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle2. \\
In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. \\
The rotation point is shown as the \(\boldsymbol{\oplus}\) symbol. \\
Note: If a static angle of rotation is specified in the "Position \(\rightarrow\) Angle" property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.
\end{tabular} &  \\
\hline
\end{tabular}

You can link the variables to a unit conversion.


See also

Element prop- The variables control the element behavior dynamically. erty 'State variables'
\begin{tabular}{|l|l|}
\hline "Invisible" & Variable (BOOL). Toggles the visibility of the element. \\
& TRUE: The element is not visible at runtime.
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

Element prop- Requirement: User management is set up for the visualization. erty 'Access
rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages:
\end{tabular} \\
& - "Not set. Full rights.": Access rights for all user groups : "operable" \\
- "Rights are set: Limited rights": Access is restricted for at least one group. \\
\hline
\end{tabular}

See also
- «\& Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

See also
- \(\boldsymbol{y y}^{4}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Meter'}

Symbol:


\section*{Category: "Measurement Controls"}

The element displays the value of a variable. The needle is positioned according to the value of the assigned variable. A meter is used to represent a tachometer, for example.

\section*{Element proper-}
ties
\begin{tabular}{|l|l|}
\hline "Element name" & Example: GenElemInst_1 \\
\hline "Type of element" & "Meter" \\
\hline "Value" & \begin{tabular}{l} 
Variable (numeric data type). \\
The variable value determines the pointer direction of the element.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center' rotation is shown as the \(\oplus\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline " \(Y\) " & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus})\) to other positions in the editor.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X\) " & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}
You can also change the values by dragging the box symbols (吊) to other
positions in the editor.

See also
- © Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop-
erty 'Back-
ground'
\begin{tabular}{|l|l|}
\hline "Image color" & List box containing background colors \\
\hline "Own image" & \begin{tabular}{l} 
- "Image": ID of the background image. \\
\\
\\
\\
\\
\\
\\
\\
You select the background image from an image pool by clicking \begin{tabular}{l} 
Inou specify the value "<default>" or select the image from the \\
"Default" category in the Input Assistant, then the original element back- \\
ground image is used. \\
- "Transparency color": Selection from list box or Input Assistant.
\end{tabular} \\
\hline
\end{tabular} \\
\hline
\end{tabular}

\section*{Element prop-}
erty 'Arrow'
\begin{tabular}{|l|l|}
\hline "Hand style" & Drop-down list with different arrow types \\
\hline "Color" & - ...: The "Color" dialog box opens. \\
- : Drop-down list with color names
\end{tabular}.

\section*{Element prop- \\ erty 'Scale'}
\begin{tabular}{|l|l|}
\hline "Sub scale position" & \begin{tabular}{l} 
- "Outside": The subscale is displayed on the outer scale ring. ("Frame \\
outside") \\
- "Inside": The subscale is displayed on the inner scale ring. ("Frame inside")
\end{tabular} \\
\hline "Scale type" & \begin{tabular}{l} 
Type of scale \\
- "Lines" \\
- "Dots" \\
- "Squares"
\end{tabular} \\
\hline "Scale start" & \begin{tabular}{l} 
Least value of the scale and the lower limit of the value range for the element \\
Example: 0 \\
": The "Variable" property is displayed in the line below this.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the scale start \\
Example: PLC_PRG.iScaleStart \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iScaleStart : INT := 0;
END_VAR

```
\end{tabular} \\
\hline "Scale end" & \begin{tabular}{l}
Greatest value of the scale and the upper limit of the value range for the element Example: 100 \\
: The "Variable" property is shown below this.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the scale end \\
Example: PLC_PRG.iScaleEnd \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iScaleEnd : INT := 120;
END_VAR

```
\end{tabular} \\
\hline "Main scale" & \begin{tabular}{l}
Distance between two values on the main scale \\
Example: 10 \\
*: The "Variable" property is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type) Contains the distance between two values on the main scale \\
Example: PLC_PRG.iMainScale \\
Declaration: \\
PROGRAM PLC_PRG \\
VAR \\
iMainScale : INT := 20; \\
END VAR
\end{tabular} \\
\hline "Sub scale" & \begin{tabular}{l}
Distance between two values on the fine scale You can hide the fine scale by setting the value to 0 . Example: 2 \\
*: The "Variable" property is shown below this.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type) Contains the distance between two values on the fine scale \\
Example: PLC_PRG.iSubScale \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iSubScale : INT := 5;
END_VAR

```
\end{tabular} \\
\hline "Scale line width" & \begin{tabular}{l}
Specified in pixels \\
Example: 3
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Scale color" & \begin{tabular}{l}
Color of scale lines \\
The "Color" dialog opens. \\
- - : A list box with style colors opens.
\end{tabular} \\
\hline "Scale in 3D" & Scale lines are displayed with soft 3D shadowing. Note: This property is not displayed in "FlatStyle". \\
\hline "Show scale" & V: The scale is displayed. \\
\hline "Frame inside" & \(\square\) : A frame is drawn at the inner end of the scale. \\
\hline "Frame outside" & - A frame is drawn at the outer end of the scale. \\
\hline
\end{tabular}

Element property 'Label'
\begin{tabular}{|c|c|}
\hline "Label" & \begin{tabular}{l}
Selection list \\
- "Outside": Scale values are placed outside of the scale. \\
- "Inside": Scale values are placed inside of the scale.
\end{tabular} \\
\hline "Unit" & \begin{tabular}{l}
Text that is displayed in the element. \\
Example: Units displayed in \(\mathrm{m} / \mathrm{s}\).
\end{tabular} \\
\hline "Font" & \begin{tabular}{l}
Font for labels (example: scale numbering). \\
Selection from the drop-down list or by clicking the "" button.
\end{tabular} \\
\hline "Scale format (C Syntax)" & \begin{tabular}{l}
Values scaled in "printf" syntax \\
Examples: \%d, \(\% 5.2 f\)
\end{tabular} \\
\hline "Max. text width of labels" & \begin{tabular}{l}
(optional) Value that redefines the maximum width of the scale label. The correct value is normally set automatically. \\
Note: Change this value only if the automatic adjustment does not yield the expected result.
\end{tabular} \\
\hline "Text height of labels" & \begin{tabular}{l}
(optional) Value that redefines the maximum height of the scale label. The correct value is normally set automatically. \\
Note: Change this value only if the automatic adjustment does not yield the expected result.
\end{tabular} \\
\hline "Font color" & Selection from the drop-down list or by clicking the ... button. \\
\hline
\end{tabular}

\section*{Element prop- \\ erty 'Positio- \\ ning'}
\begin{tabular}{|l|l|}
\hline "Usage of" & - "Preset style values": Values from the current style \\
- "User-defined settings": The subnode "Positioning" appears.
\end{tabular}\(|\)\begin{tabular}{l|l|}
\hline \begin{tabular}{l} 
Requirement: "User-defined settings" is selected as "Usage of". \\
The displayed positioning settings depend on the type of needle instrument and Potentiometer, and partially on \\
whether a custom background image is selected. The following settings are used for achieving the exact position \\
relative to the background image.
\end{tabular} \\
\hline "Needle movement" & \multicolumn{2}{|l|}{ Length of the needle (in pixels) } \\
\hline "Scale movement" & \begin{tabular}{l} 
Distance from the tick marks to the center (in pixels) \\
Requirement: A customer image is selected as "Background".
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Scale length" & \begin{tabular}{l} 
Length of the tick marks (in pixels) \\
Requirement: A customer image is selected as "Background".
\end{tabular} \\
\hline "Label offset": & Distance from the labels to the tick marks (in pixels) \\
\hline "Unit offset": & Distance of the unit text "Label \(\rightarrow\) Unit" from the upper scale edge (in pixels) \\
\hline "Origin offset" & \begin{tabular}{l} 
Offset of the element (in pixels) \\
Requirement: For the elements "Meter \(180^{\circ} "\) and "Meter \(900^{\circ} ", ~ t h i s ~ p r o p e r t y ~ i s ~\) \\
displayed only if a custom image is selected as "Background".
\end{tabular} \\
\hline
\end{tabular}

Element prop-
erty 'Colors'
\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|l|}{"Color areas"} \\
\hline "Durable color areas" & All color areas are visible, regardless of the current value.
Only the color area is visible that includes the current value. \\
\hline "Use colors for scale" & \(\checkmark\) : Colors in the color area are used only for the scale and frame. \\
\hline "Color areas" & \\
\hline "Create new" & A new color area is added to the "Elements" view. \\
\hline " Delete" & The color area is removed from the list and the list is refreshed. \\
\hline "Begin of area" & \begin{tabular}{l}
Start value of the color area \\
Example: 20 \\
*: The property "Variable" is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the start value. \\
Example: PLC_PRG.iColorAreaStart0 \\
Declaration: \\
PROGRAM PLC_PRG \\
VAR \\
iColorAreaStart0 : INT := 80;
\end{tabular} \\
\hline "End of area" & \begin{tabular}{l}
End value of the color area \\
Example: 120 \\
*: The property "Variable" is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the end value. \\
Example: iColorAreaEndo \\
Declaration: \\
PROGRAM PLC_PRG \\
VAR \\
iColorAreaEnd0 : INT := 100;
\end{tabular} \\
\hline "Color" & Color that is used for displaying the area. \\
\hline
\end{tabular}

\section*{Element property 'Absolute movement'}

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|c|c|c|}
\hline \multicolumn{3}{|l|}{"Movement"} \\
\hline "X" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular}} \\
\hline "Y" & \multicolumn{2}{|l|}{\begin{tabular}{l}
Variable (numeric data type). Defines the \(Y\) position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular}} \\
\hline "Rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle1. \\
The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol. \\
In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right.
\end{tabular} &  \\
\hline "Interior rotation" & \begin{tabular}{l}
Variable (numeric data type). Defines the angle of rotation (in degrees). \\
Example: PLC_PRG.iAngle2. \\
In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. \\
The rotation point is shown as the \(\boldsymbol{\oplus}\) symbol. \\
Note: If a static angle of rotation is specified in the "Position \(\rightarrow\) Angle" property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed.
\end{tabular} &  \\
\hline
\end{tabular}

You can link the variables to a unit conversion.


See also

Element prop- The variables control the element behavior dynamically. erty 'State variables'
\begin{tabular}{|l|l}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
\\
TRUE: The element is not visible at runtime.
\end{tabular} \\
\hline
\end{tabular}

The "Invisible" property is supported by the "Client Animation" functionality.

Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages:
\end{tabular} \\
& - "Not set. Full rights.": Access rights for all user groups : "operable"" \\
- "Rights are set: Limited rights": Access is restricted for at least one group. \\
\hline
\end{tabular}

See also
- \& Chapter 6.4.5.21.3.1 "Dialog 'Access Rights"' on page 3366

See also
- \& Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Visualization Element 'Potentiometer'}

Symbol:


Category: "Measurement Controls"
The element displays the value of a variable as a setting on the potentiometer. A visualization user can modify the value by dragging the pointer to another position.
See also
- \({ }^{4}\) Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

\section*{Element proper- \\ ties}
\begin{tabular}{|l|l|}
\hline "Element name" & Example: GenElemInst_1 \\
\hline "Type of element" & "Potentiometer" \\
\hline "Variable" & \begin{tabular}{l} 
Variable (numeric data type). Contains the position of the pointer for the potenti- \\
ometer. \\
A visualization user can modify the value by dragging the pointer to another \\
position.
\end{tabular} \\
\hline
\end{tabular}

Element prop-
erty 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the \(\boldsymbol{\oplus}\) symbol. The point is used as the center for rotating and scaling.
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline "Y" & Y-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols ( \(\boldsymbol{\Psi})\) to other positions in the editor.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.
\begin{tabular}{|c|c|}
\hline "Animation duration" & \begin{tabular}{l}
Defines the duration (in milliseconds) in which the element runs an animation \\
- Variable (integer value) \\
Example: Menu.tContent with VAR tContent : INT := 500; END_VAR \\
- Integer literal \\
Example: 500 \\
Animatable properties \\
- "Absolute movement", "Movement", " \(X\) ", " \(Y\) " \\
- "Absolute movement", "Rotation" \\
- "Absolute movement", "Interior rotation" \\
- "Absolute movement", "Exterior rotation" \\
The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth.
\end{tabular} \\
\hline "Move to foreground" & \begin{tabular}{l}
Moves the visualization element to the foreground \\
Variable (BOOL) \\
Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR \\
TRUE: At runtime, the visualization element is displayed in the foreground. \\
FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor.
\end{tabular} \\
\hline
\end{tabular}

Element prop- The position defines the location and size of the element in the visualization window. These erty 'Position' of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline "X" & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (■-) to other positions in the editor.

See also
- Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

\section*{Element prop- \\ erty 'Back- \\ ground'}
\begin{tabular}{|l|l|}
\hline "Image color" & List box containing background colors \\
\hline "Own image" & \begin{tabular}{l} 
- "Image": ID of the background image. \\
\\
\\
\\
\\
\\
\\
\\
\\
\\
You select the background image from an image pool by clicking you specify the value "<default>" or select the image from the \\
"Defautl" category in the Input Assistant, then the original element back- \\
ground image is used. \\
- "Transparency color": Selection from list box or Input Assistant.
\end{tabular} \\
\hline
\end{tabular}

\section*{Element prop- \\ erty 'Arrow'}
\begin{tabular}{|c|c|}
\hline "Hand style" & Drop-down list with different arrow types \\
\hline "Color" & \begin{tabular}{l}
- ...: The "Color" dialog box opens. \\
- - Drop-down list with color names
\end{tabular} \\
\hline "Arrow start" & Angle (in degrees) between the left edge of the element and the horizontal axis \\
\hline "Arrow end" & Angle (in degrees) between the right edge of the element and the horizontal axis \\
\hline
\end{tabular}

\section*{Element property 'Scale'}
\begin{tabular}{|c|c|}
\hline "Sub scale position" & \begin{tabular}{l}
- "Outside": The subscale is displayed on the outer scale ring. ("Frame outside") \\
- "Inside": The subscale is displayed on the inner scale ring. ("Frame inside")
\end{tabular} \\
\hline "Scale type" & \begin{tabular}{l}
Type of scale \\
- "Lines" \\
- "Dots" \\
- "Squares"
\end{tabular} \\
\hline "Scale start" & \begin{tabular}{l}
Least value of the scale and the lower limit of the value range for the element Example: 0 \\
*: The "Variable" property is displayed in the line below this.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the scale start \\
Example: PLC_PRG.iScaleStart \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iScaleStart : INT := 0;
END_VAR

```
\end{tabular} \\
\hline "Scale end" & \begin{tabular}{l}
Greatest value of the scale and the upper limit of the value range for the element \\
Example: 100 \\
p: The "Variable" property is shown below this.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the scale end \\
Example: PLC_PRG.iScaleEnd \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iScaleEnd : INT := 120;
END_VAR

```
\end{tabular} \\
\hline "Main scale" & \begin{tabular}{l}
Distance between two values on the main scale \\
Example: 10 \\
p: The "Variable" property is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type) Contains the distance between two values on the main scale \\
Example: PLC_PRG.iMainScale \\
Declaration: \\
PROGRAM PLC_PRG \\
VAR \\
iMainScale : INT := 20;
\end{tabular} \\
\hline "Sub scale" & \begin{tabular}{l}
Distance between two values on the fine scale You can hide the fine scale by setting the value to 0 . \\
Example: 2 \\
*: The "Variable" property is shown below this.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type) Contains the distance between two values on the fine scale \\
Example: PLC_PRG.iSubScale \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iSubScale : INT := 5;
END VAR

```
\end{tabular} \\
\hline "Scale line width" & \begin{tabular}{l}
Specified in pixels \\
Example: 3
\end{tabular} \\
\hline "Scale color" & \begin{tabular}{l}
Color of scale lines \\
The "Color" dialog opens. \\
- - A list box with style colors opens.
\end{tabular} \\
\hline "Scale in 3D" & Scale lines are displayed with soft 3D shadowing. Note: This property is not displayed in "FlatStyle". \\
\hline "Show scale" & - The scale is displayed. \\
\hline "Frame inside" & : A frame is drawn at the inner end of the scale. \\
\hline "Frame outside" & : A frame is drawn at the outer end of the scale. \\
\hline
\end{tabular}

Element prop-
erty 'Label'
\begin{tabular}{|l|l|}
\hline "Label" & \begin{tabular}{l} 
Selection list \\
- "Outside": Scale values are placed outside of the scale. \\
- "Inside": Scale values are placed inside of the scale.
\end{tabular} \\
\hline "Unit" & \begin{tabular}{l} 
Text that is displayed in the element. \\
Example: Units displayed in m/s.
\end{tabular} \\
\hline "Font" & \begin{tabular}{l} 
Font for labels (example: scale numbering). \\
Selection from the drop-down list or by clicking the "" button. ...
\end{tabular} \\
\hline "Scale format (C Syntax)" & \begin{tabular}{l} 
Values scaled in "printf" syntax \\
Examples: \%d, \%5.2f
\end{tabular} \\
\hline "Max. text width of labels" & \begin{tabular}{l} 
(optional) Value that redefines the maximum width of the scale label. The correct \\
value is normally set automatically. \\
Note: Change this value only if the automatic adjustment does not yield the \\
expected result.
\end{tabular} \\
\hline "Text height of labels" & \begin{tabular}{l} 
(optional) Value that redefines the maximum height of the scale label. The cor- \\
rect value is normally set automatically. \\
Note: Change this value only if the automatic adjustment does not yield the \\
expected result.
\end{tabular} \\
\hline "Font color" & Selection from the drop-down list or by clicking the ■... button. \\
\hline
\end{tabular}

\section*{Element property 'Positioning'}
\begin{tabular}{|l|l|}
\hline "Usage of" & \begin{tabular}{l} 
- "Preset style values": Values from the current style \\
- "User-defined settings": The subnode "Positioning" appears.
\end{tabular} \\
\hline \begin{tabular}{l} 
"Positioning" \\
Requirement: "User-defined settings" is selected as "Usage of". \\
The displayed positioning settings depend on the type of needle instrument and Potentiometer, and partially on \\
whether a custom background image is selected. The following settings are used for achieving the exact position \\
relative to the background image.
\end{tabular} \\
\hline "Needle movement" & Length of the needle (in pixels) \\
\hline "Scale movement" & \begin{tabular}{l} 
Distance from the tick marks to the center (in pixels) \\
Requirement: A customer image is selected as "Background".
\end{tabular} \\
\hline "Scale length" & \begin{tabular}{l} 
Length of the tick marks (in pixels) \\
Requirement: A customer image is selected as "Background".
\end{tabular} \\
\hline "Label offset": & Distance from the labels to the tick marks (in pixels) \\
\hline "Unit offset": & Distance of the unit text "Label \(\mathbf{\rightarrow}\) Unit" from the upper scale edge (in pixels) \\
\hline "Origin offset" & \begin{tabular}{l} 
Offset of the element (in pixels) \\
Requirement: For the elements "Meter 180"" and "Meter 90"", this property is \\
displayed only if a custom image is selected as "Background".
\end{tabular} \\
\hline
\end{tabular}

\section*{Element prop-}
erty 'Colors'
\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|l|}{"Color areas"} \\
\hline "Durable color areas" & \begin{tabular}{l}
: All color areas are visible, regardless of the current value. \\
Only the color area is visible that includes the current value.
\end{tabular} \\
\hline "Use colors for scale" & \: Colors in the color area are used only for the scale and frame. \\
\hline "Color areas" & \\
\hline "Create new" & A new color area is added to the "Elements" view. \\
\hline "Delete" & The color area is removed from the list and the list is refreshed. \\
\hline "Begin of area" & \begin{tabular}{l}
Start value of the color area \\
Example: 20 \\
p: The property "Variable" is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the start value. \\
Example: PLC_PRG.iColorAreaStarto \\
Declaration:
```

PROGRAM PLC_PRG
VAR
iColorAreaStart0 : INT := 80;
END_VAR

```
\end{tabular} \\
\hline "End of area" & \begin{tabular}{l}
End value of the color area \\
Example: 120 \\
: The property "Variable" is shown below.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Variable" & \begin{tabular}{l} 
Variable (integer data type). Contains the end value. \\
Example: iColorAreaEndo \\
Declaration: \\
\\
\\
\\
\\
\hline PROGRAM PLC_PRG \\
VAR iColorAreaEndo : INT \(:=100 ;\) \\
END_VAR \(\quad\) Color that is used for displaying the area. \\
\hline
\end{tabular} \\
\hline
\end{tabular}

Element prop- The variables control the element behavior dynamically. erty 'State varia-
bles'
\begin{tabular}{|l|l|}
\hline "Invisible" & \begin{tabular}{l} 
Variable (BOOL). Toggles the visibility of the element. \\
\\
\\
\\
TRUE: The element is not visible at runtime. \\
Example: bIsVisible with VAR bIsVisible : BOOL : \(=\) FALSE; \\
END_VAR
\end{tabular} \\
\hline "Deactivate inputs" & \begin{tabular}{l} 
Variable (BOOL). Toggles the operability of the element. \\
TRUE: User inputs do not have any effect in runtime more. The element is shown \\
as deactivated.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.
\begin{tabular}{|l|l|}
\hline "Movement" \\
\hline "X" & \begin{tabular}{l} 
Variable (numeric data type). Defines the X position (in pixels). \\
Example: PLC_PRG.iPos_X. \\
Increasing this value in runtime mode moves the element to the right.
\end{tabular} \\
\hline "Y" & \begin{tabular}{l} 
Variable (numeric data type). Defines the Y position (in pixels). \\
Example: PLC_PRG.iPos_Y. \\
Increasing this value in runtime mode moves the element downwards.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle1. \\
The midpoint of the element rotates at the "Center" \\
point. This rotation point is shown as the \(\oplus\) \\
symbol. \\
In runtime mode, the alignment of the element \\
remains the same with respect to the coordinate \\
system of the visualization. Increasing the value \\
rotates the element to the right.
\end{tabular} \\
\hline "Interior rotation" & \begin{tabular}{l} 
Variable (numeric data type). Defines the angle of \\
rotation (in degrees). \\
Example: PLC_PRG. iAngle2.
\end{tabular} \\
\begin{tabular}{l} 
In runtime mode, the element rotates about the \\
point of rotation specified in "Center" according to \\
the value of the variable. In addition, the alignment \\
of the element rotates according to the coordinate \\
system of the visualization. Increasing the value in \\
the code rotates clockwise. \\
The rotation point is shown as the \(\boldsymbol{\oplus}\) symbol.
\end{tabular} \\
\begin{tabular}{l} 
Note: If a static angle of rotation is specified in the \\
"opsition \(\rightarrow\) Angle" property, then the static angle \\
of rotation is added to the variable angle of rotation \\
(offset) when the visualization is executed.
\end{tabular}
\end{tabular}

You can link the variables to a unit conversion.

The " \(X\) ", " "", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

\section*{Element property 'Access rights'}
\begin{tabular}{|l|l|}
\hline "Access rights" & \begin{tabular}{l} 
Opens the "Access rights" dialog. There you can edit the access privileges for \\
the element. \\
Status messages: \\
- "Not set. Full rights.": Access rights for all user groups : "operable" \\
- "Rights are set: Limited rights": Access is restricted for at least one group.
\end{tabular} \\
\(\qquad\)\begin{tabular}{l} 
See also \\
- 』 Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366
\end{tabular}
\end{tabular}

\section*{Visualization Element 'Histogram'}

Symbol:

\section*{Category: "Measurement Controls"}

The element displays the data of a one-dimensional array as a histogram. You can define specific colors for certain value ranges.

See also
- ⓨ Chapter 6.4.5.2.4 "Displaying Array Data in a Histogram" on page 2862

\section*{Element proper- \\ ties}
\begin{tabular}{|l|l|}
\hline "Element name" & Example: GenElemInst_35 \\
\hline "Type of element" & "Histogram" \\
\hline "Data array" & \begin{tabular}{l} 
One-dimensional array with data displayed in this histogram. \\
Example: PLC_PRG.arr1
\end{tabular} \\
\hline
\end{tabular}

\section*{Element property 'Subrange of array'}
\begin{tabular}{|l|l|}
\hline "Use subrange" & : Only part of the array is displayed in the histogram. \\
\hline "Start index" & \begin{tabular}{l} 
First array index with a displayed value. \\
Requirement: "Use subrange" is activated.
\end{tabular} \\
\hline "End index" & \begin{tabular}{l} 
Last array index with a displayed value. \\
Requirement: "Use subrange" is activated.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Display type" & \begin{tabular}{l} 
- "Bars": Data is displayed as bars. \\
- "Lines": Data is displayed as lines. \\
- "Curve": Interpolation of data into a curve.
\end{tabular} \\
\hline "Line width" & \begin{tabular}{l} 
Specified in pixels \\
Requirement: "Curve" is selected as the "Display type".
\end{tabular} \\
\hline "Show horizontal lines" & \begin{tabular}{l} 
: : Horizontal lines are drawn on the main scale. \\
Note: Not all visualization styles have this property. This element property is not \\
available for visualization styles that have striped backgrounds (example: "Flat \\
style").
\end{tabular} \\
\hline "Relative bar width" & \begin{tabular}{l} 
Integer value between 1 and 100 \\
- 1: The bars are drawn as lines. \\
- 100: The entire width of the histogram is filled with the bars.
\end{tabular} \\
\hline
\end{tabular}

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal \(x\)-axis runs to the right. The positive vertical \(y\)-axis runs downwards.
\begin{tabular}{|l|l|}
\hline " \(X "\) & \begin{tabular}{l} 
X coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline " \(\mathrm{Y} "\) & \begin{tabular}{l} 
Y coordinate of the upper left corner of the element \\
Specified in pixels. \\
Example: 10.
\end{tabular} \\
\hline "Width" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 150
\end{tabular} \\
\hline "Height" & \begin{tabular}{l} 
Specified in pixels. \\
Example: 30
\end{tabular} \\
\hline
\end{tabular}

You can also change the values by dragging the box symbols (■-) to other positions in the editor.

See also
- Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center'
\begin{tabular}{|l|l|}
\hline " \(X\) " & X-coordinate of the point of rotation \\
\hline " \(Y\) " & \(Y\)-coordinate of the point of rotation \\
\hline
\end{tabular}

You can also change the values by dragging the symbols \((\boldsymbol{\oplus})\) to other positions in the editor.

Element property 'Scale'
\begin{tabular}{|l|l|}
\hline "Scale start" & \begin{tabular}{l} 
Least value of the scale and the lower limit of the value range for the element. \\
Example: 0 \\
: : The property "Variable" is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (integer data type). Contains the scale start. \\
Example: PLC_PRG. iScaleStart
\end{tabular} \\
\hline "Scale end" & \begin{tabular}{l} 
Greatest value of the scale and the upper limit of the value range for the ele- \\
ment. \\
Example: 100 \\
: The property "Variable" is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l} 
Variable (integer data type). Contains the scale end. \\
Example: PLC_PRG.iScaleEnd
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline "Main scale" & \begin{tabular}{l}
Distance between 2 values on the rough scale. \\
Example: 10 \\
»: The property "Variable" is shown below.
\end{tabular} \\
\hline "Variable" & \begin{tabular}{l}
Variable (integer data type). Contains the distance. \\
Example: PLC_PRG.iMainScale
\end{tabular} \\
\hline "Subscale" & \begin{tabular}{l}
Distance between 2 values on the fine scale. \\
You can hide the fine scale by setting the value to 0 . \\
Example: 2 \\
p: The property "Variable" is shown below.
\end{tabular} \\
\hline "Variable" & Variable (integer data type). Contains the spacing. Example: PLC_PRG.iSubScale \\
\hline "Scale color" & \begin{tabular}{l}
Color of scale lines \\
- ...: \\
The "Color" dialog box opens. \\
- - A drop-down list with color names opens.
\end{tabular} \\
\hline "Base line" & \begin{tabular}{l}
Value of the main scale where the horizontal base line of the Histogram is located. \\
The drawing of the bar starts at the base line.
\end{tabular} \\
\hline
\end{tabular}
```

Example A valid declaration is required for the variables used as an example in the table above.

```
```

PROGRAM PLC_PRG

```
PROGRAM PLC_PRG
VAR
    iScaleStart : INT := 0;
        iScaleEnd : INT := 120;
        iMainScale : INT := 20;
        iSubScale : INT := 5;
    END_VAR
```


## Element prop- <br> erty 'Label'

| "Unit" | Text that is displayed in the element. <br> Example: Units displayed in $\mathrm{m} / \mathrm{s}$. |
| :--- | :--- |
| "Font" | Font for labels (example: scale numbering). <br> Selection from the drop-down list or by clicking the "" button. ... |
| "Scale format (C Syntax)" | Values scaled in "printf" syntax <br> Examples: od, \%5.2f |
| "Max. text width of labels" | Optional value that defines the maximum width of the scale label. <br> Note: Change this value only if the automatic adjustment does not yield the <br> expected result. |
| "Text height of labels" | Optional value that defines the maximum height of the scale label. <br> Note: Change this value only if the automatic adjustment does not yield the <br> expected result. |
| "Font color" | Selection from the drop-down list or by clicking the .... button. |

Element property 'Colors'

| "Graph color" | Color of the bar in normal state. <br> Note: The normal state is in effect when the current value of the array compo- <br> nent does not fulfill the alarm condition. |
| :--- | :--- |
| "Alarm value" | Threshold for the alarm |
| "Alarm condition" | If the current value of the array component fulfills the alarm condition, then the <br> alarm condition is set. <br> - "Less": The current value is less than the "Alarm value" <br> - "More": The current value is greater than the "Alarm value" |
| "Alarm color" | Color of the bar in alarm state. |
| "Use color areas" | "目 The color areas defined in this element are used. |
| "Color areas" | A new color area is added. |
| "Create new" | The color area is removed from the list. |
| "Delete" | The start value on the "Scale" of the Histogram where the color area begins. |
| "Begin of area" | The end value on the "Scale" of the Histogram where the color area ends. |
| "End of area" | Color that is used for displaying the area. |
| "Color" |  |

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

| "Movement" | Variable (numeric data type). Defines the $X$ position (in pixels). <br> Example: PLC_PRG.iPos_X. <br> Increasing this value in runtime mode moves the element to the right. |
| :--- | :--- |
| "Y" | Variable (numeric data type). Defines the Y position (in pixels). <br> Example: PLC_PRG.iPos_Y. <br> Increasing this value in runtime mode moves the element downwards. |
| "Rotation" | Variable (numeric data type). Defines the angle of <br> rotation (in degrees). <br> Example: PLC_PRG. iAngle1. <br> The midpoint of the element rotates at the "Center" <br> point. This rotation point is shown as the $₫$ <br> symbol. <br> In runtime mode, the alignment of the element <br> remains the same with respect to the coordinate <br> system of the visualization. Increasing the value <br> rotates the element to the right. |


| "Interior rotation" | Variable (numeric data type). Defines the angle of <br> rotation (in degrees). <br> Example: PLC_PRG. iAngle2. <br> In runtime mode, the element rotates about the <br> point of rotation specified in "Center" according to <br> the value of the variabbe. In addition, the alignment <br> of the element rotates according to the coordinate <br> system of the visualization. Increasing the value in <br> the code rotates clockwise. <br> The rotation point is shown as the $\oplus$ symbol. |
| :--- | :--- |
| Note: If a static angle of rotation is specified in the <br> "Position $\rightarrow$ Angle" property, then the static angle <br> of rotation is added to the variable angle of rotation <br> (offset) when the visualization is executed. |  |

> You can link the variables to a unit conversion.

The " $X$ ", " $Y$ ", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also
-

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.

| "Animation duration" | Defines the duration (in milliseconds) in which the element runs an animation <br> - Variable (integer value) <br> Example: Menu.tContent with VAR tContent : INT := 500; END_VAR <br> - Integer literal <br> Example: 500 <br> Animatable properties <br> - "Absolute movement", "Movement", " $X$ ", " $Y$ " <br> - "Absolute movement", "Rotation" <br> - "Absolute movement", "Interior rotation" <br> - "Absolute movement", "Exterior rotation" <br> The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth. |
| :---: | :---: |
| "Move to foreground" | Moves the visualization element to the foreground <br> Variable (BOOL) <br> Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR <br> TRUE: At runtime, the visualization element is displayed in the foreground. <br> FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor. |

Element prop- Requirement: User management is set up for the visualization. erty 'Access
rights'

| "Access rights" | Opens the "Access rights" dialog. There you can edit the access privileges for <br> the element. <br> Status messages: |
| :--- | :--- |
|  | - "Not set. Full rights.": Access rights for all user groups : "operable" <br> $\bullet \quad$ "Rights are set: Limited rights": Access is restricted for at least one group. |

See also

- ※y Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366


## Visualization Element 'Image Switcher'

Symbol:


## Category: "Lamps/Switches/Bitmaps"

The element displays one of three referenced images. Mouse actions change the displayed image. The images are defined in the "Image settings" element properties. The effects of mouse clicks are defined in the "Element behavior" property.

## Element proper- <br> ties

| "Element name" | Optional <br> Hint: Assign individual names for elements so that they are found faster in the <br> element list. <br> Example: ImageSwitcher_1 |
| :--- | :--- |
| "Type of element" | "Image Switcher" |

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal $x$-axis runs to the right. The positive vertical $y$-axis runs downwards.

| " $X$ " | X coordinate of the upper left corner of the element <br> Specified in pixels. <br> Example: 10. |
| :--- | :--- |
| "Y" | Y coordinate of the upper left corner of the element <br> Specified in pixels. <br> Example: 10. |
| "Width" | Specified in pixels. <br> Example: 150 |
| "Height" | Specified in pixels. <br> Example: 30 |

You can also change the values by dragging the box symbols (ए़-) to other positions in the editor.

See also

- "y Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

| "Variable" | Variable (BOOL). <br> The value of the variable changes according to user input and it is independent <br> of the "Element behavior" element property. |
| :--- | :--- |

Image settings

| "Image "on"" | Image ID from an image pool. The image can be selected using the input assis- <br> tant. <br> The image is used if the variable of the "Variable" property has the value TRUE. |
| :--- | :--- |
| "Image "off"" | Image ID from an image pool. The image can be selected using the input assis- <br> tant. <br> The image is used if the variable of the "Variable" property has the value FALSE. |
| "Image "clicked""" | Image ID from an image pool. The image is selected using the input assistant. <br> In runtime mode, the visualization displays the referenced image when the ele- <br> ment is clicked (and the mouse button is held down). <br> Requirement: The "Element behavior" is "Image toggler". |


| "Transparency" | 冋: The "Transparent color" is selected. |
| :---: | :---: |
| "Transparent color" | The image pixels that have the transparent color are displayed as transparent. <br> Requirement: "Transparency" is activated. <br> - ...The "Color" dialog box opens. <br> - $\quad$ : A drop-down list with color names opens. |
| "Scaling type" | Defines how an image fits in the element frame. <br> - "Fixed": The original size of the image is retained, regardless of the dimensions of the element. <br> - "Isotropic": The entire image is shown in the element frame, either larger or smaller. As a result, the proportion of height and width are retained. <br> - "Anisotropic": The image resizes automatically to the dimensions of the element frame, filling the entire element frame. As a result, the proportions are not retained. |
| "Horizontal alignment" | Horizontal alignment of the image within the element frame or element <br> - Left <br> - Centered <br> - Right <br> Requirement: "Scaling type" is "lsotropic". |
| "Vertical alignment" | Vertical alignment of the image within the element frame or element <br> - Top <br> - Centered <br> - Bottom <br> Requirement: "Scaling type" is "Isotropic". |


| "Element behavior" | - "Image toggler": Every mouse click switches the image. <br> - "Image tapper"" While a visualization user holds down the mouse button, the <br> image of the "Image on" property is displayed. At the same time, the value <br> TRUE is assigned to the "Variable" property. |
| :--- | :--- |
| "Tap FALSE" | V: While the mouse button is pressed, the image of the "Image" property is <br> displayed and the "Variable" property gets the value FALSE instead of the value <br> TRUE, and back. <br> Requirement: "Image tapper" is selected in the "Element behavior" property. |

Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the $\boldsymbol{\oplus}$ symbol. The point is used as the center for rotating and scaling.

| " $X$ " | X-coordinate of the point of rotation |
| :--- | :--- |
| " $Y$ " | $Y$-coordinate of the point of rotation |

You can also change the values by dragging the symbols ( $\boldsymbol{\oplus}$ ) to other positions in the editor.

## Element property 'Texts'

| "Tooltip" | String display as tooltip for the element <br> Example: Valid access. |
| :--- | :--- |

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

| "Movement" | Variable (numeric data type). Defines the X position (in pixels). <br> Example: PLC_PRG. iPos_X. <br> Increasing this value in runtime mode moves the element to the right. |
| :--- | :--- |
| "Y" | Variable (numeric data type). Defines the Y position (in pixels). <br> Example: PLC_PRG. iPos_Y. <br> Increasing this value in runtime mode moves the element downwards. |
| "Rotation" | Variable (numeric data type). Defines the angle of <br> rotation (in degrees). <br> Example: PLC_PRG. iAngle1. <br> The midpoint of the element rotates at the "Center" <br> point. This rotation point is shown as the $\oplus$ <br> symbol. <br> In runtime mode, the alignment of the element <br> remains the same with respect to the coordinate <br> system of the visualization. Increasing the value <br> rotates the element to the right. |
| "Interior rotation" | Variable (numeric data type). Defines the angle of <br> rotation (in degrees). <br> Example: PLC_PRG. iAngle2. <br> In runtime mode, the element rotates about the <br> point of rotation specified in "Center" according to <br> the value of the variable. In addition, the alignment <br> of the element rotates according to the coordinate <br> system of the visualization. Increasing the value in <br> the code rotates clockwise. <br> The rotation point is shown as the $\oplus$ symbol. <br> Note: If a static angle of rotation is specified in the <br> "Position $\rightarrow$ Angle" property, then the static angle <br> of rotation is added to the variable angle of rotation <br> (offset) when the visualization is executed. |

You can link the variables to a unit conversion.


## See also

Element prop- The variables control the element behavior dynamically. erty 'State variables'

| "Invisible"" | Variable (BOOL). Toggles the visibility of the element. |
| :--- | :--- |
|  | TRUE: The element is not visible at runtime. <br> Example: bIsVisible with VAR bIsVisible : BOOL : <br> END_VAR |
| "Deactivate inputs" | Variable (BOOL). Toggles the operability of the element. <br> TRUE: User inputs do not have any effect in runtime more. The element is shown <br> as deactivated. |

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.

| "Animation duration" | Defines the duration (in milliseconds) in which the element runs an animation <br> -Variable (integer value) <br> Example: Menu. tContent with VAR tContent : INT $:=500 ;$ <br> END_VAR <br> - Integer literal <br> Example: 500 <br> Animatable properties <br> - "Absolute movement", "Movement", "X", " $Y$ " <br> - "Absolute movement", ""otation" <br> - "Absolute movement", "Interior rotation" <br> - "Absolute movement", "Exterior rotation" <br> The animated movement is executed when at least one value of an animatable <br> property has changed. The movement then executed is not jerky, but is smooth <br> within the specified animation duration. The visualization element travels to the <br> specified position while rotating dynamically. The transitions are smooth. |
| :--- | :--- |
| Moves the visualization element to the foreground <br> Variable (BOOL) <br> Example: bIs InForeground with VAR bIsInForeground : BooL $:=$ <br> FALSE; END_VAR |  |
| "Move to foreground" |  |
| TRUE: At runtime, the visualization element is displayed in the foreground. |  |
| FALSE: At runtime, the visualization element is displayed in the layer where it |  |
| was inserted in the visualization editor. |  |

Element prop- Requirement: User management is set up for the visualization. erty 'Access
rights'

| "Access rights" | Opens the "Access rights" dialog. There you can edit the access privileges for <br> the element. <br> Status messages: |
| :--- | :--- |
|  | - "Not set. Full rights.": Access rights for all user groups : "operable" <br> $\bullet$ <br> "Rights are set: Limited rights": Access is restricted for at least one group. |

See also

- ※y Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

See also

- $\Leftrightarrow_{2}$ Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872


## Visualization Element 'Lamp'

Symbol:


## Category: "Lamps/Switches/Bitmaps"

The element shows the value of a variable, and the element is displayed as illuminated or not.

## Element proper-

ties

| "Element name" | Optional <br> Hint: Assign individual names for elements so that they are found faster in the <br> element list. <br> Example: Lamp_green |
| :--- | :--- |
| "Type of element" | "Lamp" |

Element prop- The position defines the location and size of the element in the visualization window. These erty 'Position' are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal $x$-axis runs to the right. The positive vertical $y$-axis runs downwards.

| " $X$ " | X coordinate of the upper left corner of the element <br> Specified in pixels. <br> Example: 10. |
| :--- | :--- |
| "Y" | Y coordinate of the upper left corner of the element <br> Specified in pixels. <br> Example: 10. |
| "Width" | Specified in pixels. <br> Example: 150 |
| "Height" | Specified in pixels. <br> Example: 30 |

You can also change the values by dragging the box symbols (區) to other positions in the editor.

See also

- «y Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

| "Variable" | Variable (BOOL). <br> The variable value is displayed as a lamp that goes on (TRUE) or off (FALSE). |
| :--- | :--- |

Image settings

| "Transparency" | $\checkmark$ : The "Transparent color" property is selected. |
| :---: | :---: |
| "Transparent color" | Pixels in this color are displayed as transparent. <br> Requirement: "Transparency" is activated. <br> - ... The "Color" dialog box opens. <br> - - A drop-down list with style colors opens. |


| "Scaling type" | Reaction of the element when the dimension of the "Frame" element is changed: <br> - "/sotropic": The height and width of the image are resized proportionally to the "Frame". <br> Please note: To retain the alignment of elements also within a scaled "Frame" element, define the "Horizontal alignment" or "Vertical alignment" explicitly with "Centered". <br> - "Anisotropic": The image fills the entire "Frame" regardless of its proportions. |
| :---: | :---: |
| "Horizontal alignment" | Horizontal alignment of the image within the element frame or element <br> - Left <br> - Centered <br> - Right <br> Requirement: "Scaling type" is "lsotropic". |
| "Vertical alignment" | Vertical alignment of the image within the element frame or element <br> - Top <br> - Centered <br> - Bottom <br> Requirement: "Scaling type" is "Isotropic". |

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center' rotation is shown as the $\oplus$ symbol. The point is used as the center for rotating and scaling.

| " $X$ " | X-coordinate of the point of rotation |
| :--- | :--- |
| $Y "$ | Y-coordinate of the point of rotation |

You can also change the values by dragging the symbols (ఉ) to other positions in the editor.

## Element prop-

erty 'Texts'

| "Tooltip" | String display as tooltip for the element <br> Example: Valid access. |
| :--- | :--- |

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

| "Movement" |  |
| :--- | :--- |
| " $X "$ | Variable (numeric data type). Defines the X position (in pixels). <br> Example: PLC_PRG.iPos_X. <br> Increasing this value in runtime mode moves the element to the right. |
| "Y" | Variable (numeric data type). Defines the Y position (in pixels). <br> Example: PLC_PRG.iPos_Y. <br> Increasing this value in runtime mode moves the element downwards. |


| "Rotation" | Variable (numeric data type). Defines the angle of <br> rotation (in degrees). <br> Example: PLC_PRG. iAngle1. <br> The midpoint of the element rotates at the "Center" <br> point. This rotation point is shown as the $\oplus$ <br> symbol. <br> In runtime mode, the alignment of the element <br> remains the same with respect to the coordinate <br> system of the visualization. Increasing the value <br> rotates the element to the right. |
| :--- | :--- |
| "Interior rotation" | Variable (numeric data type). Defines the angle of <br> rotation (in degrees). <br> Example: PLC_PRG. iAngle2. <br> In runtime mode, the element rotates about the <br> point of rotation specified in "Center" according to <br> the value of the variable. In addition, the alignment <br> of the element rotates according to the coordinate <br> system of the visualization. Increasing the value in <br> the code rotates clockwise. <br> The rotation point is shown as the $\oplus$ symbol. |
| Note: If a static angle of rotation is specified in the <br> "Position $\rightarrow$ Angle" property, then the static angle <br> of rotation is added to the variable angle of rotation <br> (offset) when the visualization is executed. |  |

You can link the variables to a unit conversion.

The "X", "Y", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

## See also

Element prop- The variables control the element behavior dynamically.
erty 'State varia-
bles'

| "Invisible" | Variable (BOOL). Toggles the visibility of the element. <br> TRUE: The element is not visible at runtime. |
| :--- | :--- |



The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.

| "Animation duration" | Defines the duration (in milliseconds) in which the element runs an animation <br> -Variable (integer value) <br> Example: Menu. tContent with VAR tContent : INT $:=500 ;$ <br> END_VAR <br> - Integer literal <br> Example: 500 <br> Animatable properties <br> - "Absolute movement", "Movement", "X", " $Y$ " <br> - "Absolute movement", ""otation" <br> - "Absolute movement", "Interior rotation" <br> - "Absolute movement", "Exterior rotation" <br> The animated movement is executed when at least one value of an animatable <br> property has changed. The movement then executed is not jerky, but is smooth <br> within the specified animation duration. The visualization element travels to the <br> specified position while rotating dynamically. The transitions are smooth. |
| :--- | :--- |
| Moves the visualization element to the foreground <br> Variable (BOOL) <br> Example: bIs InForeground with VAR bIsInForeground : BooL $:=$ <br> FALSE; END_VAR |  |
| "Move to foreground" |  |
| TRUE: At runtime, the visualization element is displayed in the foreground. |  |
| FALSE: At runtime, the visualization element is displayed in the layer where it |  |
| was inserted in the visualization editor. |  |

Element prop-
erty 'Back-
ground'

| "Image" | Drop-down list with background colors <br> Depends on the visualization style |
| :--- | :--- |

Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'

| "Access rights" | Opens the "Access rights" dialog. There you can edit the access privileges for <br> the element. <br> Status messages: <br> - "Not set. Full rights.": Access rights for all user groups : "operable" <br>  <br> - "Rights are set: Limited rights": Access is restricted for at least one group. |
| :--- | :--- |

See also

- \& Chapter 6.4.5.21.3.1 "Dialog 'Access Rights"" on page 3366

See also

- 乡y Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

Visualization Element 'Dip Switch', 'Power Switch', 'Push Switch', 'Push Switch LED', 'Rocker Switch'

Symbols:


## Category: "Lamps/Switches/Bitmaps"

The element assigns a value to a Boolean variable. The switch position "on" the value TRUE to the variable, and the switch position "off" assigns the value FALSE. Use the mouse to change the switch position.

## Element proper-

ties

| "Element name" | Example: Operating_Switch <br> Optional |
| :--- | :--- |
| Hint: Assign individual names for elements so that they are found faster in the <br> element list. |  |
| "Type of element" | Depending on the element: "Dip Switch", "Power Switch", "Push Switch", "Push <br> Switch LED", or "Rocker Switch" |

Element prop- The position defines the location and size of the element in the visualization window. These erty 'Position' are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal $x$-axis runs to the right. The positive vertical $y$-axis runs downwards.

| "X" | X coordinate of the upper left corner of the element <br> Specified in pixels. <br> Example: 10. |
| :--- | :--- |
| "Y" | Y coordinate of the upper left corner of the element <br> Specified in pixels. <br> Example: 10. |
| "Width" | Specified in pixels. <br> Example: 150 |
| "Height" | Specified in pixels. <br> Example: 30 |



You can also change the values by dragging the box symbols ( ( $^{-}$) to other positions in the editor.

## See also

- Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

| "Variable" | Variable (BOOL) <br> The value of the variables TRUE and FALSE indicates the switch position on/off. |
| :--- | :--- |

## Image settings

| "Transparency" | : The "Transparent color" property is selected. |
| :--- | :--- |
| "Transparent color" | Pixels in this color are displayed as transparent. <br> Requirement: "Transparency" is activated. <br> - <br>  <br> -...The "Color" dialog box opens. |
| "Scaling type" A drop-down list with style colors opens. |  |


| "Element behavior" | - "Image toggler": Every mouse click changes the switch and the "Variable" <br> - <br> value. <br> "Image tapper": The switch is "on" and the "Variable" value is TRUE while the <br> mouse button is pressed. |
| :--- | :--- |
| "Tap FALSE" | V: The value TRUE is assigned to the "Variable" property instead of the value <br> FALSE, and back. <br> Requirement: "Image tapper" is selected in the "Element behavior" property. |

Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the $\boldsymbol{\oplus}$ symbol. The point is used as the center for rotating and scaling.

| " $X$ " | X-coordinate of the point of rotation |
| :--- | :--- |
| " $Y$ " | Y-coordinate of the point of rotation |

You can also change the values by dragging the symbols ( $\boldsymbol{\oplus})$ to other positions in the editor.

## Element prop- <br> erty 'Texts'

| "Tooltip" | String display as tooltip for the element <br> Example: Valid access. |
| :--- | :--- |

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

| "Movement" | Variable (numeric data type). Defines the X position (in pixels). <br> Example: PLC_PRG. iPos_X. <br> Increasing this value in runtime mode moves the element to the right. |
| :--- | :--- |
| "Y" | Variable (numeric data type). Defines the Y position (in pixels). <br> Example: PLC_PRG. iPos_Y. <br> Increasing this value in runtime mode moves the element downwards. |
| "Rotation" | Variable (numeric data type). Defines the angle of <br> rotation (in degrees). <br> Example: PLC_PRG. iAngle1. <br> The midpoint of the element rotates at the "Center" <br> point. This rotation point is shown as the $\oplus$ <br> symbol. <br> In runtime mode, the alignment of the element <br> remains the same with respect to the coordinate <br> system of the visualization. Increasing the value <br> rotates the element to the right. |
| "Interior rotation" | Variable (numeric data type). Defines the angle of <br> rotation (in degrees). <br> Example: PLC_PRG. iAngle2. <br> In runtime mode, the element rotates about the <br> point of rotation specified in "Center" according to <br> the value of the variable. In addition, the alignment <br> of the element rotates according to the coordinate <br> system of the visualization. Increasing the value in <br> the code rotates clockwise. <br> The rotation point is shown as the $\oplus$ symbol. <br> Note: If a static angle of rotation is specified in the <br> "Position $\rightarrow$ Angle" property, then the static angle <br> of rotation is added to the variable angle of rotation <br> (offset) when the visualization is executed. |

You can link the variables to a unit conversion.

The " $X$ ", " $Y$ ", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

## See also

Element prop- The variables control the element behavior dynamically. erty 'State variables'

| "Invisible" | Variable (BOOL). Toggles the visibility of the element. <br>  <br>  <br>  <br> TRUE: The element is not visible at runtime. <br> Example: bIsVisible with VAR bIsVisible : BOOL : $=$ EALSE; <br> END_VAR |
| :--- | :--- |
| "Deactivate inputs" | Variable (BOOL). Toggles the operability of the element. <br> TRUE: User inputs do not have any effect in runtime more. The element is shown <br> as deactivated. |

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.

| "Animation duration" | Defines the duration (in milliseconds) in which the element runs an animation <br> - Variable (integer value) <br> Example: Menu.tContent with VAR tContent : INT := 500; END_VAR <br> - Integer literal <br> Example: 500 <br> Animatable properties <br> - "Absolute movement", "Movement", " $X$ ", " $Y$ " <br> - "Absolute movement", "Rotation" <br> - "Absolute movement", "Interior rotation" <br> - "Absolute movement", "Exterior rotation" <br> The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth. |
| :---: | :---: |
| "Move to foreground" | Moves the visualization element to the foreground <br> Variable (BOOL) <br> Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR <br> TRUE: At runtime, the visualization element is displayed in the foreground. <br> FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor. |

## Element prop-

erty 'Back-
ground'

| "Image" | Drop-down list with background colors <br> Depends on the visualization style |
| :--- | :--- |

Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'

| "Access rights" | Opens the "Access rights" dialog. There you can edit the access privileges for <br> the element. <br> Status messages: <br> - "Not set. Full rights.": Access rights for all user groups : "operable" <br>  <br> - "Rights are set: Limited rights": Access is restricted for at least one group. |
| :--- | :--- |

See also

- ② Chapter 6.4.5.21.3.1 "Dialog 'Access Rights"" on page 3366

See also

- ${ }^{4}$ Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872


## Visualization Element 'Rotary Switch'

Symbol:

## Category: "Lamps/Switches/Bitmaps"

The element assigns a value to a Boolean variable. The switch position "on" the value TRUE to the variable, and the switch position "off" assigns the value FALSE. Use the mouse to change the switch position.

## Element proper-

ties

| "Element name" | Example: Operating_Switch |
| :--- | :--- |
|  | Optional <br> Hint: Assign individual names for elements so that they are found faster in the <br> element list. |
| "Type of element" | "Rotary Switch" |

Element prop- The position defines the location and size of the element in the visualization window. These erty 'Position' are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal $x$-axis runs to the right. The positive vertical $y$-axis runs downwards.

| " $X$ " | X coordinate of the upper left corner of the element <br> Specified in pixels. <br> Example: 10. |
| :--- | :--- |
| "Y" | Y coordinate of the upper left corner of the element <br> Specified in pixels. <br> Example: 10. |
| "Width" | Specified in pixels. <br> Example: 150 |
| "Height" | Specified in pixels. <br> Example: 30 |

You can also change the values by dragging the box symbols (뭉) to other positions in the editor.

See also

- ${ }^{\wedge} \boldsymbol{y}$ Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

| "Variable" | Variable (BOOL). <br> The value of the variables TRUE and FALSE indicates the switch position on/off. |
| :--- | :--- |

## Image settings

| "Transparency" | V: The "Transparent color" property is selected. |
| :---: | :---: |
| "Transparent color" | Pixels in this color are displayed as transparent. <br> Requirement: "Transparency" is activated. <br> - ...The "Color" dialog box opens. <br> - $\quad$ : A drop-down list with style colors opens. |
| "Scaling type" | Reaction of the element when the dimension of the "Frame" element is changed: <br> - "/sotropic": The height and width of the image are resized proportionally to the "Frame". <br> Please note: To retain the alignment of elements also within a scaled "Frame" element, define the "Horizontal alignment" or "Vertical alignment" explicitly with "Centered". <br> - "Anisotropic": The image fills the entire "Frame" regardless of its proportions. |
| "Horizontal alignment" | Horizontal alignment of the image within the element frame or element <br> - Left <br> - Centered <br> - Right <br> Requirement: "Scaling type" is "Isotropic". |
| "Vertical alignment" | Vertical alignment of the image within the element frame or element <br> - Top <br> - Centered <br> - Bottom <br> Requirement: "Scaling type" is "Isotropic". |


| "Element behavior" | - "Image toggler": Every mouse click changes the switch and the "Variable" <br> - value. <br> "Image tapper": The switch is "on" and the "Variable" value is TRUE while the <br> mouse button is pressed. |
| :--- | :--- |
| "Orientation" | - "At top": The rotary switch turns from the top right to the top left. <br> - "At side": The rotary switch turns from the top right to the bottom right. |
| "Color change" | ص: The element changes in color when "Variable" is TRUE. |

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center' rotation is shown as the $\boldsymbol{\oplus}$ symbol. The point is used as the center for rotating and scaling.

| " $X$ " | X-coordinate of the point of rotation |
| :--- | :--- |
| " $Y$ " | Y-coordinate of the point of rotation |



## Element property 'Texts'

| "Tooltip" | String display as tooltip for the element <br> Example: Valid access. |
| :--- | :--- |

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

| "Movement" | Variable (numeric data type). Defines the X position (in pixels). <br> Example: PLC_PRG. iPos_X. <br> Increasing this value in runtime mode moves the element to the right. |
| :--- | :--- |
| "Y" | Variable (numeric data type). Defines the Y position (in pixels). <br> Example: PLC_PRG. iPos_Y. <br> Increasing this value in runtime mode moves the element downwards. |
| "Rotation" | Variable (numeric data type). Defines the angle of <br> rotation (in degrees). <br> Example: PLC_PRG. iAngle1. <br> The midpoint of the element rotates at the "Center" <br> point. This rotation point is shown as the $\oplus$ <br> symbol. <br> In runtime mode, the alignment of the element <br> remains the same with respect to the coordinate <br> system of the visualization. Increasing the value <br> rotates the element to the right. |
| "Interior rotation" | Variable (numeric data type). Defines the angle of <br> rotation (in degrees). <br> Example: PLC_PRG. iAngle2. <br> In runtime mode, the element rotates about the <br> point of rotation specified in "Center" according to <br> the value of the variable. In addition, the alignment <br> of the element rotates according to the coordinate <br> system of the visualization. Increasing the value in <br> the code rotates clockwise. <br> The rotation point is shown as the $\oplus$ symbol. <br> Note: If a static angle of rotation is specified in the <br> "Position $\rightarrow$ Angle" property, then the static angle <br> of rotation is added to the variable angle of rotation <br> (offset) when the visualization is executed. |

You can link the variables to a unit conversion.


## See also

Element prop- The variables control the element behavior dynamically. erty 'State variables'

| "Invisible" | Variable (BOOL). Toggles the visibility of the element. |
| :--- | :--- |
|  | TRUE: The element is not visible at runtime. <br> Example: bIsVisible with VAR bIsVisible : BOOL : <br> END_VAR |
| "Deactivate inputs" | Variable (BOOL). Toggles the operability of the element. <br> TRUE: User inputs do not have any effect in runtime more. The element is shown <br> as deactivated. |

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.

| "Animation duration" | Defines the duration (in milliseconds) in which the element runs an animation <br> - Variable (integer value) <br> Example: Menu.tContent with VAR tContent : INT := 500; END_VAR <br> - Integer literal <br> Example: 500 <br> Animatable properties <br> - "Absolute movement", "Movement", "X", " $Y$ " <br> - "Absolute movement", "Rotation" <br> - "Absolute movement", "Interior rotation" <br> - "Absolute movement", "Exterior rotation" <br> The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth. |
| :---: | :---: |
| "Move to foreground" | Moves the visualization element to the foreground <br> Variable (BOOL) <br> Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR <br> TRUE: At runtime, the visualization element is displayed in the foreground. <br> FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor. |

Element prop-
erty 'Back-
ground'

| "Image" | Drop-down list with background colors <br> Depends on the visualization style |
| :--- | :--- |

Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'

| "Access rights" | Opens the "Access rights" dialog. There you can edit the access privileges for <br> the element. <br> Status messages: |
| :--- | :--- |
|  | $\bullet$ "Not set. Full rights.": Access rights for all user groups : "operable" <br> $\bullet$ <br> "Rights are set: Limited rights": Access is restricted for at least one group. |

See also

- ② Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

See also

- ⓨ Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872


## Visualization Element 'Trace'

Symbol:


## Category: "Special Controls"

The element displays the graphical curve of variable values. In addition, variables can be configured to control the view.

See also

- ${ }^{4}$ Chapter 6.4.5.12 "Displaying data curve with trace" on page 2925
- ③ "Dialog box 'Trace Configuration"' on page 3355


## Element proper-

ties

| "Element name" | Example: Velocity |
| :---: | :---: |
| "Data Source" | Location where the trace data is buffered. <br> - "<local application>": The trace record is listed below the local application. The visualization that contains the trace is located below this application. When the application is downloaded, the trace configuration is downloaded to the local device. During execution, the data is stored locally in the trace buffer. <br> - "<data source name>": Data source that identifies the remote device where the trace record is created. <br> When the local application is downloaded with the visualization, the trace configuration is downloaded to the remote device. During execution, the trace buffer is filled, and the trace data is transferred and then displayed in the local visualization as HMI . <br> Example: DataSoure_PLC_A <br> Note: The trace buffer is filled only if the remote application is being executed. The data recording is started when the local visualization is started. |
| "Application" | Application where data was recorded. <br> Lists all applications that are present below the data source. <br> Requirement: A remote data source (not "<local application>") is referenced in the "Data source" property. |
| "Type of element" | "Trace" |
| "Trace" | <name of trace configuration>": Opens the "Trace Configuration" dialog where you can modify the trace configuration. |

See also

- (3) "Dialog box 'Trace Configuration"' on page 3355
- Data Source Manager

Element property 'Position'

The position defines the location and size of the element in the visualization window. This is based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal $x$-axis runs to the right. The positive vertical $y$-axis runs downwards.

| " $X$ " | The x-coordinate of the upper left corner of the element Specified in pixels <br> Example: 10 |
| :---: | :---: |
| "Y" | The y-coordinate of the upper left corner of the element Specified in pixels <br> Example: 10 |
| "Width" | Specified in pixels <br> Example: 150 |
| "Height" | Specified in pixels <br> Example: 30 |
|  | Tip: You can change the values in " $X$ ", " $Y$ ", "Width", and "Height" by dragging the corresponding symbols |
| "Angle" | Static angle of rotation (in degrees) <br> Example: 35 <br> The element is displayed rotated in the editor. The point of rotation is the center of the element. A positive value rotates clockwise. <br> Tip: You can change the value in the editor by focusing the element to the handle. When the cursor is displayed as a rotating arrow $\mathcal{A}$, you can rotate the element about its center as a handle. <br> (1): Handle <br> Note: If a dynamic angle of rotation is also configured in the property "Absolute movement $\rightarrow$ Internal rotation", then the static and dynamic angles of rotation are added in runtime mode. The static angle of rotation acts as an offset. |

See also

- Hy Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

| "Show cursor" | V: A cursor (vertical line) is displayed at the mouse position. The trigger and <br> variable values where the cursor points are displayed as a tooltip. |
| :--- | :--- |
| "Overwrite existing trace on | V: If a trace with the same name is on the PLC, then it is overwritten at down- <br> PLC" |
| "Number format" | Number format of values in the tooltip in printf syntax (example: \%d, \%5.2f). |

Element prop- $\quad$ The control variables are assigned automatically when you click "Insert elements for controlling erty 'Control Trace". variables'

| "Reset Trigger" | Variable (BOOL). <br> Standard control variable: bResetTrigger <br> TRUE: Resets the triggering. After the action is executed, the variable is set automatically to FALSE. |
| :---: | :---: |
| "Start Trace" | Variable (BOOL). <br> Standard control variable: bStart <br> TRUE: Starts the Trace. After the action is executed, the variable is set automatically to FALSE. |
| "Stop Trace" | Variable (BOOL). <br> Standard control variable: bStop <br> TRUE: Stops the Trace. After the action is executed, the variable is set automatically to FALSE. |
| "Save Trace to a file" |  |
| "Save Trace" | Variable (BOOL). <br> Standard control variable: bStore <br> TRUE: Saves the current trace configuration and the data that is stored in the development system to a file. When the action is ended, the variable is set automatically to FALSE. |
| "File name" | Variable (STRING) that contains the file name of the file to be saved. Standard control variable: sStoreFilename |
| "Load trace from file" |  |
| "Load Trace" | Variable (BOOL). <br> Standard control variable: bRestore <br> TRUE: Reads the file specified below and loads its contents into the trace editor. The file contains a trace configuration and possibly also trace data. To do this, the stored trace configuration must match the application where the trace configuration is located. When the action is ended, the variable is set automatically to FALSE. <br> Note: A trace configuration can be loaded from a file only under special circumstances. The file must have been created with exactly the same (running) application with which it will then be loaded. The consequence of changing the running application (for example by downloading again) is that a file which was previously created from the application cannot no longer be read into the application. Even external manual changes to the file can cause this. You should edit only those configuration settings that have an effect on displaying the variables. If you change variable definitions directly in the file (for example by replacing variable x with v y ), then the file cannot be loaded. |
| "File name" | Variable (STRING) that contains the file name of the file to be read. Standard variable: sRestoreFilename |

See also

- ${ }^{*}$ Chapter 6.4.5.21.2.15 "Command 'Insert Elements for Controlling Trace"" on page 3358

Element prop-
erty 'Center'
The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the $\boldsymbol{\oplus}$ symbol. The point is used as the center for rotating and scaling.

| " $X$ " | X-coordinate of the point of rotation |
| :--- | :--- |
| $Y "$ | $Y$-coordinate of the point of rotation |

Element property 'Absolute movement'

You can also change the values by dragging the symbols ( $\boldsymbol{\oplus}$ ) to other positions in the editor.

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

| "Movement" |  |  |
| :---: | :---: | :---: |
| "X" | Variable (numeric data type). Defines the X position (in pixels). <br> Example: PLC_PRG.iPos_X. <br> Increasing this value in runtime mode moves the element to the right. |  |
| "Y" | Variable (numeric data type). Defines the Y position (in pixels). <br> Example: PLC_PRG.iPos_Y. <br> Increasing this value in runtime mode moves the element downwards. |  |
| "Rotation" | Variable (numeric data type). Defines the angle of rotation (in degrees). <br> Example: PLC_PRG.iAngle1. <br> The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol. <br> In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right. |  |
| "Interior rotation" | Variable (numeric data type). Defines the angle of rotation (in degrees). <br> Example: PLC_PRG.iAngle2. <br> In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. <br> The rotation point is shown as the $\boldsymbol{\oplus}$ symbol. <br> Note: If a static angle of rotation is specified in the "Position $\rightarrow$ Angle" property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed. |  |

## You can link the variables to a unit conversion.

The " $X$ ", " $Y$ ", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also
-

Element prop- The variables control the element behavior dynamically. erty 'State variables'

| "Invisible" | Variable (BOOL). Toggles the visibility of the element. <br> TRUE: The element is not visible at runtime. |
| :--- | :--- |

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.

| "Animation duration" | Defines the duration (in milliseconds) in which the element runs an animation <br> -Variable (integer value) <br> Example: Menu. tContent with VAR tContent : INT $:=500 ;$ <br> END_VAR <br> - Integer literal <br> Example: 500 <br> Animatable properties <br> - "Absolute movement", "Movement", "X", " $Y$ " <br> - "Absolute movement", ""otation" <br> - "Absolute movement", "Interior rotation" <br> - "Absolute movement", "Exterior rotation" <br> The animated movement is executed when at least one value of an animatable <br> property has changed. The movement then executed is not jerky, but is smooth <br> within the specified animation duration. The visualization element travels to the <br> specified position while rotating dynamically. The transitions are smooth. |
| :--- | :--- |
| Moves the visualization element to the foreground <br> Variable (BOOL) <br> Example: bIs InForeground with VAR bIsInForeground : BooL $:=$ <br> FALSE; END_VAR |  |
| "Move to foreground" |  |
| TRUE: At runtime, the visualization element is displayed in the foreground. |  |
| FALSE: At runtime, the visualization element is displayed in the layer where it |  |
| was inserted in the visualization editor. |  |

Element prop- Requirement: User management is set up for the visualization. erty 'Access
rights'

| "Access rights" | Opens the "Access rights" dialog. There you can edit the access privileges for <br> the element. <br> Status messages: |
| :--- | :--- |
|  | - "Not set. Full rights.": Access rights for all user groups : "operable" <br> $\bullet$ <br> "Rights are set: Limited rights": Access is restricted for at least one group. |

See also

- $\quad \stackrel{y}{c}$ Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366


## Visualization Element 'Trend'

Symbol:

## Category: "Special Controls"

The element displays the curve of variable values as a trend diagram. The trend diagram is suitable for representing a long-term data curve because the data is read from a trend recording and hence from a database. Moreover, you can run the "Trend" element together with the "Date Range Picker", "Legend", and "Time Range Picker" operating elements so that the user can navigate conveniently in the diagram.

You can programmatically delete the recorded trend curve at runtime. The recording starts again from the time of deletion. See the help page for "Programming a Trend Visualization".

## Element proper-

ties

| "Element name" | Example: Velocity |
| :---: | :---: |
| "Data source" | Data source for the connection via the device and the application to the "Trend Recording" object where the trend data that you want to show was saved. <br> If the "Trend Recording" object is on the local device, then it is sufficient when you specify the respective application. If the trend recording is on a remote device, then you need to specify the data source connection to this device. <br> - 0" <local application>" <br> The "Trend Recording" object is located on the local device in the local application. <br> - <device name> . <application name> <br> Example: Device_A.App_A <br> The "Trend Recording" object is located on the local device Device_A below the application App_A. <br> - $\mathrm{F}^{(\mathrm{m}}$ <data source name> <br> Example: DataSource_B <br> The "Trend Recording" object is located on a remote device that is connected via the data source DataSource_B. Below the (now visible) "Application" property, the remote application is displayed as configured in the data source. <br> Example: App_B <br> Note: If the data source is accessed symbolically by means of a symbol file (CODESYS symbolic), then the required symbol file and the corresponding project have to be saved in the same folder. |
| "Type of element" | "Trend" |
| "Trend recording" | : Trend recording whose data is displayed as a diagram <br> The trend recording is located on the device specified in the "Data source" property. |
| "Display Mode" | L: Opens the "Display Settings" dialog. |

See also

- Chapter 6.4.5.13 "Displaying data curve with trend" on page 2928
- $\triangleq$ Chapter 6.4.5.21.2.16 "Command 'Configure Display Settings of Trend"' on page 3359
- Object 'Data Source'

Element property 'Position'

The position defines the location and size of the element in the visualization window. This is based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal $x$-axis runs to the right. The positive vertical $y$-axis runs downwards.

| "X" | The x-coordinate of the upper left corner of the element Specified in pixels <br> Example: 10 |
| :---: | :---: |
| "Y" | The y-coordinate of the upper left corner of the element Specified in pixels <br> Example: 10 |
| "Width" | Specified in pixels <br> Example: 150 |
| "Height" | Specified in pixels <br> Example: 30 |
|  | Tip: You can change the values in " $X$ ", " $Y$ ", "Width", and "Height" by dragging the corresponding symbols |
| "Angle" | Static angle of rotation (in degrees) <br> Example: 35 <br> The element is displayed rotated in the editor. The point of rotation is the center of the element. A positive value rotates clockwise. <br> Tip: You can change the value in the editor by focusing the element to the handle. When the cursor is displayed as a rotating arrow $\boldsymbol{\sim}$, you can rotate the element about its center as a handle. <br> (1): Handle <br> Note: If a dynamic angle of rotation is also configured in the property "Absolute movement $\rightarrow$ Internal rotation", then the static and dynamic angles of rotation are added in runtime mode. The static angle of rotation acts as an offset. |

See also

- ⓨ Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

| "Show cursor" | A cursor (black triangle with vertical line) is shown in the trend diagram. Behavior at runtime: As soon as the graph is drawn, the user can move the cursor along the time axis in order to mark a specific time. Then the variable value belonging to the cursor position is displayed in the legend above the graph. |
| :---: | :---: |
|  |  |
|  |  |
| "Show tooltip" | Requirement: "Show cursor" is activated. A tooltip opens at the cursor. <br> Behavior at runtime: The variable value belonging to the cursor position is displayed as a tooltip. |
| "Show frame" | - The trend diagram is drawn with a frame. |
| "Number format" | Format specification in printf syntax, which determines how the values are displayed in the tooltip and in the legend <br> Example: \%d (integer variable) or $\% 5.2 \mathrm{f}$ (floating-point number) |

## Element property 'Tick mark labels'

The time stored in the trend recording are in the UTC time zone. If the time is displayed in the trend of the visualization element, then the time stamps are converted to the local time zone of the operating system of the PLC.

Change the time zone in the operating system if the times in the trend diagram are not in the zone that you need.

| "Time stamps" | X-value of the trend diagram <br> - "Absolute time stamps" <br> The absolute time with date and time is displayed at each tick mark on the time axis. <br> Example: 03/18/2016 12h30m50s <br> - "Relative time stamps" <br> The time period from the start of the recording (=0) is displayed at each tick mark. <br> Example: 5m30s |
| :---: | :---: |
| "Draw labels on two lines" | The time stamps are displayed on two lines (for example, the date is displayed on the first line and the time on the second line). The time stamp is displayed on one line. Example: 2019-11-01-12:30:50. |
| " Omit irrelevant information in timestamps" | The time stamps are displayed in a truncated form (without insignificant information). For example, the date is displayed at the first tick mark, and only the time is displayed at the following tick marks. The "Internationalization (format strings)" property is not visible and is ignored. $\square$ The time stamps are displayed with all information. This takes into consideration the "Internationalization (format strings)" property which contains the format specification for the date and time display. |
| "Internationalization (format strings)" | Format specification for the date and time display of the time stamp (when it is displayed in full) <br> Note: The property is visible only if the "Omit irrelevant information in timestamps" option is not selected. |


| "Date" | Format string that returns the date display according to the defined format. The operating system locale is used as the default setting. <br> Defined format strings for the date: <br> - Year: yyyy, yy, y <br> - Month: mM, m <br> - Day: dd, d <br> - Recommended separator: - . / <br> Example: <br> YYYy-MM-d displays 2019-10-25 <br> yyyy-MM-dd displays 2019-10-25 <br> dd.MM. yyyy displays 25 .10. 2019 <br> dd/MM/yyyy displays 25/10/2019 |
| :---: | :---: |
| "Time" | Format string that returns the time (or time of day) display according to the defined format. The operating system locale is used as the default setting. Defined format strings for the time: <br> - 24-hour time definition: $\mathrm{HH}, \mathrm{H}$ <br> - 12-hour time definition: hh, h <br> - AM/PM for 12-hour time definition: $t$ t <br> - Minutes: mm, m <br> - Seconds: ss, s <br> - Milliseconds: ms <br> - Microseconds: us <br> - Recommended separator: : or space character <br> Example: <br> HH:mm:ss:ms displays $15: 30: 59: 123$ <br> $\mathrm{h}: \mathrm{mm}$ :ss tt displays 3:30:59 PM |

## See also

- ${ }^{4}$ Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329

Element property 'Assigned control elements'

These elements are created automatically when the control elements are added with the command "Insert elements for controlling Trend".

| "Date Range Picker" | Control element for changing the date and time of the displayed data sets. <br> With [..., all elements are provided that have implemented the interface <br> IDateRangeSelector. By default, instances of the "Date Range Picker" visu- <br> alization element are available. |
| :--- | :--- |
| "Time Range Picker" | Control element for changing the time of the displayed data sets. With $\ldots .$. , all <br> elements are provided that have implemented the interface ITimeSelector. <br> By default, instances of the "Time Range Picker" visualization element are avail- <br> able. |
| "Legend" | Control element for displaying a legend for the graphs. With <br> provided that have implemented the interface ILegendDisplayer. |

See also

- $\Leftrightarrow$ Chapter 6.4.5.21.2.18 "Command 'Insert Elements for Controlling the Trend'" on page 3360

Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the $\boldsymbol{\oplus}$ symbol. The point is used as the center for rotating and scaling.

| $" X "$ | X-coordinate of the point of rotation |
| :--- | :--- |
| $" Y "$ | $Y$-coordinate of the point of rotation |



```
You can also change the values by dragging the symbols ( \(\boldsymbol{\oplus}\) ) to other positions in the editor.
```

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

| "Movement" |  |  |
| :---: | :---: | :---: |
| "X" | Variable (numeric data type). Defines the $X$ position (in pixels). <br> Example: PLC_PRG.iPos_X. <br> Increasing this value in runtime mode moves the element to the right. |  |
| "Y" | Variable (numeric data type). Defines the Y position (in pixels). <br> Example: PLC_PRG.iPos_Y. <br> Increasing this value in runtime mode moves the element downwards. |  |
| "Rotation" | Variable (numeric data type). Defines the angle of rotation (in degrees). <br> Example: PLC_PRG.iAngle1. <br> The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol. <br> In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right. |  |
| "Interior rotation" | Variable (numeric data type). Defines the angle of rotation (in degrees). <br> Example: PLC_PRG.iAngle2. <br> In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. <br> The rotation point is shown as the $\oplus$ symbol. <br> Note: If a static angle of rotation is specified in the "Position $\rightarrow$ Angle" property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed. |  |

## You can link the variables to a unit conversion.

The " $X$ ", " $Y$ ", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also
-

Element prop- The variables control the element behavior dynamically. erty 'State variables'

| "Invisible" | Variable (BOOL). Toggles the visibility of the element. <br> TRUE: The element is not visible at runtime. |
| :--- | :--- |

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.

| "Animation duration" | Defines the duration (in milliseconds) in which the element runs an animation <br> - Variable (integer value) <br> Example: Menu.tContent with VAR tContent : INT := 500; END_VAR <br> - Integer literal Example: 500 <br> Animatable properties <br> - "Absolute movement", "Movement", " $X$ ", " $Y$ " <br> - "Absolute movement", "Rotation" <br> - "Absolute movement", "Interior rotation" <br> - "Absolute movement", "Exterior rotation" <br> The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth. |
| :---: | :---: |
| "Move to foreground" | Moves the visualization element to the foreground <br> Variable (BOOL) <br> Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR <br> TRUE: At runtime, the visualization element is displayed in the foreground. <br> FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor. |

Element prop- Requirement: User management is set up for the visualization. erty 'Access
rights'

| "Access rights" | Opens the "Access rights" dialog. There you can edit the access privileges for <br> the element. <br> Status messages: |
| :--- | :--- |
|  | $\bullet$ "Not set. Full rights.": Access rights for all user groups : "operable" <br> $\bullet$ <br> "Rights are set: Limited rights": Access is restricted for at least one group. |

See also

- ※y Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

See also

- ${ }^{4}$ Chapter 6.4.5.13 "Displaying data curve with trend" on page 2928
- ${ }^{4}$ Chapter 6.4.5.13.2 "Getting Started with Trend Visualization" on page 2928
- ⓨ Chapter 6.4.5.13.3 "Programming a Trend Visualization" on page 2931
- Object 'Trend Recording'


## Visualization Element 'Legend'

Symbol:

Category: "Special Controls"

The element is used as a legend for another element (for example, a trend). The legend is assigned in the properties of the other element.
See also

- $\&$ Chapter 6.4.5.13 "Displaying data curve with trend" on page 2928


## Element proper- <br> ties

| "Element name" | Example: LegendOfTrendA <br> Optional <br> Hint: Assign individual names for elements so that they are found faster in the <br> element list. |
| :--- | :--- |
| "Type of element" | "Legend" |

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal $x$-axis runs to the right. The positive vertical $y$-axis runs downwards.

| "X" | X coordinate of the upper left corner of the element <br> Specified in pixels. <br> Example: 10. |
| :--- | :--- |
| "Y" | Y coordinate of the upper left corner of the element <br> Specified in pixels. <br> Example: 10. |
| "Width" | Specified in pixels. <br> Example: 150 |
| "Height" | Specified in pixels. <br> Example: 30 |

You can also change the values by dragging the box symbols (吊) to other positions in the editor.

## See also

- ${ }^{4}$ Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center' rotation is shown as the $\oplus$ symbol. The point is used as the center for rotating and scaling.

| " $X$ " | X-coordinate of the point of rotation |
| :--- | :--- |
| "Y" | Y-coordinate of the point of rotation |

You can also change the values by dragging the symbols ( $\mathbf{\oplus})$ to other positions in the editor.

| "Orientation" | Orientation of the element. The value is configured in the assigned element. <br> - "Horizontal" <br> - "Vertical" |
| :--- | :--- |
| "Attached element instance" | Example: Element_A |
| "Show frame" | : The element is displayed with frames. |
| "Number format" | The format of the value in printf syntax (example: \%d, \%5.2f) |

## Element Prop- <br> erty 'Layout'

Defines how many variables can be displayed at a maximum and is calculated from the row and column number.

| "Max. number of rows" | Example: 3 |
| :--- | :--- |
| "Max. number of columns" | Example: 2 |

Element Prop- The property affects the text configured in the associated element. erty 'Text properties'

| "Text format" | "Default": The text will be cut and displayed in only the part that fits into the <br> visualization element. <br> "Linebreak": The text will be wrapped in rows. <br> "Ellipsis": The text is cut and ellipsis . . . are added to indicate that something is <br> missing. |
| :--- | :--- |
| "Font" | Font of the text. The entries of the selection list are defined in the visualization <br> style. |
| "Font color" | Text color, for example Grey. The entries of the selection list are defined in the <br> visualization style. |
| "Transparency" | Transparency value (255 to 0), which defines the transparency of the corre- <br> sponding color. <br> Example: 255 : The color is opaque. 0: The color is fully transparent. |

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

| "Movement" |  |
| :--- | :--- |
| "X" | Variable (numeric data type). Defines the X position (in pixels). <br> Example: PLC_PRG.iPos_X. <br> Increasing this value in runtime mode moves the element to the right. |


| " $\mathrm{Y} "$ | Variable (numeric data type). Defines the Y position (in pixels). <br> Example: PLC_PRG. iPos_Y. <br> Increasing this value in runtime mode moves the element downwards. |
| :--- | :--- |
| "Rotation" | Variable (numeric data type). Defines the angle of <br> rotation (in degrees). <br> Example: PLC_PRG. iAngle1. <br> The midpoint of the element rotates at the "Center" <br> point. This rotation point is shown as the $\oplus$ <br> symbol. <br> In runtime mode, the alignment of the element <br> remains the same with respect to the coordinate <br> system of the visualization. Increasing the value <br> rotates the element to the right. |
| "Interior rotation" | Variable (numeric data type). Defines the angle of <br> rotation (in degrees). <br> Example: PLC_PRG. iAngle2. <br> In runtime mode, the element rotates about the <br> point of rotation specified in "Center" according to <br> the value of the variable. In addition, the alignment <br> of the element rotates according to the coordinate <br> system of the visualization. Increasing the value in <br> the code rotates clockwise. <br> The rotation point is shown as the $\oplus$ symbol. |
| Note: If a static angle of rotation is specified in the <br> "Position $\rightarrow$ Angle" property, then the static angle <br> of rotation is added to the variable angle of rotation <br> (offset) when the visualization is executed. |  |

You can link the variables to a unit conversion.

The " $X$ ", " $Y$ ", "Rotation", and "Interior rotation" properties are supported by the
"Client Animation" functionality.

See also

## -

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.

| "Animation duration" | Defines the duration (in milliseconds) in which the element runs an animation <br> -Variable (integer value) <br> Example: Menu. tContent with VAR tContent : INT $:=500 ;$ <br> END_VAR <br> - Integer literal <br> Example: 500 <br> Animatable properties <br> - "Absolute movement", "Movement", "X", " $Y$ " <br> - "Absolute movement", ""otation" <br> - "Absolute movement", "Interior rotation" <br> - "Absolute movement", "Exterior rotation" <br> The animated movement is executed when at least one value of an animatable <br> property has changed. The movement then executed is not jerky, but is smooth <br> within the specified animation duration. The visualization element travels to the <br> specified position while rotating dynamically. The transitions are smooth. |
| :--- | :--- |
| Moves the visualization element to the foreground <br> Variable (BOOL) <br> Example: bIs InForeground with VAR bIsInForeground : BooL $:=$ <br> FALSE; END_VAR |  |
| "Move to foreground" |  |
| TRUE: At runtime, the visualization element is displayed in the foreground. |  |
| FALSE: At runtime, the visualization element is displayed in the layer where it |  |
| was inserted in the visualization editor. |  |

Element prop- Requirement: User management is set up for the visualization. erty 'Access
rights'

| "Access rights" | Opens the "Access rights" dialog. There you can edit the access privileges for <br> the element. <br> Status messages: |
| :--- | :--- |
|  | - "Not set. Full rights.": Access rights for all user groups : "operable" <br> $\bullet$ <br> "Rights are set: Limited rights": Access is restricted for at least one group. |

See also

- ※y Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366


## Visualization Element 'ActiveX'

Symbol:


## Category: "Special Controls"

The element is used to link an existing ActiveX control in the visualization. You can configure the method calls and their parameters in the element properties of the "ActiveX" element.

## Element proper- <br> ties

| "Element name" | Example: GenElemInst_1 |
| :--- | :--- |
| "Type of element" | "ActiveX" |
| "Element" | Installed ActiveX component that is linked to the visualization. <br> Hint: To avoid typing errors, select the required ActiveX component by means of <br> the Input Assistant. |

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal $x$-axis runs to the right. The positive vertical $y$-axis runs downwards.

| "X" | X coordinate of the upper left corner of the element <br> Specified in pixels. <br> Example: 10. |
| :--- | :--- |
| "Y" | Y coordinate of the upper left corner of the element <br> Specified in pixels. <br> Example: 10. |
| "Width" | Specified in pixels. <br> Example: 150 |
| "Height" | Specified in pixels. <br> Example: 30 |

You can also change the values by dragging the box symbols (吊) to other positions in the editor.

See also

- \#y Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the $\boldsymbol{\oplus}$ symbol. The point is used as the center for rotating and scaling.

| " $X$ " | X-coordinate of the point of rotation |
| :--- | :--- |
| " $Y$ " | Y-coordinate of the point of rotation |

$\square$

[^43]| "Movement" |  |  |
| :---: | :---: | :---: |
| " $\times$ " | Variable (numeric data type). Defines the X position (in pixels). <br> Example: PLC_PRG.iPos_X. <br> Increasing this value in runtime mode moves the element to the right. |  |
| " $Y$ " | Variable (numeric data type). Defines the Y position (in pixels). <br> Example: PLC_PRG.iPos_Y. <br> Increasing this value in runtime mode moves the element downwards. |  |
| "Rotation" | Variable (numeric data type). Defines the angle of rotation (in degrees). <br> Example: PLC_PRG.iAngle1. <br> The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol. <br> In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right. |  |
| "Scaling" | Variable (integer data type). Causes centric stretching. <br> Example: PLC_PRG.iScaling. <br> The reference point is the "Center" property. <br> The value 1 shrinks the element by a factor of 0.001 . The value 1000 returns the element to its original size. |  |
| "Interior rotation" | Variable (numeric data type). Defines the angle of rotation (in degrees). <br> Example: PLC_PRG.iAngle2. <br> In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. <br> The rotation point is shown as the $\boldsymbol{\oplus}$ symbol. <br> Note: If a static angle of rotation is specified in the property "Position $\rightarrow$ Angle", then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed. |  |

You can link the variables to a unit conversion.

| "Invisible" | Variable (BOOL). Toggles the visibility of the element. |
| :--- | :--- |
|  | TRUE: The element is not visible at runtime. |

The "Invisible" property is supported by the "Client Animation" functionality.

Element prop- These method calls are executed during initialization. They are executed in the first cycle only. erty 'Initial calls'

| "Method calls" | Button '"Create new" <br> Creates a subnode below "Methods" with parameters for the method call. |
| :--- | :--- |
| "Methods" | " $<$ number $>$ "" <br> - "Method": Name of the method <br> - "Parameter": Parameter passed at the method call <br> - "Result parameter": Optional variable for the return value of the method |

Element prop- These method calls are executed in every cycle. They are executed in the refresh rate of the erty 'Cyclic visualization. calls'

| "Method calls" | Button '"Create new" <br> Creates a subnode below "Methods" for a method call and its parameters. |
| :--- | :--- |
| "Methods" | " $<$ number $>$ "" <br>  <br> - "Method": Name of the method <br> - "Parameter": Parameter passed at the method call <br>  <br>  |

Element prop- These method calls are executed in the refresh rate of the visualization. You define the call erty 'Conditional condition in the property "Methods $\rightarrow$ [<number $>\rightarrow$ Call condition". calls'

| "Method calls" | Button '"Create new" <br> Creates a subnode below "Methods" with a call condition and parameters for the <br> method call. |
| :--- | :--- |
| "Methods" | " $<$ number $]$ " <br> - "Method": Name of the method <br> - "Call condition": Variable (BOOL). A rising edge of this variable triggers the <br> - call of this method. <br> - "Parameter": Parameter passed at the method call <br> - "Result parameter": Optional variable for the return value of the method |

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.

| "Animation duration" | Defines the duration (in milliseconds) in which the element runs an animation <br> - Variable (integer value) <br> Example: Menu.tContent with VAR tContent : INT := 500; END_VAR <br> - Integer literal <br> Example: 500 <br> Animatable properties <br> - "Absolute movement", "Movement", " $X$ ", " $Y$ " <br> - "Absolute movement", "Rotation" <br> - "Absolute movement", "Interior rotation" <br> - "Absolute movement", "Exterior rotation" <br> The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth. |
| :---: | :---: |
| "Move to foreground" | Moves the visualization element to the foreground <br> Variable (BOOL) <br> Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR <br> TRUE: At runtime, the visualization element is displayed in the foreground. <br> FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor. |

Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'

| "Access rights" | Opens the "Access rights" dialog. There you can edit the access privileges for <br> the element. <br> Status messages: |
| :--- | :--- |
|  | - "Not set. Full rights.": Access rights for all user groups : "operable" <br> $\bullet \quad$ "Rights are set: Limited rights": Access is restricted for at least one group. |

See also

- ${ }^{*} y$ Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366


## Visualization Element 'Web Browser'

Symbol:


## Category: "Special Controls"

The element shows a website, PDF file, or video that has a URL.

## - NOTICE! <br> The display options of the "Web Browser" element depend on the operating system and the display variant of the visualization.

Requirement: The software components of the web browser are available in the runtime and configured accordingly (example: videos to be shown on Linux).

See also

- $\sum^{*}$ Chapter 6.4.5.2.6 "Displaying Web Contents" on page 2865


## Element proper- <br> ties

| "Element name" | Example: GenElemInst_59 |
| :--- | :--- |
| "Type of element" | "Web Browser" |

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal $x$-axis runs to the right. The positive vertical $y$-axis runs downwards.

| "X" | X coordinate of the upper left corner of the element <br> Specified in pixels. <br> Example: 10. |
| :--- | :--- |
| "Y" | Y coordinate of the upper left corner of the element <br> Specified in pixels. <br> Example: 10. |
| "Width" | Specified in pixels. <br> Example: 150 |
| "Height" | Specified in pixels. <br> Example: 30 |

You can also change the values by dragging the box symbols (ए-) to other positions in the editor.

See also

- ${ }^{\text {® }}$, Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the $\boldsymbol{\oplus}$ symbol. The point is used as the center for rotating and scaling.

| " $X$ " | X-coordinate of the point of rotation |
| :--- | :--- |
| "Y" | Y-coordinate of the point of rotation |

```
You can also change the values by dragging the symbols (\oplus) to other positions in the editor.
```

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

| "Movement" |  |  |
| :---: | :---: | :---: |
| " $X$ " | Variable (numeric data type). Defines the $X$ position (in pixels). <br> Example: PLC_PRG.iPos_X. <br> Increasing this value in runtime mode moves the element to the right. |  |
| "Y" | Variable (numeric data type). Defines the Y position (in pixels). <br> Example: PLC_PRG.iPos_Y. <br> Increasing this value in runtime mode moves the element downwards. |  |
| "Rotation" | Variable (numeric data type). Defines the angle of rotation (in degrees). <br> Example: PLC_PRG.iAngle1. <br> The midpoint of the element rotates at the "Center" point. This rotation point is shown as the symbol. <br> In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right. |  |
| "Scaling" | Variable (integer data type). Causes centric stretching. <br> Example: PLC_PRG.iScaling. <br> The reference point is the "Center" property. <br> The value 1 shrinks the element by a factor of 0.001 . The value 1000 returns the element to its original size. |  |
| "Interior rotation" | Variable (numeric data type). Defines the angle of rotation (in degrees). <br> Example: PLC_PRG.iAngle2. <br> In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. <br> The rotation point is shown as the $\boldsymbol{\oplus}$ symbol. <br> Note: If a static angle of rotation is specified in the property "Position $\rightarrow$ Angle", then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed. |  |



See also

Element prop- The variables control the element behavior dynamically. erty 'State variables'

| "Invisible" | Variable (BOOL). Toggles the visibility of the element. <br> TRUE: The element is not visible at runtime. |
| :--- | :--- |

The "Invisible" property is supported by the "Client Animation" functionality.

## Element prop- <br> erty 'Control <br> variables'

| "URL" | URL of the web page that is displayed in the visualization. <br> - Variable (STRING or wSTRING) <br> Example: PLC_PRG. stURL <br> - Literal in single straight quotation marks <br> Example: ' http: / /de.wikipedia.org' |
| :--- | :--- |
| "Show" | Variable (BOOL). <br> Example: PLC_PRG. bSetURL <br> Controls the display of the "Web browser" element. <br> If the variable contains a rising edge, then the visualization calls the web page <br> given in "URL" and displays its contents in the 'Web browser' visualization ele- <br> ment. |
| "Back" | Variable (BOOL). <br> Example: PLC_PRG. bGoBack <br> Controls the back navigation in the "Web browser". If the variable has a rising <br> edge, then the visualization displays the contents of the previously displayed <br> page. |
| Variable (BOOL). <br> Example: PLC_PRG. bGoForward <br> Controls the forward navigation in the "Web browser". If the variable has a rising <br> edge, then the visualization displays the contents of the previously displayed <br> page. |  |

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.

| "Animation duration" | Defines the duration (in milliseconds) in which the element runs an animation <br> - Variable (integer value) <br> Example: Menu.tContent with VAR tContent : INT := 500; END_VAR <br> - Integer literal <br> Example: 500 <br> Animatable properties <br> - "Absolute movement", "Movement", " $X$ ", " $Y$ " <br> - "Absolute movement", "Rotation" <br> - "Absolute movement", "Interior rotation" <br> - "Absolute movement", "Exterior rotation" <br> The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth. |
| :---: | :---: |
| "Move to foreground" | Moves the visualization element to the foreground <br> Variable (BOOL) <br> Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR <br> TRUE: At runtime, the visualization element is displayed in the foreground. <br> FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor. |

Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'

| "Access rights" | Opens the "Access rights" dialog. There you can edit the access privileges for <br> the element. <br> Status messages: |
| :--- | :--- |
|  | - "Not set. Full rights.": Access rights for all user groups : "operable" <br> $\bullet \quad$ "Rights are set: Limited rights": Access is restricted for at least one group. |

See also

- $y^{3}$ Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366


## Visualization Element 'Busy Symbol, Cube'

Symbol:


## Category: "Special Controls"

At runtime, this element indicates automatically that the runtime is busy or waiting for data.

## Element proper-

ties

| "Element name" | Example: Data_Transfer <br> Optional <br> Hint: Assign individual names for elements so that they are found faster in the <br> element list. |
| :--- | :--- |
| "Type of element" "Busy Symbol, Cube" |  |

Element prop- The position defines the location and size of the element in the visualization window. These erty 'Position' are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal $x$-axis runs to the right. The positive vertical $y$-axis runs downwards.

| " $X$ " | X coordinate of the upper left corner of the element <br> Specified in pixels. <br> Example: 10. |
| :--- | :--- |
| "Y" | Y coordinate of the upper left corner of the element <br> Specified in pixels. <br> Example: 10. |
| "Width" | Specified in pixels. <br> Example: 150 |
| "Height" | Specified in pixels. <br> Example: 30 |

You can also change the values by dragging the box symbols (무-) to other positions in the editor.

See also

- «y Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center' rotation is shown as the $\boldsymbol{\oplus}$ symbol. The point is used as the center for rotating and scaling.

| " $X$ " | X-coordinate of the point of rotation |
| :--- | :--- |
| " $Y$ " | Y-coordinate of the point of rotation |

You can also change the values by dragging the symbols ( $\boldsymbol{\oplus}$ ) to other positions in the editor.

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

| "Movement" |  |  |
| :---: | :---: | :---: |
| " $\times$ " | Variable (numeric data type). Defines the X position (in pixels). <br> Example: PLC_PRG.iPos_X. <br> Increasing this value in runtime mode moves the element to the right. |  |
| " $Y$ " | Variable (numeric data type). Defines the $Y$ position (in pixels). <br> Example: PLC_PRG.iPos_Y. <br> Increasing this value in runtime mode moves the element downwards. |  |
| "Rotation" | Variable (numeric data type). Defines the angle of rotation (in degrees). <br> Example: PLC_PRG.iAngle1. <br> The midpoint of the element rotates at the "Center" point. This rotation point is shown as the $\boldsymbol{\oplus}$ symbol. <br> In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right. |  |
| "Interior rotation" | Variable (numeric data type). Defines the angle of rotation (in degrees). <br> Example: PLC_PRG.iAngle2. <br> In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. <br> The rotation point is shown as the $\boldsymbol{\oplus}$ symbol. <br> Note: If a static angle of rotation is specified in the "Position $\rightarrow$ Angle" property, then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed. |  |

You can link the variables to a unit conversion.


See also

Element prop- The variables control the element behavior dynamically. erty 'State variables'

| "Invisible" | Variable (BOOL). Toggles the visibility of the element. <br>  <br> TRUE: The element is not visible at runtime. |
| :--- | :--- |

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.

| "Animation duration" | Defines the duration (in milliseconds) in which the element runs an animation <br> - Variable (integer value) <br> Example: Menu.tContent with VAR tContent : INT := 500; END_VAR <br> - Integer literal <br> Example: 500 <br> Animatable properties <br> - "Absolute movement", "Movement", " $X$ ", " $Y$ " <br> - "Absolute movement", "Rotation" <br> - "Absolute movement", "Interior rotation" <br> - "Absolute movement", "Exterior rotation" <br> The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth. |
| :---: | :---: |
| "Move to foreground" | Moves the visualization element to the foreground <br> Variable (BOOL) <br> Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR <br> TRUE: At runtime, the visualization element is displayed in the foreground. <br> FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor. |

Element prop- Requirement: User management is set up for the visualization.
erty 'Access
rights'

| "Access rights" | Opens the "Access rights" dialog. There you can edit the access privileges for <br> the element. <br> Status messages: |
| :--- | :--- |
|  | - "Not set. Full rights.": Access rights for all user groups : "operable" <br> $\bullet$ <br> "Rights are set: Limited rights": Access is restricted for at least one group. |

See also

- $\xi^{*}$ Chapter 6.4.5.21.3.1 "Dialog 'Access Rights"' on page 3366

See also

- $\Leftrightarrow$ Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872


## Visualization Element 'Busy Symbol, Flower'

## Symbol:

## Category: "Special Controls"

The element indicates that the system is busy or waiting for data.

## Element proper- <br> ties

| "Element name" | Example: Data_Transfer <br> Optional <br> Hint: Assign individual names for elements so that they are found faster in the <br> element list. |
| :--- | :--- |
| "Type of element" "Busy Symbol, Flower" |  |

Element prop- The position defines the location and size of the element in the visualization window. These erty 'Position' are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal $x$-axis runs to the right. The positive vertical $y$-axis runs downwards.

| " $X$ " | X coordinate of the upper left corner of the element <br> Specified in pixels. <br> Example: 10. |
| :--- | :--- |
| "Y" | Y coordinate of the upper left corner of the element <br> Specified in pixels. <br> Example: 10. |
| "Width" | Specified in pixels. <br> Example: 150 |
| "Height" | Specified in pixels. <br> Example: 30 |

You can also change the values by dragging the box symbols (ए-) to other positions in the editor.

## See also

- $y^{y}$ Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center' rotation is shown as the $\boldsymbol{\oplus}$ symbol. The point is used as the center for rotating and scaling.

| " $X$ " | X-coordinate of the point of rotation |
| :--- | :--- |
| " $Y$ " | $Y$-coordinate of the point of rotation |

You can also change the values by dragging the symbols (ఉ) to other positions in the editor.

Element prop- The properties contain fixed values for setting colors. erty 'Colors'

| "Frame color" |  |
| :--- | :--- |
| "Fill color" |  |
| "Transparency" | Value (0 to 255) for defining the transparency of the selected color. <br> Example 255: The color is opaque. 0: The color is completely transparent. |

See also

- $\Leftrightarrow$ Chapter 6.4.5.21.3.5 "Dialog 'Gradient Editor"' on page 3369
- « Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872

Element prop- The properties contain fixed values for setting the look of the element. erty 'Appearance'

| "Line width" | Value in pixels <br> Example: 2 <br> Note: The values 0 and 1 both result in a line weight of 1 pixel. If no line should be displayed, then the "Line style" property must be set to the option "Invisible". |
| :---: | :---: |
| "Fill attributes" | The way in which the element is filled. <br> - "Filled":The element is filled with the color from property "Colors $\rightarrow$ Fill color". <br> - "Invisible": The fill color is invisible. |
| "Line style" | Type of line representation <br> - "Solid" <br> - "Dashes" <br> - "Dots" <br> - "Dash Dot" <br> - "Dash Dot Dot" <br> - "not visible" |

You can assign variables in the "Appearance variables" property for controlling the appearance dynamically. The fixed values here are overwritten.

See also

- 宏 "Element property 'Appearance variables'" on page 3717

| "Symbol color" | Selection of a color for the flower symbol. |
| :--- | :--- |
| "Line" | Stroke width of the lines (in pixels). |

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

| "Movement" | Variable (numeric data type). Defines the X position (in pixels). <br> Example: PLC_PRG.iPos_X. <br> Increasing this value in runtime mode moves the element to the right. |
| :--- | :--- |
| "Y" | Variable (numeric data type). Defines the Y position (in pixels). <br> Example: PLC_PRG.iPos_Y. <br> Increasing this value in runtime mode moves the element downwards. |
| "Rotation" | Variable (numeric data type). Defines the angle of <br> rotation (in degrees). <br> Example: PLC_PRG. iAngle1. <br> The midpoint of the element rotates at the "Center" <br> point. This rotation point is shown as the $\boldsymbol{\oplus}$ <br> symbol. <br> In runtime mode, the alignment of the element <br> remains the same with respect to the coordinate <br> system of the visualization. Increasing the value <br> rotates the element to the right. |
| "Interior rotation" | Variable (numeric data type). Defines the angle of <br> rotation (in degrees). <br> Example: PLC_PRG. iAngle2. <br> In runtime mode, the element rotates about the <br> point of rotation specified in "Center" according to <br> the value of the variable. In addition, the alignment <br> of the element rotates according to the coordinate <br> system of the visualization. Increasing the value in <br> the code rotates clockwise. <br> The rotation point is shown as the $\oplus$ symbol. <br> Note: If a static angle of rotation is specified in the <br> "Position $\rightarrow$ Angle" property, then the static angle <br> of rotation is added to the variable angle of rotation <br> (offset) when the visualization is executed. |

You can link the variables to a unit conversion.

See also

Element prop- The variables control the element behavior dynamically. erty 'State variables'

| "Invisible" | Variable (BOOL). Toggles the visibility of the element. <br>  <br> TRUE: The element is not visible at runtime. |
| :--- | :--- |

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.

| "Animation duration" | Defines the duration (in milliseconds) in which the element runs an animation <br> - Variable (integer value) <br> Example: Menu.tContent with VAR tContent : INT := 500; END_VAR <br> - Integer literal <br> Example: 500 <br> Animatable properties <br> - "Absolute movement", "Movement", "X", " $Y$ " <br> - "Absolute movement", "Rotation" <br> - "Absolute movement", "Interior rotation" <br> - "Absolute movement", "Exterior rotation" <br> The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth. |
| :---: | :---: |
| "Move to foreground" | Moves the visualization element to the foreground <br> Variable (BOOL) <br> Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR <br> TRUE: At runtime, the visualization element is displayed in the foreground. <br> FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor. |

[^44]| "Access rights" | Opens the "Access rights" dialog. There you can edit the access privileges for <br> the element. <br> Status messages: |
| :--- | :--- |
|  | - "Not set. Full rights.": Access rights for all user groups : "operable" |
| - "Rights are set: Limited rights": Access is restricted for at least one group. |  |

See also

- ${ }^{\#}$ Chapter 6.4.5.21.3.1 "Dialog 'Access Rights"" on page 3366

See also

- $\Leftrightarrow$ Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872


## Visualization Element 'Text Editor'

Symbol:


## Category: "Special Controls"

The element shows the contents of text files that are saved on the controller. Files can be encoded in ASCII or Unicode formats.
A visualization user can also edit the text.

## Element proper- <br> ties

| "Element name" | Example: GenElemInst_1 |
| :--- | :--- |
| "Type of element" | "Text Editor" |

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal $x$-axis runs to the right. The positive vertical $y$-axis runs downwards.

| " $X "$ | X coordinate of the upper left corner of the element <br> Specified in pixels. <br> Example: 10. |
| :--- | :--- |
| " $Y$ " | Y coordinate of the upper left corner of the element <br> Specified in pixels. <br> Example: 10. |
| "Width" | Specified in pixels. <br> Example: 150 |
| "Height" | Specified in pixels. <br> Example: 30 |

You can also change the values by dragging the box symbols (ए) to other positions in the editor.

See also

- ๕ Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874


## Element prop-

 erty 'Font'| "Font name" | Non-proportional font used by the visualization to display the contents of the file <br> Example: "Courier New" |
| :--- | :--- |
| "Size" | Font size <br> Example: 12 |

Element prop-
erty 'Control
variables'
Table 720: Element property "Control variables --> File"

| "Variable" | Variable (STRING). Contains the file names and optionally the location of the file. <br> It is located in the file system of the controller. <br> Example: PLC_PRG. strFile: STRING : $=\quad$ / Documentation/ <br> Info.txt'; |
| :--- | :--- |
| "Open" | Variable (BOOL). Controls opening the file which is defined in the "Variable" <br> property <br> Example: bOpen: BOOL; <br> TRUE: The file is opened. |
| "Close" | Variable (BOOL). Controls closing the file which is defined in the "Variable" prop- <br> erty <br> Example: bClose : BOOL; <br> TRUE: The file is closed. |
| "Save" | Variable (BOOL). Controls saving the file which is defined in the "Variable" prop- <br> erty <br> Example: bStore : BOOL; <br> TRUE: The file is saved. |
| "New" | Variable (BOOL). Controls creating a new file. The name is defined in the <br> "Variable" property. <br> Example: bCreate: BOOL; <br> TRUE: A file is created and opened. |

Table 721: Element property "Control variables --> Edit"

| "Variable" | Variable (STRING). Contains the string to search for in the file <br> Example: strFind: STRING : $=~ ' a b c ' ; ~$ |
| :--- | :--- |
| "Find" | Variable (BOOL). Controls executing the search for the string in the "Variable" <br> property <br> Example: bFind: BOOL; <br> TRUE: The search is performed. The variable is automatically reset to FALSE. |
| "Find next" | Variable (BOOL). Controls the location to begin the search in the file <br> Example: bFindNext: BOOL; <br> TRUE: The search begins at the last search result location. |
|  | FALSE: The search begins at the beginning of the file. |

Table 722: Element property "Control variables --> Cursor position"

| "Line" | Variable (integer data type). Contains the line of the cursor <br> Example: iRowCursor: INT; |
| :--- | :--- |
| "Column" | Variable (integer data type). Contains the column of the cursor <br> Example: iColumnCursor: INT; |
| "Position" | Output variable (integer data type). Shows the absolute cursor position in the <br> text. <br> Example: iPosCursor: INT; |
| "Set cursor" | Variable (BOOL). Controls the setting of the cursor at a specific location <br> Example: iSetCursor: INT; <br> TRUE: The cursor is moved. The new position is defined in the "Line" and <br> "Column" properties. <br> FALSE: The "Line", "Column", and "Position" properties contain the actual <br> values. <br> Note: The variable is used as the control variable for an input event triggered <br> by a visualization user. |

Table 723: Element property "Control variables --> Selection"

| "Start position" | Output variable (integer data type). Shows the absolute position for starting the <br> text selection <br> Example: iPosSelection: INT; |
| :--- | :--- |
| "End position" | Output variable (integer data type). Shows the absolute position for ending the <br> text selection. <br> Example: iPosEndSelection: INT; |
| "Start line number" | Output variable (integer data type). Shows the line where the text selection <br> begins <br> Example: iRowSelection: INT; |
| "Start column index" | Output variable (integer data type). Shows the column where the text selection <br> begins <br> Example: iColumnSelection: INT; |
| "End line number" | Output variable (integer data type). Shows the line where the text selection ends <br> Example: iRowEndSelection: INT; |


| "End column index" | Output variable (integer data type). Shows the column where the text selection <br> ends <br> Example: iColumnEndSelection: INT; |
| :--- | :--- |
| "Line to select" | Variable (integer data type). Contains the line number that is selected <br> Note: The selection is controlled by the variables in the "Trigger selection" prop- <br> erty. |
| "Set selection" | Variable (BOOL). Controls the selection of a line. <br> Example: bSetSelection: BooL; |
| TRUE: The line from the "Line to select" property is selected and highlighted in <br> the Text Editor. <br> if the line is not in the current text segment of the Text Editor, then the text <br> segment is moved to this line. <br> Note: The variable is used as the control variable for an input event triggered <br> by a visualization user. The control variable is not reset automatically. You are <br> responsible for this to occur in the visualization. |  |

Table 724: Element property "Control variables --> Error handling"

| "Variable for error code" | Variable (integer data type). Contains the error code when an error occurs |
| :--- | :--- |
|  | Example: iError: INT; |
|  | The error codes are declared in GVL_ErrorCodes in the |
|  | VisuElemTextEditor library. To display the error text, the |
|  | VisuFctTextEditorGetErrorText () function of the library must be called. |


| "Variable for content changed" | Variable (BOOL). Shows whether the contents have changed <br> Example: bIsContentEdited: BOOL; <br> TRUE: The contents of the Text Editor have changed. |
| :--- | :--- |
| "Variable for access mode" | Variable (BOOL). Controls the access privileges to the file <br> Example: bIsReadOnly: BOOL; <br>  <br>  <br>  <br> TRUE: A visualization user has read-only permission. At runtime, the file contents <br> are highlighted in gray in the Text Editor. <br> FALSE: A visualization user has read/write permission. <br> Note: The variable overwrites the setting in the "Editor mode" property. |


| "Maximum line length" | Maximum number of characters per line |
| :--- | :--- |
| "Editor mode" | - "Read-only": A visualization user has read-only permissions to the file. At <br> runtime, the file contents are highlighted in gray in the text editor. |

## Element prop- <br> erty 'New files'

| "Encoding" | Character encoding of the new file: <br> - "ASCII" <br> - "Unicode (Little endian)" <br> - "Unicode (Big endian)" |
| :--- | :--- |
| "New line character sequence" | End of line character of the new file: <br> - "CR/LF": Normal for Windows systems <br> - "LF": Normal for UNIX systems <br> Please note: When a visualization user opens an existing file, the end-of-line <br> character of the file is detected and used automatically. |

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.

| "Animation duration" | Defines the duration (in milliseconds) in which the element runs an animation <br> - Variable (integer value) <br> Example: Menu.tContent with VAR tContent : INT := 500; END_VAR <br> - Integer literal <br> Example: 500 <br> Animatable properties <br> - "Absolute movement", "Movement", " $X$ ", " $Y$ " <br> - "Absolute movement", "Rotation" <br> - "Absolute movement", "Interior rotation" <br> - "Absolute movement", "Exterior rotation" <br> The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth. |
| :---: | :---: |
| "Move to foreground" | Moves the visualization element to the foreground <br> Variable (BOOL) <br> Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR <br> TRUE: At runtime, the visualization element is displayed in the foreground. <br> FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor. |

Element prop- Requirement: User management is set up for the visualization. erty 'Access
rights'

| "Access rights" | Opens the "Access rights" dialog. There you can edit the access privileges for <br> the element. <br> Status messages: <br>  <br>  <br>  <br> - "Not set. Full rights.": Access rights for all user groups : "operable" <br> - "Rights are set: Limited rights": Access is restricted for at least one group. |
| :--- | :--- |

See also

- $\xi^{\wedge}$ Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

See also

- ${ }^{*} \boldsymbol{y}$ Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872


## Visualization Element 'Path3D'

Symbol:


## Category: "Special Controls"

The "Path3D" visualization element graphically displays the curves of two independent records as a 3D path. It is specially designed for use with Motion Solution CNC in order to display the trajectory of a machine tool or a robot. The programmed path (path) and the path actually traveled (track) is displayed.

Although the visualization element is designed for use with Motion Solution CNC, it can also be used to display any other record. In this case the application has to provide the path data. The sample application 3D Path Generator, which is available in CODESYS Forge, shows how these data can be generated.
If the element is used together with SoftMotion CNC, then function blocks from the library SM3_CNC_Visu help to generate the data from the path and track. These function blocks are used by the sample project CNC_File_3DPath, which is stored in the installation directory of CODESYS.

- SMC_PathCopier
- SMC_PathCopierCompleteQueue
- SMC_PathCopierFile
- SMC_PositionTracker

A description of the function blocks can be found in the Library Manager in the library SM3_CNC_Visu.

The element does not work with the CODESYS HMI display variant.

See also

- CNC Example 6: Using Path3D with SoftMotion CNC
- Sample project in CODESYS Forge


## Element proper-

ties

| "Element name" | Example: GenElemInst_1 |
| :--- | :--- |
| "Type of element" | "Path3D" |

[^45]| "X" | X coordinate of the upper left corner of the element <br> Specified in pixels. <br> Example: 10. |
| :--- | :--- |
| "Y" | Y coordinate of the upper left corner of the element <br> Specified in pixels. <br> Example: 10. |
| "Width" | Specified in pixels. <br> Example: 150 |
| "Height" | Specified in pixels. <br> Example: 30 |



See also

- ${ }^{*}$ Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874


## Path description

| "Path data |  |
| :--- | :--- |
| (VisuStruct3DTrack)" | Variable of the type VisuSt ruct 3DTrack, which is declared in the IEC code. <br> Example: PLC_PRG.pc. vs 3dt. A description of the structure can be found in <br> the library manager in the library VisuElem3DPath. Iibrary. <br> The data structure describes a path or track through a certain number of points. <br> The points are determined and buffered by the application. The track typically <br> displays the last n positions, so that only a certain part of them is ever displayed <br> at any one time. VisuStruct 3DTrack. pProjection is a variable that is <br> set by the visualization element and contains information about the path/track <br> projection. It can be read (only) by the application. In addition, the methods <br> Projection. Apply or .Appl yV can be used in order to see whether the <br> transformed position lies inside or outside the visualization display area, which is <br> defined by Project ion. ElementRect. |
| Color of the path drawn |  |
| "Path color" | Path line width in pixels, e.g.: "2" |
| "Style of boundary points" | Display of the points between two successive objects in the path <br> - End points are not displayed |

Track descrip- The track data are structured in exactly the same way as the path data: VisuStruct3DTrack tion

| "Track data |  |
| :--- | :--- |
| (VisuStruct3DTrack)" | Variable of the type VisuSt ruct3DTrack, which is declared in the IEC code. <br> Example: PLC_PRG.pc.vs3dt. A description of the structure can be found in <br> the library manager in the library VisuElem3DPath. library. |
| "Track color" | Color of the track drawn |
| "Track line width" | Track line width in pixels, e.g.: "2" |

Camera control The camera position for the 3D mode is controlled with a reference to the external data structure. This structure allows the following operations:

- Shifting to the left/to the right/upwards/downwards
- Rotation around the $X / Y / Z$ axis
- Resetting of the view at $X / Y, Y / Z$ or $Z / X$ level, so that the path and the track are completely visible.

| "Control data |  |
| :--- | :--- |
| (VisuStruct3DControl)" | Variable of the type VisuStruct3DControl, which is declared in the IEC <br> code. Example: PLC_PRG.pc.vs3dc. <br> A description of the structure can be found in the library manager in the library <br> VisuElem3DPath. |
|  | The values can be set via the application itself or via the visualization element <br> "ControlPanel". The library VisuElem3DPath contains ready-to-use visualiza- <br> tion frames that provide a possible user interface for these data. |

## Additional

aspects

| "Coordinate system" | $\boxed{ }:$ : The coordinate system is displayed |
| :--- | :--- |
| "Grid" | $\boxed{\square}:$ Grid lines are displayed |
| "Grid color" | Color of the grid lines |

Highlighting Individual parts of the path can be visually highlighted. This is typically used to mark the already processed part of a track with a different color. Each point in the path is given a unique ID, which in the case of a CNC editor is linked with the object ID on which the point lies. This ID ("highlight ID") can be specified via the application so that dynamic elements/parts of the track can be highlighted.

| Highlight mode | Select one of the following highlight modes: <br> $\bullet \quad$ Only the element whose ID corresponds to the value of the variable is high- <br> lighted. <br> All elements whose ID (linked with the object ID in the case of a CNC editor) <br> is smaller than or equal to the value in Variable are highlighted. |
| :--- | :--- |
| Variable | Project variable that specifies the ID of an element. Example: <br> PLC_PRG. iVarElement ID. This "highlight ID" is taken into account for the <br> setting of the highlight mode. The variable must be set in the IEC application. |
| Highlight color |  |

## Element look

| "Frame line width" | Width of the frame around the element, in pixels, for example: "1" |
| :---: | :---: |
| "Frame line style" | Select one of these style types for the frame line: <br> - Solid <br> - Dashes <br> - Dots <br> - Dash Dot <br> - Dash Dot Dot <br> - Hollow |


| "Transparent background" | 鸟: The background of the element is displayed transparently. <br> $\square$ <br> $\square$ : The background of the element is displayed in the defined background color. |
| :--- | :--- |
| "Background color" |  |

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.

| "Animation duration" | Defines the duration (in milliseconds) in which the element runs an animation <br> - Variable (integer value) <br> Example: Menu.tContent with VAR tContent : INT := 500; END_VAR <br> - Integer literal <br> Example: 500 <br> Animatable properties <br> - "Absolute movement", "Movement", "X", " $Y$ " <br> - "Absolute movement", "Rotation" <br> - "Absolute movement", "Interior rotation" <br> - "Absolute movement", "Exterior rotation" <br> The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth. |
| :---: | :---: |
| "Move to foreground" | Moves the visualization element to the foreground <br> Variable (BOOL) <br> Example:bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR <br> TRUE: At runtime, the visualization element is displayed in the foreground. <br> FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor. |

Element prop- Requirement: User management is set up for the visualization.
erty 'Access
rights'

| "Access rights" | Opens the "Access rights" dialog. There you can edit the access privileges for <br> the element. <br> Status messages: |
| :--- | :--- |
|  | - "Not set. Full rights.": Access rights for all user groups : "operable"" |
| - "Rights are set: Limited rights": Access is restricted for at least one group. |  |

See also

- \& Chapter 6.4.5.21.3.1 "Dialog 'Access Rights"' on page 3366


## Visualization Element 'Control Panel'

Symbol:


## Category: "Special Controls"

This visualization element is used in connection with the "Path3D" visualization element. It is used for changing the position and orientation to the CNC path shown with "Path3D".

See also

- *y Chapter 6.4.5.21.5.42 "Visualization Element 'Path3D'" on page 3704


## Element proper- <br> ties

| "Element name" | Optional. <br> Hint: Assign individual names for elements so that they are found faster in the element list. <br> Example: Camera_Path_1 |
| :---: | :---: |
| "Type of element" | "Frame" |
| "Clipping" | If you have set the "Scaling type" to "Fixed", then only that part of the visualization is displayed that fits in the frame. |
| "Show frame" | Displays the frame <br> - "No frame": The displayed area of the frame does not have borders. <br> - "Frame": The displayed area of the frame has borders. <br> - "No frame with offset": The displayed area of the frame does not have a border and the displayed area of the referenced visualization is reduced inwards by one pixel as compared to the frame area. <br> The resulting gap prevents the referenced visualization from touching any adjacent elements. |
| "Scaling type" | Describes how the frame reacts when the visualization is resized: <br> - "Isotropic": The frame retains its proportions. This allows the ratio of height to width to be preserved, even if the height and width of the visualization have been changed separately. <br> - "Anisotropic": The frame depends on the size of the visualization, so that height and width of the referenced visualization can be changed separately. <br> - "Fixed": The original size of the frame is retained, regardless of the visualization size. If you have also selected the "Clipping" option, then only the fitting part is displayed. <br> - "Fixed and scrollable": The referenced visualization is displayed without scaling. If the value is greater than the window area of the frame, then scrollbars are added to the frame. To set the position of the scroll bar with a variable, use the "Scroll position variable horizontal" or "Scroll position variable vertical" property. |
| "Deactivation of the background drawing" | : To optimize the performance of the visualization, the non-animated elements of the frame element are drawn as a background bitmap. This could result in the elements not being displayed in the expected order. <br> Deactivation of the background drawing. This can prevent the behavior described above. |

[^46]| "References" | Clicking "Configure" opens the "Frame Configuration" dialog. This is used to <br> manage the referenced visualizations. <br> Caution: Visualizations can be nested at any depth by means of Frame ele- <br> ments. In order to use the "Switch to any visualization" Frame selection type <br> without any problems, a Frame must not contain more than 21 referenced <br> visualizations. For more information, see also the description for the "Input <br> configuration" of an element: Action "Switch Frame visualization". |
| :--- | :--- |
| List of the currently referenced <br> visualizations | Visualizations that have a button also have this displayed as a subnode. Each <br> interface variable is listed with the currently assigned transfer parameters. <br> Example: <br> vis_FormA |
| - iDataToDisplay_1 : PLC_PRG.iVar1 |  |
| $\bullet \quad$ iDataToDisplay_2 : PLC_PRG.iVar2 |  |
| Hint: You can change the assignment of the variables to an interface variable |  |
| here and edit the value field. Or click the "Configure" button instead. |  |

See also

- « Chapter 6.4.5.21.2.1 "Command 'Interface Editor"" on page 3340
- $\Leftrightarrow$ Chapter 6.4.5.17 "Creating a structured user interface" on page 2940
- « "Input action 'Switch Frame Visualization"' on page 3377

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal $x$-axis runs to the right. The positive vertical $y$-axis runs downwards.

| " $X$ " | X coordinate of the upper left corner of the element <br> Specified in pixels. <br> Example: 10. |
| :--- | :--- |
| "Y" | Y coordinate of the upper left corner of the element <br> Specified in pixels. <br> Example: 10. |
| "Width" | Specified in pixels. <br> Example: 150 |
| "Height" | Specified in pixels. <br> Example: 30 |

You can also change the values by dragging the box symbols (古) to other
positions in the editor.

## See also

- ${ }^{4}$ Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the $\boldsymbol{\oplus}$ symbol. The point is used as the center for rotating and scaling.

| " $X$ " | X-coordinate of the point of rotation |
| :--- | :--- |
| " $Y$ " | Y-coordinate of the point of rotation |

You can also change the values by dragging the symbols (ఉ) to other positions in the editor.

Element prop- The properties contain fixed values for setting colors. erty 'Colors'

| "Color" | Color for the element in its normal state. <br> Please note that the normal state is in effect if the expression in the <br> "Color variables $\rightarrow$ Toggle color" property is not defined or it has the value <br> FALSE. |
| :--- | :--- |
| "Alarm color" | Color for the element in alarm state. <br> Please note that the alarm state is in effect if the expression in the <br> "Color variables $\rightarrow$ Toggle color" property has the value TRUE. |
| "Transparency" | Value (0 to 255) for defining the transparency of the selected color. <br> Example 255: The color is opaque. 0: The color is completely transparent. |
| "Use gradient color" | $\boxed{\checkmark}:$ The element is displayed with a color gradient. |
| "Gradient setting" | The "Color gradient editor" dialog box opens. |
| "Frame color" | Example: "Black" |
| "Fill color" | Example: "Light gray" |

See also

- ${ }^{\text {® }}$ Chapter 6.4.5.21.3.5 "Dialog 'Gradient Editor'" on page 3369

Element prop- The properties contain fixed values for setting the look of the element. erty 'Appearance'

| "Line width" | Value in pixels |
| :--- | :--- |
|  | Example: 2 |
|  | Note: The values 0 and 1 both result in a line weight of one pixel. If no line <br> should be displayed, then the "Line style" property must be set to the option <br> "Invisible". |
| "Line style" | Type of line representation <br>  <br>  <br>  <br>  <br>  <br>  <br>  <br>  <br>  <br>  <br>  <br>  <br>  <br> - "Solid" "Dashes" "Dots" <br> - "Dash Dot" <br> - "Dash Dot Dot" <br> - "not visible" |

You can assign variables in the "Appearance variables" property for controlling the appearance dynamically. The fixed values are defined here.

See also

- « "Element property 'Appearance variables'" on page 3717

Element prop- The properties contains character strings for labeling the element. The character string can also erty 'Texts' contain a placeholder with a format definition. In runtime mode, the placeholder is replaced by the current value in the specified format.
CODESYS accepts the specified texts automatically into the "GlobalTextList" text list. Therefore, these texts can be localized.

| "Text" | Character string (without single straight quotation marks) for the labeling the <br> element. Add a line break by pressing the keyboard shortcut [CtrI] + [Enter]. <br> Example: Accesses: \%i <br> The variable that contains the current value for the placeholder is specified in the <br> property "Text variable $\rightarrow$ Text". |
| :--- | :--- |
| "Tooltip" | Character string (without single straight quotation marks) that is displayed as the <br> tooltip of an element. <br> Example: Number of valid accesses. <br> The variable that contains the current value for the placeholder is specified in the <br> property "Text variable $\rightarrow$ Tooltip". |

See also

- \#2 "Element property 'Text variables"' on page 3714
- © Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872
- Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329

Element prop- The properties contain fixed values for the text properties. erty 'Text properties'

| "Horizontal alignment" | Horizontal alignment of the text within the element. |
| :---: | :---: |
| "Vertical alignment" | Vertical alignment of the text within the element. |
| "Text format" | Definition for displaying texts that are too long <br> - "Default": The long text is truncated. <br> - "Line break": The text is split into parts. <br> - "Ellipsis": The visible text ends with "..." indicating that it is not complete. |
| "Font" | Example: "Default" : The "Font" dialog box opens. Drop-down list with style fonts. |


| "Font color" | Example: "Black" <br>  <br>  <br> ... : The "Color" dialog box opens. <br> マ: Drop-down list with style colors. |
| :--- | :--- |
|  | Whole number (value range from 0 to 255 ). This determines the transparency of <br> the respective color. <br> Example: $255:$ The color is opaque. <br> $0:$ The color is completely transparent. <br> Please note: If the color is a style color and already has a transparency value, <br> then this property is write-protected. |

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

| "Movement" | $X "$ Variable (numeric data type). Defines the X position (in pixels). <br> Example: PLC_PRG. iPos_X. <br> Increasing this value in runtime mode moves the element to the right. <br> " $Y$ " Variable (numeric data type). Defines the Y position (in pixels). <br> Example: PLC_PRG. iPos_Y. <br> Increasing this value in runtime mode moves the element downwards. <br> "Rotation" Variable (numeric data type). Defines the angle of <br> rotation (in degrees). <br> Example: PLC_PRG. iAngle1. <br> The midpoint of the element rotates at the "Center" <br> point. This rotation point is shown as the $\oplus$ <br> symbol. <br> In runtime mode, the alignment of the element <br> remains the same with respect to the coordinate <br> system of the visualization. Increasing the value <br> rotates the element to the right. <br> "Scaling" Variable (integer data type). Causes centric stretching. <br> Example: PLC_PRG. iScaling. <br> The reference point is the "Center" property. <br> The value 1 shrinks the element by a factor of 0.001. <br> to its original size.$\quad$The value 1000 returns the element |
| :--- | :--- |


| "Interior rotation" | Variable (numeric data type). Defines the angle of <br> rotation (in degrees). <br> Example: PLC_PRG. iAngle2. <br> In runtime mode, the element rotates about the <br> point of rotation specified in "Center" according to <br> the value of the variable. In addition, the alignment <br> of the element rotates according to the coordinate <br> system of the visualization. Increasing the value in <br> the code rotates clockwise. <br> The rotation point is shown as the $\oplus$ symbol. <br> Note: If a static angle of rotation is specified in the <br> property "Position $\rightarrow$ Angle", then the static angle <br> of rotation is added to the variable angle of rotation <br> (offset) when the visualization is executed. |
| :--- | :--- |



The properties " $X$ ", " $Y$ ", "Rotation", and "Interior rotation" are supported by the

See also

Element prop- The properties contains variables for moving the element. The reference point is the position of erty 'Relative the element ("Position" property). The shape of the element can change.

You can link the variables to a unit conversion. "Client Animation" functionality.
You can link the variables to a unit conversion.

| "Movement top-left" |  |
| :---: | :---: |
| " $X$ " | Variable (integer data type). It contains the number (in pixels) that the left edge is moved horizontally. Incrementing the value moves the element to the right. <br> Example: PLC_PRG.iDeltaX |
| "Y" | Variable (integer data type). It contains the number (in pixels) that the top edge is moved vertically. Incrementing the value moves the element to the down. <br> Example: $\qquad$ PRG.iDeltaY |
| "Movement bottom-right" |  |
| "X" | Variable (integer data type). It contains the number (in pixels) that the right edge is moved horizontally. Incrementing the value moves the element to the right. <br> Example: $\qquad$ _PRG.iDeltaWidth |
| "Y" | Variable (integer data type). It contains the number (in pixels) that the bottom edge is moved vertically. Incrementing the value moves the element to the down. <br> Example: $\qquad$ PRG.iDeltaHeight |

See also

- "Element property 'Absolute movement'" on page 3739


## Element prop- These properties are variables with contents that replace a format definition. erty 'Text variables'

| "Text variable" | Variable (data type compliant with the format definition). It contains what is <br> printed instead of the format definition. <br> Example: PLC_PRG. iAccesses |
| :--- | :--- |
|  | Note: The format definition is part of the text in the property "Texts $\rightarrow$ Text". <br> Note: If you specify a variable of type enumeration with text list support, then <br> the name of the enumeration data type is added automatically in angle brackets <br> after the variable name. Example: PLC_PRG.enVar <enumeration name>. <br> Then the symbolic value of the enumeration component is printed instead of the <br> numeric value when text is printed. Refer to the help page for the enumerations. |
| "Tooltip variable" | Variable (data type compliant with the format definition). It contains what is <br> printed instead of the format definition. <br> Example: PLC_PRG. iAccesses InTooltip <br> Note: The format definition is part of the text in the property "Texts $\rightarrow$ Tooltip". |

See also

- ${ }^{4}$ Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329
- そy "Element property 'Texts'" on page 3711
- $\xi^{4}$ Chapter 6.4.1.20.5.18 "Enumerations" on page 2263

Element prop- Dynamic texts are variably indexed texts of a text list. At runtime, the text is displayed that is erty 'Dynamic texts' currently indexed in the variable.

| "Text list" | Variable (string) or name of the text list as a fixed string in single straight quotation marks. <br> Example: 'Errorlist' <br> : Drop-down list with the dialogs available in the text lists. |
| :---: | :---: |
| "Text index" | Text list ID. This refers to the desired output text. <br> - As fixed string with the ID in single straight quotation marks. Example: '1' <br> - As a variable (STRING) for dynamically controlling the text output. <br> Example: strTextID <br> Sample assignment: PLC_PRG.strTextID := '1'; |
| "Tooltip index" | Text list ID. This refers to the desired output text. <br> - As fixed string with the ID in single straight quotation marks. <br> Example: '2' <br> - As a variable (STRING) for dynamically controlling the text output. <br> Example: strToolTipID <br> Sample assignment: PLC_PRG.strToolTipID := '2'; |
| See also <br> - $\boldsymbol{y y}^{4}$ Chapter 6.4.1.21.2.28 "Object 'Text List"" on page 2532 |  |
| Element property 'Font variables' | allow for dynamic control of the text display. |


| "Font name" | Variable (STRING). Includes the font of the text. <br> Example: PLC_PRG.stFontVar := 'Arial'; <br> The selection of fonts corresponds to the default "Font" dialog. |
| :---: | :---: |
| "Size" | Variable (numeric data type). Contains the font size (in pixels or points). The applied unit is specified in brackets after the variable name. <br> - <pt>: Points (default) <br> Example: PLC_PRG.iFontHeight <pt> <br> Code: iFontHeight : INT := 12; <br> - <px>: Pixels <br> Example: PLC_PRG.iFontHeight <px> <br> Code: iFontHeight : INT := 19; <br> If you click in the value field, a drop-down list opens on the right for setting the unit. <br> Hint: The font size is specified in points (example: Arial 12). Use points when the variable font size should match a font, for example if a font is set in the property "Text property $\rightarrow$ Font". |
| "Flags" | Variable (DWORD). Contains the flags for displaying fonts. <br> Flags: <br> - 1: Italics <br> - 2: Bold <br> - 4: Underline <br> - 8: Strikethrough <br> Note: You can combine the font displays by adding the coding of the flags. For example, a bold and underlined text: PLC_PRG.dwFontType :=6; |
| "Character set" | Variable (DWORD). Contains a character set number for the font. <br> The selection of character set numbers corresponds to the "Script" setting of the standard "Font" dialog. |
| "Color" | Variable (DWORD). Includes the color of the text. <br> Example: PLC_PRG.dwColorFont:= 16\#FF000000; |
| "Flags for text alignment" | Variable (integer data type). Contains the coding for text alignment. <br> Example: PLC_PRG.dwTextAlignment. <br> Coding: <br> - 0 : Top left <br> - 1: Horizontal center <br> - 2: Right <br> - 4: Vertical center <br> - 8: Bottom <br> Note: You can combine the text alignments by adding the coding of the flags. For example, a vertical and horizontal centered text: PLC_PRG.dwFontType := 5; |

Fixed values for displaying texts are set in "Text properties".

See also

- «\& "Element property 'Text properties"' on page 3711

Element prop- The Element property is used as an interface for project variables to dynamically control colors erty 'Color varia- at runtime. bles'

| "Toggle color" | The property controls the toggled color at runtime. <br> Value assignment: <br> - FALSE: The element is displayed with the color specified in the "Color" property. <br> - TRUE: The element is displayed with the color specified in the "Alarm color" property. <br> Assignment options: <br> - Placeholder for the user input variable <br> - "<toggle/tap variable>" <br> - "<NOT toggle/tap variable>" <br> The color change is not controlled by its own variable, but by a user input variable. <br> Note: Specify a variable for the mouse events "Tap" or "Toggle" in the input configuration of the element. Only then is the placeholder set. If you configure a variable in both "Toggle" and "Tap", then the variable specified in "Tap" is used. <br> Hint: Click the symbol to insert the placeholder "<toggle/tap variable>". When you activate the "Inputconfiguration", "Tap FALSE" property, then the "<NOT toggle/tap variable>" placeholder is displayed. <br> - Instance path of a project variable (BOOL) <br> Example: PLC_PRG.xColorIsToggeled <br> Note: In the code, declare and implement the variable specified here. Its value assignment determines when the color changes. |
| :---: | :---: |
| "Normal state" <br> "Alarm state" | The properties listed below control the color depending on the state. The normal state is in effect if the variable in "Color variables", "Toggle colo"" is not defined or it has the value FALSE. The alarm state is in effect if the variable in "Colorvariables", "Toggle color" has the value TRUE. |
| "Frame color" | Assignment options: <br> - Variable (DWORD) for the frame color Example: PLC_PRG.dwBorderColor <br> - Color literal Example of green and opaque: $16 \#$ FFO OFF00 |
| "Filling color" | Assignment options: <br> - Variable (DWORD) for the fill color <br> Example: PLC_PRG.dwFillColor <br> - Color literal <br> Example of gray and opaque: 16\#FF888888 |

The transparency part of the color value is evaluated only if the "Activate semitransparent drawing" option of the visualization manager is selected.

Select the "Advanced" option in the toolbar of the properties view. Then all element properties are visible.

See also

- ${ }^{4}$ Chapter 6.4.5.10.4 "Animating a color display" on page 2914

Element prop- The properties contain variables for controlling the appearance of the element dynamically. erty 'Appearance variables'

| "Line width" | Variable (integer data type). Contains the line weight (in pixels). <br> Note: The values 0 and 1 both result in a line weight of one pixel. If no line should be displayed, then the "Line style" property must be set to the option "Invisible". |
| :---: | :---: |
| "Line style" | Variable (DWORD). Controls the line style. <br> Coding: <br> - $0:$ Solid line <br> - 1: Dashed line <br> - 2: Dotted line <br> - 3: Line type "Dash Dot" <br> - 3: Line type "Dash Dot Dot" <br> - 8: Invisible: The line is not drawn. |

Element property 'Switch frame variable'

Fixed values can be set in the "Appearance" property. These values can be overwritten by dynamic variables at runtime.

## See also

- ② "Element property 'Appearance"' on page 3710

| "Variable" | - Variable (integer data type) that contains the index of the active visualization Example: PLC_PRG.uiIndexVisu <br> Hint: The "Frame Configuration" dialog includes a list of referenced visualizations. The visualizations are automatically numerically indexed via the order in the list. <br> Note: This variant of switching usually affects all connected display variants. <br> - Array element (integer data type) for index access via CURRENTCLIENTID <br> Example: PLC_PRG.aIndexVisu[CURRENTCLIENTID] <br> Note: This variant of switching applies to the current client only, and therefore only on one display variant. That is the display variant where the value change was triggered (for example, by means of user input). |
| :---: | :---: |

## See also

- Chapter 6.4.5.21.2.9 "Command 'Frame Selection"' on page 3348

Element prop- The variables control the element behavior dynamically. erty 'State varia-
bles'

| "Invisible" | Variable (BOOL). Toggles the visibility of the element. <br>  <br>  <br>  <br>  <br> TRUE: The element is not visible at runtime. <br> Example: bIsVisible with VAR bIsVisible : BOOL : $=$ FALSE; <br> END_VAR |
| :--- | :--- |
| "Deactivate inputs" | Variable (BOOL). Toggles the operability of the element. <br> TRUE: User inputs do not have any effect in runtime more. The element is shown <br> as deactivated. |

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.

| "Animation duration" | Defines the duration (in milliseconds) in which the element runs an animation <br> - Variable (integer value) <br> Example: Menu.tContent with VAR tContent : INT := 500; END_VAR <br> - Integer literal <br> Example: 500 <br> Animatable properties <br> - "Absolute movement", "Movement", "X", " $Y$ " <br> - "Absolute movement", "Rotation" <br> - "Absolute movement", "Interior rotation" <br> - "Absolute movement", "Exterior rotation" <br> The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth. |
| :---: | :---: |
| "Move to foreground" | Moves the visualization element to the foreground Variable (BOOL) <br> Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR <br> TRUE: At runtime, the visualization element is displayed in the foreground. <br> FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor. |

Element prop- The properties contain the configurations for the user input when using the mouse or keyboard. erty 'Input con- A user input defines an event and one or more actions that are executed when an event occurs. figuration'

| The "Configure" button opens the "Input Configuration" dialog. There you can create or edit user inputs. <br> Configured user inputs are listed below the events. They each include the action that is triggered and the setting <br> in short form. <br> Example: "Execute ST Code": |  |
| :--- | :--- |
| "Onc__PRG.i_x $:=0 ;$ |  |
| "OnDialogClosed" | Input event: The user closes the dialog. |
| "OnMouseDown" | Input event: The user clicks the mouse button completely in the element area. <br> The mouse button is clicked and released. |
| "OnMouseEnter" | Input event: The user clicks down on the mouse button. |
| "OnMouseLeave" | Input event: The user drags the mouse pointer to the element. |

$\left.\begin{array}{|l|l|}\hline \text { "OnMouseMove" } & \text { Input event: The user moves the mouse pointer over the element area. } \\ \hline \text { "OnMouseUp" } & \begin{array}{l}\text { Input events: } \\ \text { - } \\ \text { The user releases the mouse button within the element area. It is irrelevant } \\ \text { whether the user has previously pressed the mouse button inside or outside } \\ \text { the element area. }\end{array} \\ \text { - The user presses the mouse button within the element area, leaves the } \\ \text { element area, and then releases the mouse button. } \\ \text { Note: This CODESYS-specific triggering behavior guarantees that actions for } \\ \text { key elements are completed. A key element starts an action for "OnMouseDown" } \\ \text { and ends the action for "OnMouseUp". } \\ \text { Example: A visualization user presses the mouse button within the element area } \\ \text { of the key element and then moves the cursor position so that it lies outside the } \\ \text { element area. The action is ended anyway because "OnMouseUp" is triggered. }\end{array}\right\}$

| "Tap" | When a mouse click event occurs, the variable defined in "Variable" is described <br> in the application. The coding depends on the "Tap FALSE" and "Tap on enter if <br> captured" options. |
| :--- | :--- |
| "Variable" | Variable (BOOL) that is set on mouse click event. <br> Example: PLC_PRG . bI sTapped <br> TRUE: A mouse click event exists. It lasts as long as the user presses the mouse <br> button over the element. It ends when the button is released. |
|  | FALSE: A mouse click event does not exist. <br> Requirement: The "Tap FALSE" option is not activated. |
| "Tap FALSE" | V: The mouse click event leads to a complementary value in "Variable". <br> TRUE: A mouse click event does not exist. |
| "Tap on enter if captured" | FALSE: While the mouse click event exists. |
| pointer is dragged within the element area or not while the mouse button is <br> pressed. <br> TRUE: While the mouse click event exists and the mouse pointer is moved over <br> the element area. <br> FALSE: A mouse click event does not exist. Or the user moves the mouse <br> pointer outside of the element area while the mouse button is pressed. <br> The value is TRUE again as soon as the user moves the pointer back to the <br> element area. The mouse is then captured. |  |


| "Toggle" | With the onset of a mouse click event, the variable is set; when the mouse click <br> event is completed, the variable is reset. |
| :--- | :--- |
| "Variable" | Variable (BOOL). Its value toggled when the mouse click event is ended. This is <br> when the user releases the mouse button while the mouse pointer is over the <br> element area. <br> If the user releases the mouse button while the mouse pointer is outside of the <br> element area, then the mouse click event is not ended and the value is not <br> toggled. <br> Hint: The user can cancel a started toggle input by dragging the mouse pointer <br> out of the element area. |
| "Toggle on up if captured" | V: The value toggles regardless of where the mouse pointer is when the mouse <br> button is released. The mouse is then captured. |


| "Hotkey" | Keyboard shortcut on the element for triggering specific input actions. <br> When the keyboard shortcut event occurs, the input actions in the "Events" property are triggered. In this way, it is not the input action itself that leads to this input action, but the mouse input action. |
| :---: | :---: |
| "Key" | Key pressed for input action. <br> Example: [T] <br> Note: The following properties appear when a key is selected. |
| "Events" | - "None" <br> - "Mouse down": Pressing the key triggers the input actions that are configured in the "OnMouseDown" property. <br> - "Mouse up": Releasing the key triggers the input actions that are configured in the "OnMouseUp" property. <br> - "Mouse down/up": Pressing and releasing the key triggers the input actions that are configured in the "OnMouseDown" property and the "OnMouseUp" property. |
| "Shift" | $\checkmark$ : Combination with the Shift key Example: [Shift]+[T]. |
| "Control" | $\boxed{\square}$ : Combination with the Ctrl key Example: [Ctrl]+[T]. |
| "Alt" | - Combination with the Alt key Example: $[A t t]+[T]$. |

All keyboard shortcuts and their actions that are configured in the visualization are listed on the "Keyboard Configuration" tab.

## See also

- ⓨ Chapter 6.4.5.21.2.2 "Command 'Keyboard Configuration'" on page 3341
- $\sum^{*}$ Chapter 6.4.5.21.3.6 "Dialog 'Input Configuration'" on page 3370

See also

- ② Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872


## Visualization Element 'Date Range Picker'

Symbol:


## Category: "Date/Time Controls"

The element provides the capability of selecting the date and time range of a saved data set. The element is used with the "Trend" visualization element.

## Element proper- <br> ties

| "Element name"" | Example: DateTrend1 <br> Optional <br> Hint: Assign individual names for elements so that they are found faster in the <br> element list. |
| :--- | :--- |
| "Type of element" | Date Range Picker" |

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal $x$-axis runs to the right. The positive vertical $y$-axis runs downwards.

| " $X "$ | X coordinate of the upper left corner of the element <br> Specified in pixels. <br> Example: 10. |
| :--- | :--- |
| "Y" | Y coordinate of the upper left corner of the element <br> Specified in pixels. <br> Example: 10. |
| "Width" | Specified in pixels. <br> Example: 150 |
| "Height" | Specified in pixels. <br> Example: 30 |

You can also change the values by dragging the box symbols (ए़-) to other positions in the editor.

See also

- 乡 Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the $\boldsymbol{\oplus}$ symbol. The point is used as the center for rotating and scaling.

| " $X$ " | X-coordinate of the point of rotation |
| :--- | :--- |
| " $Y$ " | Y-coordinate of the point of rotation |

You can also change the values by dragging the symbols ( $\boldsymbol{\oplus}$ ) to other positions in the editor.

| "Show frame" | $\boxed{\square}$ : The visualization element is drawn with a frame. |
| :--- | :--- |
| "Resolution" | Resolution saved for the time stamp: "Millisecond" or "Microsecond" |
| "Attached element instance" | The element can be assigned to a "Trend" visualization element. As a result, the <br> time range of the trend element can be changed. The available visual elements <br> are selected with the help of the Input Assistant ( $\ldots$ ). |

## Element property 'Tick mark labels'

| "Two-line labelling" | The time stamps are displayed in two lines. The date is displayed in the first line and the time is displayed in the second line. $\square$ : Time stamp is displayed in one line. The date and time can also be displayed in one line depending on the formatting. |
| :---: | :---: |
| " Omit irrelevant information in time stamp" | The time stamp has a shorter form. For example, the date is displayed only for the first tick mark, and only the time for the following tick marks. The settings in "Internationalization (format strings)" are ignored for this setting. $\square$ : All information is displayed for all time stamps. |
| "Internationalization (format strings)" | Only active when the parameter "Omit irrelevant information in timestamps" is deactivated. |
| "Date" | Definition of the date format. The default setting is taken from the Windows control panel. |
| "Time" | Definition of the time format. The default setting is taken from the Windows control panel. |

Element prop- The properties contain fixed values for the text properties.
erty 'Text prop-
erties'

| "Horizontal alignment" | Horizontal alignment of the text within the element. |
| :---: | :---: |
| "Vertical alignment" | Vertical alignment of the text within the element. |
| "Font" | Example: "Default" The "Font" dialog box opens. <br> - Drop-down list with style fonts. |
| "Font color" | Example: "Black" The "Color" dialog box opens. <br> *: Drop-down list with style colors. |
| "Transparency" | Whole number (value range from 0 to 255). This determines the transparency of the respective color. <br> Example: 255: The color is opaque. <br> 0 : The color is completely transparent. <br> Please note: If the color is a style color and already has a transparency value, then this property is write-protected. |

## Element property 'Additional buttons'

| "Jump to the largest possible time stamp" | $\checkmark$ : An additional button ( $\quad$ ) is displayed for jumping to the last time stamp. |
| :---: | :---: |
| "Jump to the smallest possible time stamp" | $\checkmark$ : An additional button (14) is displayed for jumping to the first time stamp. |
| "Zoom out" | An additional button (o) is displayed for setting the current min./max. range to the maximum range. The selected range is left. |

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

| "Movement" | Variable (numeric data type). Defines the X position (in pixels). <br> Example: PLC_PRG. iPos_X. <br> Increasing this value in runtime mode moves the element to the right. |
| :--- | :--- |
| "Y" | Variable (numeric data type). Defines the Y position (in pixels). <br> Example: PLC_PRG. iPos_Y. <br> Increasing this value in runtime mode moves the element downwards. |
| "Rotation" | Variable (numeric data type). Defines the angle of <br> rotation (in degrees). <br> Example: PLC_PRG. iAngle1. <br> The midpoint of the element rotates at the "Center" <br> point. This rotation point is shown as the $\oplus$ <br> symbol. <br> In runtime mode, the alignment of the element <br> remains the same with respect to the coordinate <br> system of the visualization. Increasing the value <br> rotates the element to the right. |
| "Interior rotation" | Variable (numeric data type). Defines the angle of <br> rotation (in degrees). <br> Example: PLC_PRG. iAngle2. <br> In runtime mode, the element rotates about the <br> point of rotation specified in "Center" according to <br> the value of the variable. In addition, the alignment <br> of the element rotates according to the coordinate <br> system of the visualization. Increasing the value in <br> the code rotates clockwise. <br> The rotation point is shown as the $\oplus$ symbol. <br> Note: If a static angle of rotation is specified in the <br> "Position $\rightarrow$ Angle" property, then the static angle <br> of rotation is added to the variable angle of rotation <br> (offset) when the visualization is executed. |

You can link the variables to a unit conversion.

The " $X$ ", " $Y$ ", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also

Element prop- The variables control the element behavior dynamically. erty 'State variables'

| "Invisible" | Variable (BOOL). Toggles the visibility of the element. |
| :--- | :--- |

TRUE: The element is not visible at runtime.

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.

| "Animation duration" | Defines the duration (in milliseconds) in which the element runs an animation <br> - Variable (integer value) <br> Example: Menu.tContent with VAR tContent : INT := 500; END_VAR <br> - Integer literal <br> Example: 500 <br> Animatable properties <br> - "Absolute movement", "Movement", "X", " $Y$ " <br> - "Absolute movement", "Rotation" <br> - "Absolute movement", "Interior rotation" <br> - "Absolute movement", "Exterior rotation" <br> The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth. |
| :---: | :---: |
| "Move to foreground" | Moves the visualization element to the foreground <br> Variable (BOOL) <br> Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR <br> TRUE: At runtime, the visualization element is displayed in the foreground. <br> FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor. |

[^47]| "Access rights" | Opens the "Access rights" dialog. There you can edit the access privileges for <br> the element. <br> Status messages: |
| :--- | :--- |
|  | - "Not set. Full rights.": Access rights for all user groups : "operable" |
|  | - "Rights are set: Limited rights": Access is restricted for at least one group. |

See also

- $y^{y}$ Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366


## Visualization Element 'Time Range Picker'

Symbol:


## Category: "Date/Time Controls"

The element provides configurable buttons for setting the time range of a trend display to a defined time. In the process the end time of the previous display is left unchanged and the start time is adapted.

Element proper-
ties

| "Element name" | Optional <br> Hint: Assign individual names for elements so that they are found faster in the <br> element list. <br> Example: TimeRangeTemperature |
| :--- | :--- |
| "Type of element" | "Time range picker" |

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal $x$-axis runs to the right. The positive vertical $y$-axis runs downwards.

| "X" | X coordinate of the upper left corner of the element <br> Specified in pixels. <br> Example: 10. |
| :--- | :--- |
| "Y" | Y coordinate of the upper left corner of the element <br> Specified in pixels. <br> Example: 10. |
| "Width" | Specified in pixels. <br> Example: 150 |
| "Height" | Specified in pixels. <br> Example: 30 |

You can also change the values by dragging the box symbols ( ( $^{-}$) to other positions in the editor.

See also

- © Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

| "Orientation" | Specifies whether the time picker element is aligned horizontally or vertically in <br> the editor. <br> Hint: Change the width to height ratio of the element in the editor. |
| :--- | :--- |
| "Show frame" | $\boxed{\square}$ : The visualization element is drawn with a frame. |
| "Resolution" | Resolution saved for the time stamp: "Millisecond" or "Microsecond" |
| "Attached element instance" | Assignment to the element that processes the time picker <br> The element can be assigned for example to a "Trend" visualization element. <br> Then the time range of the trend element can be changed. The available visual <br> elements are selected with the help of the input assistance (...). <br> Example: GenElemInst_1 |

## Element prop-

erty 'Texts'

| "Text" | String label for the element. <br> Example: Zoom |
| :--- | :--- |

Element prop- The properties contain fixed values for the text properties.
erty 'Text properties'

| "Font" | Example: "Default" <br> ...: The "Font" dialog box opens. <br> v: Drop-down list with style fonts. |
| :--- | :--- |
| "Font color" | Example: "Black" <br> ...: The "Color" dialog box opens. <br> v: Drop-down list with style colors. |
| "Transparency" | Whole number (value range from 0 to 255). This determines the transparency of <br> the respective color. <br> Example: $255:$ The color is opaque. <br> $0:$ The color is completely transparent. <br> Please note: If the color is a style color and already has a transparency value, <br> then this property is write-protected. |

Property 'Times' In "Times", the buttons that the element provides at runtime are defined and configured in an array.

| "Provide "All" selection" | :Time Range Picker bar extended by "All" button. The diagram represents a time interval that covers all time stamps. |
| :---: | :---: |
| "Times" | Adds another button to the Time Range Picker bar and increases the array by one entry. An additional index is present in the property "Times $\rightarrow$ Times $\rightarrow$ Times $\rightarrow$ [<new $>$ ]". "Time" is located under this index. The configuration of the button is to be entered there. |
| "Times" <br> - " [Index]" <br> with index $\in\{0,1,2, \ldots\}$ | Array of all buttons in the time selection bar. Index corresponds to the number of buttons. <br> $x$ : The associated button is removed from the Time Range Picker bar. The configuration entry is deleted from the "Times" property list. |
| " [Index]" <br> - "Time" | (ㄴ): Time interval in standardized notation. Example: 3M for 3 months; 30 m for 30 minutes. If a time interval is indicated in the field, then the button is labelled with it. If a user clicks on the button at runtime, the command is executed to switch the diagram to this time interval. The default is empty. |

Element property 'Control variables'

| "Time" | Displays which time is currently selected. <br> Variable (STRING) <br> Example: PLC_PRG.strSelcetedTime |
| :--- | :--- |
| ""All" selected" | Displays the state of the "All" button <br> Variable (BOOL) <br> Example: PLC_PRG.AllTimesAreSelected |

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

| "Movement" |  |
| :--- | :--- |
| " $X "$ | Variable (numeric data type). Defines the X position (in pixels). <br> Example: PLC_PRG.iPos_X. <br> Increasing this value in runtime mode moves the element to the right. |
| $Y "$ | Variable (numeric data type). Defines the Y position (in pixels). <br> Example: PLC_PRG.iPos_Y. <br> Increasing this value in runtime mode moves the element downwards. |


| "Rotation" | Variable (numeric data type). Defines the angle of <br> rotation (in degrees). <br> Example: PLC_PRG. iAngle1. <br> The midpoint of the element rotates at the "Center" <br> point. This rotation point is shown as the $\oplus$ <br> symbol. <br> In runtime mode, the alignment of the element <br> remains the same with respect to the coordinate <br> system of the visualization. Increasing the value <br> rotates the element to the right. |
| :--- | :--- |
| "Interior rotation" | Variable (numeric data type). Defines the angle of <br> rotation (in degrees). <br> Example: PLC_PPRG. iAngle2. <br> In runtime mode, the element rotates about the <br> point of rotation specified in "Center" according to <br> the value of the variable. In addition, the alignment <br> of the element rotates according to the coordinate <br> system of the visualization. Increasing the value in <br> the code rotates clockwise. <br> The rotation point is shown as the $\boldsymbol{\oplus}$ symbol. |
| Note: If a static angle of rotation is specified in the <br> "Position $\rightarrow$ Angle" property, then the static angle <br> of rotation is added to the variable angle of rotation <br> (offset) when the visualization is executed. |  |

You can link the variables to a unit conversion.

The "X", "Y", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

## See also

Element prop- The variables control the element behavior dynamically.
erty 'State varia-
bles'

| "Invisible" | Variable (BOOL). Toggles the visibility of the element. <br>  <br>  $\mathrm{TRUE}:$ The element is not visible at runtime. |
| :--- | :--- |



The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.

| "Animation duration" | Defines the duration (in milliseconds) in which the element runs an animation <br> - Variable (integer value) <br> Example: Menu.tContent with VAR tContent : INT := 500; END_VAR <br> - Integer literal <br> Example: 500 <br> Animatable properties <br> - "Absolute movement", "Movement", " $X$ ", " $Y$ " <br> - "Absolute movement", "Rotation" <br> - "Absolute movement", "Interior rotation" <br> - "Absolute movement", "Exterior rotation" <br> The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth. |
| :---: | :---: |
| "Move to foreground" | Moves the visualization element to the foreground <br> Variable (BOOL) <br> Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR <br> TRUE: At runtime, the visualization element is displayed in the foreground. <br> FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor. |

Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'

| "Access rights" | Opens the "Access rights" dialog. There you can edit the access privileges for the element. <br> Status messages: <br> - "Not set. Full rights.": Access rights for all user groups : "operable" <br> - "Rights are set: Limited rights": Access is restricted for at least one group. |
| :---: | :---: |
|  | 6.4.5.21.3.1 "Dialog 'Access Rights"' on page 3366 |

## Visualization Element 'Date Picker'

Symbol:


## Category: "Date/Time Controls"

The element is a calendar that displays the current date. A user can click a day to select a date, which is saved to a variable. In addition, it can customize the time interval that the calendar displays. Clicking the calendar header changes the year. Clicking the arrows in the calendar header changes the month.

Language- The element contains language-dependent texts that are managed in the System text list. dependent texts This deals with the names of the month and the days of the week written out completely or of the element abbreviated. When the date picker is added to a visualization, CODESYS generates the text list automatically below the POU view. The IDs correspond to the standard text and therefore English terms. The text list makes it possible to translate these texts.

```
Example System text list
```

| ID | Default |
| :--- | :--- |
| Apr | Apr |
| April | April |

## See also

- \& Chapter 6.4.5.8 "Setting Up Multiple Languages" on page 2906


## Element proper- <br> ties

| "Element name" | Example: DueDateCalendar |
| :--- | :--- |
| "Type of element" | "Date Picker" |

Element prop- The position defines the location and size of the element in the visualization window. These erty 'Position' of the window. The positive horizontal $x$-axis runs to the right. The positive vertical $y$-axis runs downwards.

| "X" | X coordinate of the upper left corner of the element <br> Specified in pixels. <br> Example: 10. |
| :--- | :--- |
| "Y" | Y coordinate of the upper left corner of the element <br> Specified in pixels. <br> Example: 10. |
| "Width" | Specified in pixels. <br> Example: 150 |
| "Height" | Specified in pixels. <br> Example: 30 |

You can also change the values by dragging the box symbols (ए.) to other positions in the editor.

See also

- Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center' rotation is shown as the $\oplus$ symbol. The point is used as the center for rotating and scaling.

| " $X$ " | X-coordinate of the point of rotation |
| :--- | :--- |
| " $Y$ " | Y-coordinate of the point of rotation |

You can also change the values by dragging the symbols ( $\boldsymbol{\oplus}$ ) to other positions in the editor.

| "Variable" | Input variable (DATE). Contains the date that a user selects in the calendar. <br> Example: PLC_PRG. dtDueDate |
| :--- | :--- |
| "Design" | $\bullet$ "From style": All settings are preconfigured according to the style. <br> $\bullet$ "Explicit": The "Design settings" property is available. You can customize the <br> calendar here. |

Design settings Requirement: This property is visible only if the "Design" property is set to "Explicit". The values of the property can be predefined in the style. Then they are available in the drop-down list.

Table 725: "Header of Date Picker"

| Design of the header | Style font or user-defined font |
| :--- | :--- |
| "Font" | Style color or user-defined color |
| "Font color" |  |
| "Arrows" | Style color or user-defined color |
| "Arrow color" |  |
| "Color of printed arrow" |  |
| "Background" | "From style": The style defines whether and how a background is drawn. |
| "Draw background" | "Yes": The background is filled with the color in the "Background color" property. <br> "No": The background is not filled with a color. |
| "Fill color" | Style color or user-defined color |

Table 726: Design of the main display area

| Design of the main display <br> area |  |
| :--- | :--- |
| "Today" | Design of today |
| "Font" | Style font or user-defined font |
| "Font color" | Style color or user-defined color |
| "Draw background" | "From style": The style defines whether and which background is drawn. <br> "Yes": The background is filled with the color in the "Background color" property. <br> "No": The background is not filled with a color. |
| "Background color" | Style color or user-defined color. Used if "Yes" is selected in "Draw background". |


| "Show frame" | "From style": The style defines whether and how a frame is drawn. <br> "Yes": The frame is displayed with the following properties. <br> "No": A frame is not displayed. |
| :--- | :--- |
| "Frame color" | Used if "Yes" is selected in "Show frame". |
| "Rectangle type" |  |
| "Line width" |  |


| "Selected day" | Design of the selected day |
| :--- | :--- |
| "Font" | Style font or user-defined font |
| "Font color" | Style color or user-defined color |
| "Draw background" | "From style": The style defines whether and how a background is drawn. <br> "Yes": The background is filled with the color in the "Background color" property. <br> "No": The background is not filled with a color. |
| "Background color" | Style color or user-defined color |
| "Show frame" | "From style": The style defines whether and how a background is drawn. <br> "Yes": The frame is displayed with the following properties. <br> "No": A frame is not displayed. |
| "Frame color" | Used if "Yes" is selected in "Show frame". |
| "Rectangle type" |  |
| "Line width" |  |


| "Current month" | Design of the current month |
| :--- | :--- |
| "Font" | Style font or user-defined font |
| "Font color" | Style color or user-defined color |
| "Draw background" | "From style": The style defines whether and how a background is drawn. <br> "Yes": The background is filled with the color in the "Background color" property. <br> "No": The background is not filled with a color. |
| "Background color" | "From style": The style defines whether and how a frame is drawn. <br> "Yes": The frame is displayed with the following properties. <br> "No": A frame is not displayed. |
| "Frame color" | Used if "Yes" is selected in "Show frame". |
| "Rectangle type" |  |
| "Line width" |  |


| "Other months" | Design of the previous and subsequent months |
| :--- | :--- |
| "Font" | Style font or user-defined font |
| "Font color" | Style color or user-defined color |
| "Display other month" | Design of the previous and subsequent months |


| "Draw background" | "From style": The style defines whether and how a background is drawn. <br> "Yes": The background is filled with the color in the "Background color" property. <br> "No": The background is not filled with a color. |  |  |  |
| :--- | :--- | :---: | :---: | :---: |
| "Background color" | "Show frame" "From style": The style defines whether and how a frame is drawn. <br> "Yes": The frame is displayed with the following properties. <br> "No": A frame is not displayed. <br> "Frame color" Used if "Yes" is selected in "Show frame". <br> "Rectangle type"  |  |  |  |


| "Day of week heading" | Design of the heading with the days of the week |
| :--- | :--- |
| "Font" | Style font or user-defined font |
| "Font color" | Style color or user-defined color |
| "Draw background" | "From style": The background is filled with the style color "From style". The style <br> defines whether and how a background is drawn. <br> "Yes": The background is filled with the color in the "Background color" property. <br> "No": The background is not filled with a color. |
| "Background color" |  |
| "Show frame" | "From style": The style defines whether and how a frame is drawn. <br> "Yes": The frame is displayed with the following properties. <br> "No": A frame is not displayed. |
| "Frame color" | Used if "Yes" is selected in "Show frame". |
| "Rectangle type" | "Line width" "From style": The style defines whether and how a separator line is drawn. <br> "Yes": Display with the following properties.  |
| "Color of the separator line" | "No": A separator line is not displayed. |


| "Background" | Design of the calendar days |
| :--- | :--- |
| "Draw background" | "From style": The style defines whether and how a background is drawn. <br> "Yes": The background is filled with the color in the "Fill color" property and <br> framed in the "Frame color". <br> "No": The background is not filled with a color. |
| "Fill color" | Style color or user-defined color |
| "Frame color" |  |

[^48]| "Rows" | Number of month calendars per row (preset: 1) |
| :--- | :--- |
| "Columns" | Number of month calendars per column (preset: 1) |

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

| "Movement" | Variable (numeric data type). Defines the X position (in pixels). <br> Example: PLC_PRG. iPos_X. <br> Increasing this value in runtime mode moves the element to the right. |
| :--- | :--- |
| "Y" | Variable (numeric data type). Defines the Y position (in pixels). <br> Example: PLC_PRG.i Pos_Y. <br> Increasing this value in runtime mode moves the element downwards. |
| "Rotation" | Variable (numeric data type). Defines the angle of <br> rotation (in degrees). <br> Example: PLC_PRG. iAngle1. <br> The midpoint of the element rotates at the "Center" <br> point. This rotation point is shown as the $\oplus$ <br> symbol. <br> In runtime mode, the alignment of the element <br> remains the same with respect to the coordinate <br> system of the visualization. Increasing the value <br> rotates the element to the right. |
| "Interior rotation" | Variable (numeric data type). Defines the angle of <br> rotation (in degrees). <br> Example: PLC_PRG. iAngle2. <br> In runtime mode, the element rotates about the <br> point of rotation specified in "Center" according to <br> the value of the variable. In addition, the alignment <br> of the element rotates according to the coordinate <br> system of the visualization. Increasing the value in <br> the code rotates clockwise. <br> The rotation point is shown as the $\oplus$ symbol. <br> Note: If a static angle of rotation is specified in the <br> "Position $\rightarrow$ <br> of Angle" property, then the static angle <br> (offset) when the visualization is executed. |



You can link the variables to a unit conversion.


## See also

Element prop- The variables control the element behavior dynamically. erty 'State variables'

| "Invisible" | Variable (BOOL). Toggles the visibility of the element. |
| :--- | :--- |
|  | TRUE: The element is not visible at runtime. <br> Example: bIsVisible with VAR bIsVisible : BOOL : <br> END_VAR |
| "Deactivate inputs" | Variable (BOOL). Toggles the operability of the element. <br> TRUE: User inputs do not have any effect in runtime more. The element is shown <br> as deactivated. |

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.

| "Animation duration" | Defines the duration (in milliseconds) in which the element runs an animation <br> -Variable (integer value) <br> Example: Menu. tContent with VAR tContent : INT $:=500 ;$ <br> END_VAR <br> - Integer literal <br> Example: 500 <br> Animatable properties <br> - "Absolute movement", "Movement", "X", " $Y$ " <br> - "Absolute movement", ""otation" <br> - "Absolute movement", "Interior rotation" <br> - "Absolute movement", "Exterior rotation" <br> The animated movement is executed when at least one value of an animatable <br> property has changed. The movement then executed is not jerky, but is smooth <br> within the specified animation duration. The visualization element travels to the <br> specified position while rotating dynamically. The transitions are smooth. |
| :--- | :--- |
| Moves the visualization element to the foreground <br> Variable (BOOL) <br> Example: bIs InForeground with VAR bIsInForeground : BooL $:=$ <br> FALSE; END_VAR |  |
| "Move to foreground" |  |
| TRUE: At runtime, the visualization element is displayed in the foreground. |  |
| FALSE: At runtime, the visualization element is displayed in the layer where it |  |
| was inserted in the visualization editor. |  |

Element prop- Requirement: User management is set up for the visualization. erty 'Access
rights'

| "Access rights" | Opens the "Access rights" dialog. There you can edit the access privileges for <br> the element. <br> Status messages: |
| :--- | :--- |
|  | - "Not set. Full rights.": Access rights for all user groups : "operable" <br> $\bullet$ <br> "Rights are set: Limited rights": Access is restricted for at least one group. |

See also

- ※y Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366

See also

- $\Leftrightarrow_{2}$ Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872


## Visualization Element 'Analog Clock'

Symbol:


## Category: "Date/Time Controls"

The element is a clock that displays the current time of day. The clock can also display a random time.

## Element proper-

ties

| "Element name" | Example: GenElemInst_1 |
| :--- | :--- |
| "Type of element" | "Analog Clock" |

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal $x$-axis runs to the right. The positive vertical $y$-axis runs downwards.

| " $X "$ | X coordinate of the upper left corner of the element <br> Specified in pixels. <br> Example: 10. |
| :--- | :--- |
| "Y" | Y coordinate of the upper left corner of the element <br> Specified in pixels. <br> Example: 10. |
| "Width" | Specified in pixels. <br> Example: 150 |
| "Height" | Specified in pixels. <br> Example: 30 |

You can also change the values by dragging the box symbols (區) to other positions in the editor.

## See also

- 出 Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element prop- The properties contain fixed values for the coordinates of the point of rotation. This point of erty 'Center' rotation is shown as the $\oplus$ symbol. The point is used as the center for rotating and scaling.

| " $X$ " | X-coordinate of the point of rotation |
| :--- | :--- |
| $" Y$ " | $Y$-coordinate of the point of rotation |

You can also change the values by dragging the symbols ( $\boldsymbol{\oplus}$ ) to other positions in the editor.

## Element property 'Time Display'

| "Use system time" | 目: The system time of the PLC is displayed. |
| :--- | :--- |
| "Variable" | Variable (time data type TOD, TIME_OF_DAY). This receives the time of day that <br> is not the system time. <br> Example: PLC_PRG. todTimeTokio <br> Requirement: The "Use system time" property is not activated. |

See also

- 㘯 Chapter 6.4.1.20.5.6 "Data Type 'TIME"' on page 2237

| "Design" | $\bullet$ <br> $\bullet$ "From style": All settings are preconfigured according to the style. <br> "Explicit": The "Settings" property is available. Here you can customize the <br> analog clock. |
| :--- | :--- |

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

| "Movement" | Variable (numeric data type). Defines the X position (in pixels). <br> Example: PLC_PRG.iPos_X. <br> Increasing this value in runtime mode moves the element to the right. |
| :--- | :--- |
| "Y" | Variable (numeric data type). Defines the Y position (in pixels). <br> Example: PLC_PRG.iPos_Y. <br> Increasing this value in runtime mode moves the element downwards. |
| "Rotation" | Variable (numeric data type). Defines the angle of <br> rotation (in degrees). <br> Example: PLC_PRG.iAngle1. <br> The midpoint of the element rotates at the "Center" <br> point. This rotation point is shown as the $\oplus$ <br> symbol. <br> In runtime mode, the alignment of the element <br> remains the same with respect to the coordinate <br> system of the visualization. Increasing the value <br> rotates the element to the right. |
| "Scaling" | Variable (integer data type). Causes centric stretching. <br> Example: PLC_PRG. iscaling. <br> The reference point is the "Center" property. <br> The value 1 shrinks the element by a factor of $0.001 . ~ T h e ~ v a l u e ~$ <br> to its original size. |


| "Interior rotation" | Variable (numeric data type). Defines the angle of <br> rotation (in degrees). <br> Example: PLC_PRG. iAngle2. <br> In runtime mode, the element rotates about the <br> point of rotation specified in "Center" according to <br> the value of the variable. In addition, the alignment <br> of the element rotates according to the coordinate <br> system of the visualization. Increasing the value in <br> the code rotates clockwise. <br> The rotation point is shown as the $\oplus$ symbol. <br> Note: If a static angle of rotation is specified in the <br> property "Position $\rightarrow$ Angle", then the static angle <br> of rotation is added to the variable angle of rotation <br> (offset) when the visualization is executed. |
| :--- | :--- |
| "Use REAL values"" | Note: Only available if the device supports the use <br> of REAL coordinates. <br> V: The properties of the absolute movement are <br> interpreted as REAL values. The values are not <br> rounded. <br> The option allows for the individual fine-tuning of <br> drawing the element, for example for the visualiza- <br> tion of a smoother rotation. <br> Hint: If a horizontal or vertical line is drawn blurry <br> on a specific visualization platform, then this can <br> be corrected by an offset of 0.5px in the direction <br> of the line thickness. |

You can link the variables to a unit conversion.

The properties " $X$ ", " $Y$ ", "Rotation", and "Interior rotation" are supported by the "Client Animation" functionality.

Element prop- Requirement: The "Property" is "Explicit". Only then is the "Clock Settings" category visible. erty 'Settings'

Table 727: "Background"

| "Background color" | Color variants of the default background image |
| :--- | :--- |
|  | $\bullet$ "Yellow" |
|  | $\bullet$ "Red" |
|  | $\bullet$ "Blue" |
|  | $\bullet$ "Green" |
| $\bullet$ | "Black" |
| "Own background" | Background display with the specific "Image". Replaces the default background <br> image. |


| "Image" | Image from an image pool or library <br> Example: my Imagepool .my Image |
| :--- | :--- |
| "Transparency color" | The transparent color in the image representation. <br> Example: "White". The white parts of the image are transparent. |
| "Use background color" | V: The image background is displayed using the color defined in the <br> "Background color" property. <br> Requirement: No image reference is given in the "Image" property. |
| "Background color" | Style color or color <br> Requirement: "Use background color" is activated. |

Table 728: "Hands"

| "Hand style" | Example: "Thin arrow" |
| :--- | :--- |
| "Color hour hand" | Style color or color for the hands |
| "Color minute hand" |  |
| "Color second hand" |  |

Table 729: "Lines"

| "Lines style" | Clock face graduation <br>  <br>  <br>  <br>  <br> $\bullet$ <br> $\bullet \quad$ "None" <br> $\bullet$ "Hine": Graduation lines by hour <br> $\bullet \quad$ "Dots": Graduation dots by hour |
| :--- | :--- |
| "Color" | Color of the clock face graduation |
| "Line width" | Line weight of the clock face graduation |
| "Scale in 3D" | $\boxed{\square}:$ Representation of the clock face with 3D effect |

Table 730: "Numerics"

| "Style of numerics" | Digits on the clock face <br> $\bullet$ <br>  <br> $\bullet$ <br> $\bullet$ <br> $\bullet$ <br> $\bullet$ "Quane" "All" |
| :--- | :--- |
| "Font" | Font for displaying the digits |
| "Font color" | Font for displaying the digits |

Table 731: "Center point"

| "Color" | Color of the center of the clock |
| :--- | :--- |

Table 732: "Positioning"

| "Usage of" | $\bullet$ <br> $\bullet$ "Default style values": Presetting of the style values <br> "Individual settings": User-defined settings in the subordinate "Positioning" <br> property. |
| :--- | :--- |
| "Positioning" | Requirement: Visible when the "Usage or" property is set to "Individual settings". |


| "Numerics movement" | Value (in pixels) for shifting the digits. <br> Example: 80 |
| :--- | :--- |
| "Line movement" | Value (in pixels) for shifting the hour lines. <br> Example: 100 |
| "Hands scaling" | Factor for scaling the length of the hour hand. You can customize the exact <br> position of the hour hand relative to the background image. <br> Example: 100 |


| "Scaling type" | Defines the scaling of the height and width of the element. <br> - "Anisotropic": The background image is scaled to the size of the element The <br> height and width are scaled independently of each other. <br> "Isotropic": The background image is scaled to the size of the element, <br> retaining its proportion. The proportion of height and width is fixed. |
| :--- | :--- |

"Optimized drawing"
$\checkmark$ : The background image is drawn one time. When the hour hand moves, only the affected part of the image is redrawn.
$\square$ : The background image is redrawn in cycles.
Hint: Disable this option only for extreme exceptions.

Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

| "Movement" | Variable (numeric data type). Defines the X position (in pixels). <br> Example: PLC_PRG.iPos_X. <br> Increasing this value in runtime mode moves the element to the right. |
| :--- | :--- |
| "Y" | Variable (numeric data type). Defines the Y position (in pixels). <br> Example: PLC_PRG.iPos_Y. <br> Increasing this value in runtime mode moves the element downwards. |
| "Rotation" | Variable (numeric data type). Defines the angle of <br> rotation (in degrees). <br> Example: PLC_PRG. iAngle1. <br> The midpoint of the element rotates at the "Center" <br> point. This rotation point is shown as the $\ddagger$ <br> symbol. <br> In runtime mode, the alignment of the element <br> remains the same with respect to the coordinate <br> system of the visualization. Increasing the value <br> rotates the element to the right. |


| "Interior rotation" | Variable (numeric data type). Defines the angle of <br> rotation (in degrees). <br> Example: PLC_PRG. iAngle2. <br> In runtime mode, the element rotates about the <br> point of rotation specified in "Center" according to <br> the value of the variable. In addition, the alignment <br> of the element rotates according to the coordinate <br> system of the visualization. Increasing the value in <br> the code rotates clockwise. <br> The rotation point is shown as the $\oplus$ symbol. |
| :--- | :--- |
| Note: If a static angle of rotation is specified in the <br> "Position $\rightarrow$ Angle" property, then the static angle <br> of rotation is added to the variable angle of rotation <br> (offset) when the visualization is executed. |  |

You can link the variables to a unit conversion.

The "X", " $Y$ ", "Rotation", and "Interior rotation" properties are supported by the "Client Animation" functionality.

See also

Element prop- The variables control the element behavior dynamically.
erty 'State varia-
bles'

| "Invisible" | Variable (BOOL). Toggles the visibility of the element. <br> TRUE: The element is not visible at runtime. |
| :--- | :--- |

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.

| "Animation duration" | Defines the duration (in milliseconds) in which the element runs an animation <br> - Variable (integer value) <br> Example: Menu.tContent with VAR tContent : INT := 500; END_VAR <br> - Integer literal <br> Example: 500 <br> Animatable properties <br> - "Absolute movement", "Movement", " $X$ ", " $Y$ " <br> - "Absolute movement", "Rotation" <br> - "Absolute movement", "Interior rotation" <br> - "Absolute movement", "Exterior rotation" <br> The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth. |
| :---: | :---: |
| "Move to foreground" | Moves the visualization element to the foreground <br> Variable (BOOL) <br> Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR <br> TRUE: At runtime, the visualization element is displayed in the foreground. <br> FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor. |

See also

- « Chapter 6.4.5.5 "Designing a visualization with elements" on page 2872


## Visualization Element 'Date/Time Picker'

Symbol:


## Category: "Date/Time Controls"

The element provides the capability of selecting the date and time. The value can be changed by means of the arrow keys on the keyboard. The date can be selected from a calendar.

## Element proper- <br> ties

| "Element name" | Optional <br> Hint: Assign individual names for elements so that they are found faster in the <br> element list. <br> Example: StartDateAndTime |
| :--- | :--- |
| "Type of element" | "Date/Time Picker" |

Element property 'Position'

The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal $x$-axis runs to the right. The positive vertical $y$-axis runs downwards.

| " $X "$ | X coordinate of the upper left corner of the element <br> Specified in pixels. <br> Example: 10. |
| :--- | :--- |
| "Y" | Y coordinate of the upper left corner of the element <br> Specified in pixels. <br> Example: 10. |
| "Width" | Specified in pixels. <br> Example: 150 |
| "Height" | Specified in pixels. <br> Example: 30 |

You can also change the values by dragging the box symbols (무-) to other positions in the editor.

See also

- *a Chapter 6.4.5.5.3 "Positioning the Element, Adapting Size and Layer" on page 2874

Element property 'Center'

The properties contain fixed values for the coordinates of the point of rotation. This point of rotation is shown as the $\boldsymbol{\oplus}$ symbol. The point is used as the center for rotating and scaling.

| " $X$ " | X-coordinate of the point of rotation |
| :--- | :--- |
| $Y "$ | $Y$-coordinate of the point of rotation |

You can also change the values by dragging the symbols ( $\boldsymbol{\oplus})$ to other positions in the editor.

| "Variable" | Variable (DATE, DT, TIME, LTIME, TOD) <br> The value of the value of the variable is displayed and modified by means of the element. <br> The data type automatically determines the displayed value units: <br> - TIME: Day, hour, minute, and second (by default, milliseconds are not displayed) <br> - DATE: Year, month, and day <br> - DT: Year, month, day, hour, minute, and second <br> - TOD: Hour, minute, and second (by default, milliseconds are not displayed) <br> - LTIME: Day, hour, minute, and second (by default, milliseconds, microseconds, and nanoseconds are not displayed) |
| :---: | :---: |
| "Format string" | The format can restrict the output to individual values. <br> Example for LTIME: Format: HH:mm:ss.ms.us.ns --> displayed: 08:15:12.780.150.360 LTIME restricted: format: $\mathrm{HH}: \mathrm{mm}$--> displayed: 08:15 <br> Example for DATE: Format: yyyy/MM/dd --> displayed: 2015/12/17 . <br> Basically, all usual formats available for \%t are also supported. |
| "Design date time picker" | - "From style": All settings are preconfigured according to the style. <br> - "Explicit": The "Design settings" property is available. You can customize the calendar here. |
| "Design date picker" | - "From style": All settings are preconfigured according to the style. <br> - "Explicit": The "Design settings" property is available. You can customize the calendar here. |
| "Positioning date picker" | - "Dynamic": The calendar is adapted and positioned automatically. <br> - "Manual": The "Position settings" property is available. You can customize the calendar here. |

See also

- ${ }^{\sharp}$ Chapter 6.4.5.20.2 "Placeholders with Format Definition in the Output Text" on page 3329


## Element property 'Absolute movement'

The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

| "Movement" | $X "$ Variable (numeric data type). Defines the X position (in pixels). <br> Example: PLC_PRG.iPos_X. <br> Increasing this value in runtime mode moves the element to the right. <br> " $Y$ " Variable (numeric data type). Defines the $Y$ position (in pixels). <br> Example: PLC_PRG.iPos_Y. <br> Increasing this value in runtime mode moves the element downwards. |
| :--- | :--- |


| "Rotation" | Variable (numeric data type). Defines the angle of rotation (in degrees). <br> Example: PLC_PRG.iAngle1. <br> The midpoint of the element rotates at the "Center" point. This rotation point is shown as the $\boldsymbol{\oplus}$ symbol. <br> In runtime mode, the alignment of the element remains the same with respect to the coordinate system of the visualization. Increasing the value rotates the element to the right. |  |
| :---: | :---: | :---: |
| "Scaling" | Variable (integer data type). Causes centric stretchi <br> Example: PLC_PRG.iScaling. <br> The reference point is the "Center" property. <br> The value 1 shrinks the element by a factor of 0.00 to its original size. | The value 1000 returns the element |
| "Interior rotation" | Variable (numeric data type). Defines the angle of rotation (in degrees). <br> Example: PLC_PRG.iAngle2. <br> In runtime mode, the element rotates about the point of rotation specified in "Center" according to the value of the variable. In addition, the alignment of the element rotates according to the coordinate system of the visualization. Increasing the value in the code rotates clockwise. <br> The rotation point is shown as the $\boldsymbol{\oplus}$ symbol. <br> Note: If a static angle of rotation is specified in the property "Position $\rightarrow$ Angle", then the static angle of rotation is added to the variable angle of rotation (offset) when the visualization is executed. |  |



The properties " $X$ ", " $Y$ ", "Rotation", and "Interior rotation" are supported by the "Client Animation" functionality.

[^49]| "Invisible" | Variable (BOOL). Toggles the visibility of the element. |
| :--- | :--- |
|  | TRUE: The element is not visible at runtime. <br> Example: bIsVisible with VAR bIsVisible : BOOL : $=$ FALSE; <br> END_VAR |
| "Deactivate inputs" | Variable (BOOL). Toggles the operability of the element. <br>  <br>  <br> TRUE: User inputs do not have any effect in runtime more. The element is shown <br> as deactivated. |

The "Invisible" property is supported by the "Client Animation" functionality.

These properties are available only when you have selected the "Support client animations and overlay of native elements" option in the Visualization Manager.

| "Animation duration" | Defines the duration (in milliseconds) in which the element runs an animation <br> - Variable (integer value) <br> Example: Menu.tContent with VAR tContent : INT := 500; END_VAR <br> - Integer literal <br> Example: 500 <br> Animatable properties <br> - "Absolute movement", "Movement", " $X$ ", " $Y$ " <br> - "Absolute movement", "Rotation" <br> - "Absolute movement", "Interior rotation" <br> - "Absolute movement", "Exterior rotation" <br> The animated movement is executed when at least one value of an animatable property has changed. The movement then executed is not jerky, but is smooth within the specified animation duration. The visualization element travels to the specified position while rotating dynamically. The transitions are smooth. |
| :---: | :---: |
| "Move to foreground" | Moves the visualization element to the foreground <br> Variable (BOOL) <br> Example: bIsInForeground with VAR bIsInForeground : BOOL := FALSE; END_VAR <br> TRUE: At runtime, the visualization element is displayed in the foreground. <br> FALSE: At runtime, the visualization element is displayed in the layer where it was inserted in the visualization editor. |

Element prop- Requirement: User management is set up for the visualization. erty 'Access rights'

| "Access rights" | Opens the "Access rights" dialog. There you can edit the access privileges for <br> the element. <br>  <br> Status messages: <br>  <br>  <br> - "Not set. Full rights.": Access rights for all user groups : "operable" <br> $\bullet \quad$ "Rights are set: Limited rights": Access is restricted for at least one group. |
| :--- | :--- |

See also

- $\boldsymbol{y}^{\boldsymbol{y}}$ Chapter 6.4.5.21.3.1 "Dialog 'Access Rights'" on page 3366
6.4.5.22

Reference, visualization style editor
6.4.5.22.1 Dialog 'Create a New Visualization Style'...................................... 3749
6.4.5.22.2 Dialog 'Open Existing Style as a Copy'......................................... 3749
6.4.5.22.3 Editor 'Visualization Style Editor'.................................................. 3750

### 6.4.5.22.1 Dialog 'Create a New Visualization Style'

Symbol:
Function: The dialog prompts you to specify data for a new created style file.
Call:

- In CODESYS:

In the "Visualization Manager" object (tab "Settings", group "Style Settings"):
Clicking 3 opens a drop-down list. Click "Create and Edit Derived Style".

- In the visualization style editor:

Menu bar: "File $\rightarrow$ New Style"

| "Name" | Name of the new style. <br> Example: Style_CI |
| :--- | :--- |
| "Storage location" | Working directory for style editing |
| "Base style" | Style to base the new style on The drop-down list includes all styles that are <br> installed in the repository. <br> "<none>": The new style does not derive itself from an existing style. |
| "Visualization profile" | The profile is intended for informational purposes. For example, you find ele- <br> ments that are not preconfigured with special style entries, and information from <br> the profile. In addition, CODESYS checks in the profile whether a required style <br> is missing. <br> Example: CODESYS V3.5 SP9 |
| Click "OK" | The new style is created and opened for editing in the visualization style editor. <br> It already includes all required style entries and the localization in German (lan- <br> guage column de). |

See also

- ${ }^{4}$ Chapter 6.4.5.22.3 "Editor 'Visualization Style Editor'" on page 3750
- ${ }^{\text {® }}$ Chapter 6.4.5.21.4.2 "Object 'Visualization manager'" on page 3398
- $\Leftrightarrow$ Chapter 6.4.5.19 "Applying Visualization Styles" on page 2979


### 6.4.5.22.2 Dialog 'Open Existing Style as a Copy'

Function: This dialog prompts you to specify data for copying a style file.
Call:

- In CODESYS:

In the editor of the "Visualization Manager" object (tab "Settings", group "Style Settings", click 3 for a drop-down list). Click "Copy and Edit Style".

- In the visualization style editor:

Menu bar: "File $\rightarrow$ Open as Copy"

| ＂Style＂ | Style to be copied． <br> Example：Basic Style，3．5．9．0 <br> Note：You can also select a style from the repository． |
| :--- | :--- |
| ＂Destination＂ | Working directory for style editing |
| ＂OK＂ | A copy of the style is created and opened for editing in the visualization style <br> editor． |

## See also

－\＆Chapter 6．4．5．19＂Applying Visualization Styles＂on page 2979
－\＆Chapter 6．4．5．22．3＂Editor＇Visualization Style Editor＇＂on page 3750
－\＆Chapter 6．4．5．21．4．2＂Object＇Visualization manager＇＂on page 3398

## 6．4．5．22．3 Editor＇Visualization Style Editor＇

Symbol：略
Function：The editor is used for creating，deriving，editing，and localizing visualization styles．In addition，it makes it possible to check and install a style or a hierarchy of styles．

Call：
－In CODESYS：
In the＂Visualization Manager＂object（tab＂Settings＂，group＂Style Settings＂，click 3 for a drop－down list）．Click＂Open Style Editor＂．
－Start menu＞CODESYS installation folder＞＇CODESYS＇＞＇Visualization Style Editor＇

## Menu＇File＇

| 面＂＂New style＂ | The＂Create a New Visualization Style＂dialog box opens． |
| :--- | :--- |
|  | The＂Open Dialog＂dialog box opens． <br> This dialog prompts you to select a style file（format ．visustyle．xml）to be <br> opened and edited． |
|  | The＂Open Existing Style as Dialog＂dialog box opens． <br> This dialog prompts you to select a style that is copied，saved to the target <br> location，and opened for editing． |
| ＂Close＂ | Closes the style open in the editor． |
| ＂Save As＂ | Saves the changes of the open style． |
| 畐＂Save and Install＂ | The＂Select Visualization Style（s）＂dialog box opens． <br> This dialog prompts you to select a file to save the current settings． |
| ＂Recently opened files＂ | Saves the open visualization style and installs it to the visualization style reposi－ <br> tory． |
| ＂Abort＂ | Lists the files for selection that were last opened． |

See also
－${ }^{\star} \Rightarrow$ Chapter 6．4．5．22．1＂Dialog＇Create a New Visualization Style＇＂on page 3749

Menu＇Styles＇The commands affect the contents of the＂Style Properties＂tab．

| " "New Entry (as Child)" | Creates another style entry as a child of the selected style property. |
| :---: | :---: |
| \& $\downarrow$ "New Entry (Afterwards)" | Creates a new style entry in the list after the selected style property. |
| Ғ "Move Down" | Moves the selected style entry down. Requirement: Sort order is flat. |
| $\pm$ "Move Up" | Moves the selected style entry up. <br> Requirement: Sort order is flat. |
| Pb, "Sort Order" | Toggles between three sort orders: <br> - Flat structure and alphabetical order <br> - Flat structure and order according to the position of the entry in the XML style file <br> This position also determines the position of the property in CODESYS. The property appears, for example, in the "Properties" view below the "Values" column in the drop-down list for style properties. <br> - Hierarchical structure of entries <br> Requirement: The names of the style properties contain at least one dash. |
| $\checkmark$ "Check" | The settings of the style properties are checked for consistency errors. This check is also performed when saving the style. |

Menu 'Localiza- The commands affect the contents of the "Localization" tab.
tion'

| V4 "Add Language" | The dialog box "Add New Language" opens. The dialog prompts you to specify <br> data for creating a new language column. |
| :--- | :--- |
| Rang "Remove Selected | Removes the columns of the selected cell. |
| "Rename Selected Language"" | The "Rename Language" dialog box opens. The dialog is used for renaming the <br> column that defines the selected cell and removes all previous translations. |

Table 733: Dialog box "Add New Language"

| "Name" | Name of the new language as a language code according to ISO 639-1. <br> Examples: de, en, es, it, fr, ja |
| :--- | :--- |
| "Copy from existing" | All existing language columns are available for selection. The selected language <br> is copied with all entered translations. <br> "<do not copy text>": The new language receives a blank translation column. |

Tab 'General' This tab contains the general metadata of the open style file and allows it to be edited.

Table 734: "Identification"

| "Company" | Example: $\mathrm{xy}-\mathrm{z}$ GmbH <br> Tip: In the installed styles, CODESYS can filter by the company names specified <br> here. |
| :--- | :--- |
| "Name" | Example: Style_A |
| "Version" | User-defined version number <br> Example: 1.1 .1 .1 |

Table 735: "General Settings"

| "Base style" | Name and version of the style that the open style is based on. <br> Tip: The derived style properties from the base style are highlighted in yellow in <br> the "Style Properties" tab. |
| :--- | :--- |
| "Partial style (usable only as <br> base for other visualization <br> styles)" | 局: The style is identified as incomplete. Therefore, it can be used for other <br> styles as a base style only. <br> Example: Style only with color definitions that derive this to many other styles. <br> Note: CODESYS does not check for consistency errors of an incomplete style for <br> itself. <br> $\square:$ The style is identified as complete. |
| "From" | The "Select Base Style" dialog box opens. This dialog prompts you to select a <br> style file that is saved to the file system (and not does not have to be installed). <br> The file is used as a base style. |

Table 736: "Informational"

| "Visualization profile" | The profile is intended for informational purposes. For example, you find ele- <br> ments that are not preconfigured with special style entries, and information from <br> the profile. In addition, CODESYS checks in the profile whether a required style <br> is missing. |
| :--- | :--- |

Tab 'Style Prop- This tab lists the names of the style properties with the associated values and makes it possible erties' to edit it, even by means of the commands in the "Styles" menu.

The style properties can be defined for colors, fonts, images, and any values.
The style properties defined in a base style are derived and highlighted in yellow.

| "Name" | Name of the style property. <br> If the name contains a dash, then the Visualization Style Editor can sort the style <br> properties by the prefixed terms before the dash and display them in a hierarchy. <br> A name can contain more than one dash. |
| :--- | :--- |
| "Value" | Value that is assigned to the style property. |
| "Type" | Data type of the style property; selected from a drop-down list. <br> Note: This is possible and necessary only for specific style properties with a data <br> type that is not implicitly defined. |
| "Attribute" | "hide": The associated style property is not listed in the drop-down lists in <br> CODESYS. |
| "Used by" | Visualization element that can be configured with this style property. Can be <br> edited. |
| Comment | Example: Special setting for Bar Display. Optional. |
| Double-click a cell. | An input field opens for editing. |
| [Del] | Removes the selected row. |

Tab 'Localiza- This tab makes it possible to translate the names of the style properties into other languages. tion'

| "Name" | Lists the name of the style properties as they are defined in the "Style <br> Properties" tab. |
| :--- | :--- |
| "<language>" | Identification of the language name (as language code according to ISO 639-1) <br> in which the style property name should be translated. |
| Double-click a cell. | An input field opens for editing. |

### 6.4.6 CODESYS Visualization Support

With CODESYS Visualization Support, you get the functionality to collect and manage images in image pools and texts in text lists. For image pools, the focus is on the most optimal storage of image files. For text lists, the focus is on the localization of the texts.

### 6.4.6.1 Using an Image Pool

An image pool is a table of image files where an entry for each image is listed with the image ID, file name, thumbnail, and link type. At this central location, you add an image to your project or application and can further edit the image entry. At the location in the code where an image is accessed, the image ID can then be briefly specified instead of the file name of the image. The centrally stored images are accessed primarily in visualizations.


We recommend that you reduce the size of image files as much as possible before integrating them. This will optimize the loading time of the visualization in every visualization type: TargetVisu, WebVisu, and development system.

You can create any number of image pools in the project in order to sort and categorize the images thematically.
You can create image pools in the POU pool so that they are available globally in all applications. If necessary, it is located there next to the GlobalImagePool image pool, which is created automatically.
In a library project, you can use the object properties of an image pool to convert it into a symbol library for the visualization.

### 6.4.6.1.1 Creating an image pool

1. In the device tree, select the "Application" object.

Click "Project $\rightarrow$ Add Object $\rightarrow$ Image Pool".
$\Rightarrow$ The "Add Image Pool" dialog opens.
2. Type a name for the image pool (for example, "Images1") and click "Add".
$\Rightarrow$ The ${ }^{\leftrightarrows}$ image poolis added to the device tree.
3. Select the "ImagePool" object and click "Project $\rightarrow$ Edit Object" to open it.
4. Double-click the field in the "ID" column and assign an appropriate ID (for example, "Icon1").

Alternatively, you could also click ${ }^{\wedge} \Rightarrow$ Add Image File to add new images to the list.
5. Double-click the field in the "File Name" column. Click the ... button.
$\Rightarrow$ The "Select Image" dialog opens.
6. Click the ... button and then select the image file.
$\Rightarrow$ A thumbnail of the image file is displayed in the field of the column "Image". The name of the file is displayed in the "File name" field.

The image file can be references only by the name Images1. Icon1.

### 6.4.6.1.2 Implementing access to an image

At locations in the application where an image should to be accessed, the image ID can be briefly specified. This is especially the case with visualizations. There you can configure elements which can display images with the image ID (qualified with <name of image pool>.<image ID>).
Elements with static image display:

- *Image visualization element: "Static ID" property
- *sutton visualization element: "Static ID" property
- \& Chapter 6.4.5.21.5.30 "Visualization Element 'Image Switcher"' on page 3646 visualization element: "Image on", "Image off", and "Image clicked" properties
- Cartesian XY Chart visualization element: Configuration of the background in the Dialog: XY Chart Configuration dialog
If an element should dynamically display multiple images one after the other, then a string variable is specified for it, not the image ID itself. The variable is set programmatically (in IEC code) to change the image IDs. In this way, a dynamic image display in the visualization element is achieved.

Elements with dynamic image display:

- "Image" visualization element: "Image ID variable" property
- "Button" visualization element: "Image ID variable" property


### 6.4.6.1.3 Tab: Build

The tab contains options for compiling the object.

### 6.4.6.1.4 Running the HMI application

1. Click the | 4 |
| :--- |
| symbol. | $\Rightarrow$ The application is compiled.
2. Click the $\sigma_{\text {symbol }}$.
$\Rightarrow$ The application is downloaded to HMI device.
3. Click the start symbol.
$\Rightarrow$ The HMI application is executed. The visualization starts. The actual value from the controller is output.

Output: -24714
You can operate the visualization using the keyboard, mouse, or gestures.

### 6.4.6.1.5 For experts

When accepting the remote adapter by means of the "Copy to Project" command, the I/O dimensions with which the adapter responded are set for the first "exclusive owner" connection. In order to log all of the detected assembly instances after scanning, the definition IODRVETHERNETIP_PRINT_SCAN_RESULT must be set. By default, it is scanned by the instance ID 100-199. This can be adapted by means of the library parameters ParamScanStartOfInstanceAssem and ParamScanLastOfInstanceAssem from the library IoDrvEtherNetIP Library. This might be necessary, for example to scan in another manufacturer-specific range (assembly instance ID ranges).

### 6.4.6.1.6 Configuring a visualization background with an image

You can set an image in the background definition of a visualization. You can define the image by the name of the image pool plus the file name, as described above for a visualization element.

For more information, see: Command: ${ }^{*} \Rightarrow$ Background

### 6.4.6.1.7 Labeling an Image Element with Static Text

Requirement: A project with a visualization is open. You have an image file for a stop symbol.

1. Below the application, insert an "ImagePool" object named ImagePool_A.
2. Add an image with the Stop ID to ImagePool_A.

## STOP

3. Open the visualization and drag an "Image" element to the editor. $\Rightarrow$ The Input Assistant opens. ImagePool_A is listed on the Category tab.
4. Select the Stop image and click "OK" to close the dialog.
5. Configure its "Text" property: ImagePool_A, Stop
6. Configure the "Text properties $\rightarrow$ Horizontal alignment" property: Links.
7. Configure the "Text properties $\rightarrow$ Vertical alignment" property: Bottom.

## STOP

ImagePool_A, Stop

### 6.4.6.2 Using a Text List

A text list is a table of text entries which lists an entry with a text ID, the default text, and its translations into other languages for each text used in the project. At this central location, you can add a new text or further edit an existing text entry.
The centrally stored texts are accessed primarily by visualizations. The text list is the basis for localization, selecting a language, and changes languages in visualizations.

|  | ServiceMenue $\mathbf{x}$ |  |  |
| :--- | :--- | :--- | :--- |
| ID | Default | de | en |
| 0 | Service - Usermanagement | Service - Benutzerverwaltung | Service - Usermanagement |
| 1 | Servie - Service parameter | Service - Service Einstellung... | Servie - Service parameter |
| 2 | Service - System settings | Service - Systemeinstellungen | Service - System settings |

You can specify the texts in Unicode format so that all languages and characters are possible.
You can translate the default texts and add more languages when needed.
Moreover, you can export text lists and then translate the texts outside of the current project.
For more information, see:

- Esporting a text list
- ${ }^{4}$ Importing files with text list entries


### 6.4.6.2.1 Output of dynamic text by a variable

Besides the automatically created global text list, there are also custom text lists.
If an element should dynamically display multiple texts one after the other, then a string variable is specified for it, not the text ID itself. The variable is then set programmatically (in IEC code) to change the text IDs. In this way, a dynamic text display in the visualization element is achieved.
In custom text lists, you can translate the default texts and add more languages when needed. If language switching is implemented for visualizations, then the translation is displayed at runtime for the configured text.
Elements which can dynamically display the text referenced under the "Dynamic texts" property:

- Ey Image visualization element
- ${ }^{\text {- }}$ Frame visualization element
- $y^{4}$ Pie visualization element
- $\Leftrightarrow$ Polygon, Polyline, and Bézier Curve visualization elements
- Rectangle visualization element
- $\Leftrightarrow$ Button visualization element
- $\Leftrightarrow$ Scroll Bar visualization element
- $\Leftrightarrow$ Table visualization element
- $\Leftrightarrow$ Text Field visualization element


## Example

1. All texts about a topic (such as error handling) are listed in a text list. $\Rightarrow$

| ErrorHandling $\mathbf{x}$ |  |  |  |  |
| :--- | :--- | :--- | :---: | :---: |
| ID | Default |  |  |  |
| ID_1 | OK |  |  |  |
| ID_2 | Execute service 2 |  |  |  |
| ID_3 | Execute service 3 |  |  |  |
| ID_4 | Switch off the device |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

2. The "Text Field" visualization element is configured for dynamic display:
3. A case distinction is implemented in the application code where the text for error handling is displayed depending on the error.
$\Rightarrow$ For this purpose, a text ID from the ErrorHandling text list is assigned to the strHandlingID variable depending on the iError error.
// Text list name
strErrorHandling := 'ErrorHandling';
//Assigning text ID
CASE iError OF
2: strHandlingID := 'ID_2'; 3: strHandlingID := 'ID_3'; 4: strHandlingID := 'ID_3';
ELSE strHandlingID := 'ID_1';
END_CASE;

### 6.4.6.2.2 Managing Languages and Translations

## Adding a language and translating text

Requirement: A project is open with a text list or global text.

1. In the device tree or POU view, double-click an object of type "TextList" or "GlobalTextList".
$\Rightarrow$ The "Text List" menu is shown in the menu bar and the text list opens in the editor.
2. Click "Text List $\rightarrow$ Add Language".
3. Specify a name for the language (example: en-US). Click "OK" to exit the dialog.
$\Rightarrow$ A column is displayed with the heading en-US.
4. Type in the translation of the source text into the column.

You can use the "Rename Language" command in the context menu of the text list to correct the name of a language in the table.

## Exporting a text list

$\checkmark \quad$ Requirement: A project is open with a text list or global text.

1. Double-click the "GlobalTextList" object or a "TextList" object type.
$\Rightarrow$ The object opens.
2. Click "Text List $\rightarrow$ Import/Export Text Lists".
$\Rightarrow$ The "Import/Export" dialog opens.
3. At "Choose export file", click ... and select the directory and file name.

Example: Text_lists_exported
4. Select the "Export" option.
5. Click "OK" to close the "Import/Export" dialog.
$\Rightarrow$ CODESYS exports to a file the text list entries of all text lists of the project. The table contains a column with the text list names.

| Example | Contents of the Text_lists_exportedfile |  |  |
| :--- | :--- | :--- | :--- |
|  | TextList | Id | Default en_US |

## Preparing the Exported File for the Input Assistant

$\square$ Requirement: A file is created (example: Text_lists exported) by means of the "Import/Export Text Lists" command. It contains the texts of the text lists of the project.

1. Click "Tools $\rightarrow$ Options", "Visualization" category, "File Options" tab.
2. In "Text file for textual "List components"", click ... and select a file (example: Text_lists_exported). Click "OK" to exit the dialog.
$\Rightarrow$ When you specify a static text in the "Texts" property for an element in a visualization, CODESYS offers the source text of the file as input assistance when typing in the first letter.


## Importing files with text list entries

A file to be imported has the .csv format. The first line is a header (example: TextList Id Default en US). The other lines contain text list entries. You get this kind of file by exporting the text lists of the project to a file. There you can edit the text list entries and then import the file outside of CODESYS. When importing, CODESYS handles the text list entries differently for the GlobalTextList and for dynamic text lists.

GlobalTextList

- CODESYS does not create new text list entries for an unknown ID.
- CODESYS ignores changes that affect the ID or the source text.
- CODESYS accepts changes in the translations.

Text List

- For a new ID, CODESYS supplements the corresponding text list with a text list entry.
- For an existing ID which does not match in the source text, the source text of the text list is overwritten with the source text of the file.
- CODESYS accepts changes in the translations.

Importing a file $\quad \checkmark$ Requirement: A project is open with a text list or global text.

1. Double-click the "GlobalTextList" object or a "TextList" object type.
$\Rightarrow$ The object opens.
2. Click "Text List $\rightarrow$ Import/Export Text Lists".
$\Rightarrow$ The "Import/Export" dialog opens.
3. In the "Choose file to compare or to import" input field, click... and select the directory and file.

Example: Text_lists_corrected.csv
4. Select the "Import" option.
5. Click "OK" to exit the dialog.
$\Rightarrow$ CODESYS imports the text list entries of the file into the respective text lists.

| Example | Contents of the file Text_lists_corrected.csv |  |  |  |
| :---: | :---: | :---: | :---: | :---: |
|  | TextList | Id | Default en US |  |
|  | Text_list_A | A | Information A | Information A2_en |
|  | Text_list_A | B | Information B: | OK Information B_en: OK |
|  | Text_list_A | C | Information C | Information C_en |
|  | Text_list_A | D | Information D | Information D_en |
|  | Text_list_A | E | Information E | Information E_en |
|  | Text_list_A | F | Information F | Information $\mathrm{F}_{-}^{-}$en |
|  | Text_list_A | G | Information G | Information G_en |
|  | Alarmgroup | 2 | Warnung 2 |  |
|  | AlarmGroup | 1 | Warnung 1 |  |
|  | GlobalTextList |  | Information B | Information B_en |
|  | GlobalTextList |  | Information A | Information A_en |
|  | GlobalTextList |  | Umschalten | Switch |
|  | GlobalTextList |  | Zähler: \%i | Counter : \%i |

These contents are applied to the text lists with the same name in the project.
For more information, see: ${ }^{\$ y}$ Import/Export Text Lists

## Comparing text lists with a file and exporting differences

$\checkmark$ Requirement: A project is open with a text list or global text.

1. Double-click the "GlobalTextList" object or a "TextList" object type.
$\Rightarrow$ The object opens.
2. In the context menu, click "Text List $\rightarrow$ Import/Export Text Lists". $\Rightarrow$ The "Import/Export" dialog opens.
3. In the "Choose file to compare or to import" input field, click ... and select the directory and file name of the comparison file (example: Text_lists_corrected.csv).
4. For "Choose export file", click ... and select the directory and file which contains the comparison result.
5. Select the "Export only text differences" option.
6. Click "OK" to exit the dialog.
$\Rightarrow$ CODESYS reads the import file and compares the text list entries that have the same ID. If they do not match, then CODESYS writes the text list entries of the text list to the export file.

For the global text list, CODESYS compares the translations of the same source texts. If they do not match, then CODESYS writes the text list entries to the export file.

### 6.4.6.2.3 Using static text in GlobalTextList

The global text list is the central location for texts that are displayed in the visualization.
When you configure a text for the first time in visualization element, CODESYS creates the global text list. CODESYS fills in the table as you create more texts. Therefore, the table includes all texts automatically that you create in the project visualizations. CODESYS assigns incremental IDs as integers, beginning at 0.

You can check, update, and align the global text list with the static texts of the visualization. You cannot edit the source text or the ID directly in the table. However, you can replace a source text with another source text by creating and importing a replacement file. Menu commands are provided for this purpose.

## Configuring visualization elements with static text

A text in a "GlobalTextList" can contain a format definition.
$\checkmark$ Requirement: A project with a visualization is open. The "GlobalTextList" object contains the texts that are defined in the project visualizations.

1. Double-click the visualization.
$\Rightarrow$ The editor opens.
2. Select an element with the "Text" property (example: "Text Field").
3. Type in some text in the "Text" property (example: Static Information A).
$\Rightarrow$ CODESYS adds the text to the global text list in the POU view.

## Checking GlobalTextList

$\checkmark$ Requirement: A project with a visualization is open. The "GlobalTextList" object contains the texts that are defined in the project visualizations.

1. Double-click the "GlobalTextList" object in the POUs tree.
$\Rightarrow$ The table opens with the static texts.
2. Click "Text List $\rightarrow$ Visualization Text IDs".
$\Rightarrow$ CODESYS reports when a source text of the text list does not match the static text that is identified by the ID. The source text in the global text list and the text in the visualization with the same ID do not match.

## Updating IDs in GlobalTextList

$\checkmark$ Requirement: A project with a visualization is open. The "GlobalTextList" object contains the texts that are defined in the project visualizations.

1. In the POU tree, double-click the "GlobalTextList" object.
$\Rightarrow$ The list opens with the text list entries.
2. Click "Text List $\rightarrow$ Update Visualization Text IDs".
$\Rightarrow$ CODESYS adds text to the global text list when a text in the "Static Text" property does not match the source text in the project visualizations.

## Removing GlobalTextList and creating current IDs again

Requirement: A project with a visualization is open. The "GlobalTextList" object contains the texts that are defined in the project visualizations.

1. Right-click the "GlobalTextList" object in the POUs tree and select the "Delete" command. $\Rightarrow$ The object is removed.
2. Open a visualization.
3. Click "Visualization $\rightarrow$ Create Global Text List".
$\Rightarrow$ In the POU view, a new "GlobalTextList" object is created. The global text list contains the static text from the existing project visualizations.

## Removing IDs from GlobalTextList

$\checkmark$ Requirement: A project with a visualization is open. The "GlobalTextList" object contains the texts that were defined in the project visualizations.

1. Double-click the "GlobalTextList" object in the POUs tree.
$\Rightarrow$ The table opens with the texts.
2. Click "Text List $\rightarrow$ Remove Unused Text List Entries".
$\Rightarrow$ CODESYS removes the text list entries with IDs not referenced in the project visualizations.

## Editing GlobalTextList with a replacement file

A replacement file has the CSV format. The first row is a header: defaultold defaultnew REPLACE. The following rows contain the old source texts, the new source texts, and then the REPLACE command. Tabs, commas, and semicolons are permitted separators. A combination of separator characters in a file is not permitted.

Example (tab as separator character)

```
defaultalt defaultneu REPLACE
Information A Information A1 REPLACE
```

When you import a replacement file, CODESYS processes the replacement file row by row and performs the specified replacements in the "GlobalTextList". In addition, CODESYS replaces the previous text with the replacement text in the visualizations. If the replacement text already exists as static text, then CODESYS recognizes this and harmonizes the static text and leaves only one text list entry.

Requirement: A project is open with a text list or global text.

1. Double-click the "GlobalTextList" object.
$\Rightarrow$ The object opens.
2. Click "Text List $\rightarrow$ Import/Export Text Lists".
$\Rightarrow$ The "Import/Export" dialog opens.
3. At the "Choose file to compare or to import" input field, click ... and select the directory and file (example: ReplaceGlobalTextList.csv).
4. Select the "Import replacement file" option.
5. Click "OK" to exit the dialog.
$\Rightarrow$ The texts in the text lists and the visualizations are replaced.


### 6.4.6.2.4 Creating Text in Text Lists and Displaying Dynamically

You can create and translate texts in a text list for dynamic texts in order to display them dynamically in a visualization or in the alarm management. The object of type "Text list" can be located globally in the POU view or below an application in the device tree. It contains a table with text list entries that you can edit and extend. A text list entry consists of an ID for identification, the output text, and its translation. You can add new text list entries to a text list. Menu commands are provided for this purpose.

## Creating text lists for dynamic text display

Requirement: A project with a visualization is open.

1. Select an application in the POU view or device tree and click "Project $\rightarrow$ Add Object".
2. Select "Text List".
3. Type a name (example: Text_List_A). Click "Add" to exit the dialog.
$\Rightarrow$ A $\Leftrightarrow$ Text List type object is created.
4. Click below the "Default" column and open the input field. Type a text (example:

Information).
$\Rightarrow$ The source text is created. It is used as a key in the table and as a source text for translations.
5. Type any string in the "ID" column (example: A).
$\Rightarrow$ A text list entry is defined with source text and ID. When you configure the "Dynamic texts" property of an element in a visualization, you can select the text list Text_List_A and assign the ID A.
6. Double-click in the blank line at the end of the table below "Default" and type in more text list entries.
$\Rightarrow$

|  |  |  |  |
| :--- | :--- | :--- | :--- |
|  | Textliste_A $\times$. |  | en |
| ID | Standard |  |  |
| A | Information A |  |  |
| B | Information B |  |  |
| C | Information: OK |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Dynamically Displaying a Text

In a visualization, you can configure the dynamic output of texts which were created in a text list by configuring the "Dynamic texts" property of an element. You can directly assign a text list and an ID, as well as IEC variables, where you set the values programmatically.
$\checkmark$ Requirement: A project with visualization is open and a text list is in the device tree.

1. Open the text list (example: Text_List_A).
2. Double-click the visualization.
$\Rightarrow$ The editor opens.
3. Drag an element (example: "Rectangle") to the visualization.
4. Configure its "Dynamic texts" property by selecting one in the "Text list" property (example:
'Text_List_A') and add an ID from the text list into the "Text index" (example: 'A').
Pay attention to the single straight quotation marks. You can also assign an IEC variable of type STRING for the text list name and ID.
$\Rightarrow$ The IEC variables allow for programmatic access to the texts of the text lists.
5. Build the application, download it to the controller, and start it.
$\Rightarrow$ The visualization shows the text from the text list in the text field: Information A.

### 6.4.6.2.5 Displaying a text dynamically

You can define a text in "Texts $\rightarrow$ Text" in order to display it statically. A text in "Texts $\rightarrow$ Tooltip" is shown as a tooltip. You can configure the text in such a way that the contents of a variable are displayed with it.

You can expand a static text by exactly one placeholder with a formatting specification in order to display the contents of a variable at this location at runtime. The variable which you have assigned in the "Text variable" property is displayed. When the variable changes in the application code, its display in the visualization changes at the same time.
$\square$ Requirement: A project with a visualization is open.

1. Open the visualization and add a "Text Field" element.
$\Rightarrow$ The "Properties" view shows the configuration of the element.
2. Configure the "Texts $\rightarrow$ Text" property with File name: \%s.
$\Rightarrow$ The text contains the $\% s$ placeholder.
3. In the application in the PLC_PRG POU, declare a type-compliant variable strFileName : STRING : = 'File_A';
4. Configure the "Text variable" property of the button with PLC_PRG.strFileName.
$\Rightarrow$ The contents of the variable will be displayed instead of the placeholder at runtime.
5. Compile, download, and start the application.
$\Rightarrow$ The application runs. The visualization opens. The element displays the text: File name: File_A

A text entered in the "Texts $\rightarrow$ Text" or "Texts $\rightarrow$ Tooltip" visualization property is automatically entered in GlobalTextList under POUs. There you can translate the text into other languages.
For more information see: ${ }^{\mu}$ Using static text in GlobalTextList

### 6.4.6.3 Reference

### 6.4.6.3.1 Objects

Object: ImagePool
The "ImagePool" object contains a table with image ID assignments.
For more information, see: $y_{y}$ Using an Image Pool

| "ID" | ID of the image <br> You reference this ID, for example in the visualization of the image. |
| :--- | :--- |
| "File name" | File path of the image file <br> When you click the ... button for more settings, the "Select Image" dialog opens. |
| "Image" | Show a thumbnail of the image |
| "Link Type" | Opens the "Select Image" dialog <br> In the dialog, you define the type of link. |

## Dialog: Select Image

"Image $\quad$ Name and directory of the image file (example: "C:\Program Fileslimages\logo.bmp") PNG, SVG, and TIFF. Note that a controller might not support all formats.
Whether or not you can use images formatted as scalable vector graphics (*.svg) depends on the operating system. Any necessary information is located in the device description of the hardware vendor.
Hint: It is recommended to use only images in SVG format where the aspect ratio of width to height corresponds to the aspect ratio of the view box value. Otherwise an unexpected display might result.

Table 737: File Handling

| "Remember the |
| :--- | :--- |
| link" | | Only the link is saved. This automatically updates a change to the image file |
| :--- |
| in the image pool. |
| You need to make sure that the path of the image file does not change. |
| When saving the project as a project archive, the image file is embedded |
| into the project archive. | \left\lvert\, | "Remember the |
| :--- | :--- |
| link and embed |
| into project" |$\quad$| The image in the image pool is copied, but the link information is retained. |
| :--- |
| This makes it possible to detect changes to the image file so that, if neces- |
| sary, an update of the image can be made in the image pool. |
| The behavior is controlled with the options in the next table. |
| Note: Embedded image files increase the memory requirement of the |
| project. |\right.

Table 738: Change Tracking

| These options are available only if you have <br> selected the "Remember the link and embed <br> into project" option as described above. |  |
| :--- | :--- |
| "Reload the file automatically" | The image file is automatically updated in the <br> project without any prompts. |
| "Prompt whether to reload the file" | If the image file has changed, then you may be <br> prompted whether or not the image file should <br> be updated. |
| "Do nothing" | The image file in the image pool is not updated. |

## Object: GlobalTextList

## Symbol: 国

The object is used to manage and translate texts which are composed as static text in visualizations in the project. The object contains a table with these texts. When you compose a text in a visualization in an element in the "Texts" property, CODESYS automatically adds a line in the table. You can only edit existing text here, not compose any new text.

For more information, see: $\left.{ }^{\Downarrow}\right\rangle$ Using a Text List, $\Leftrightarrow$ Text List

Moreover, CODESYS provides the following commands in order to consolidate the "GlobalTextList":

- "Check Visualization Text IDs"
- "Update Visualization Text IDs"
- "Remove Unused Text List Entries"

The object is located in the POU view and exists at most one time.

| "ID" | Unique identifier of the text |
| :--- | :--- |
| "Default" | Source text as string with one formatting specification at the most <br> (example: Information A: \%i options) <br> If a translation is not provided below a language column, then the <br> default text is used. <br> Double-click in the field to edit the text. |
| You can add as many lan- <br> guage columns to the table <br> as you want. A language <br> column is named with <br> a language code which <br> you already specified when <br> creating the column with <br> the "Insert Language" com- <br> mand. |  |
| "<language code>" | Name of the language as a language code (example: en-US) <br> This column contains the translation of the text that is made in <br> "Default". <br> When the language code is selected as a language in the Visuali- <br> zation Manager, a visualization displays the translation when it is <br> run. A running visualization can toggle between languages at the <br> request of a user. <br> Double-click in the field to edit the text. |

## Object: Text List

Symbol:
The object is used to create, manage, and translate texts. It contains a table with texts where you can add new texts. You can select a text which you have composed here can be selected in a visualization in the "Dynamic texts" property of an element. In runtime mode, the visualization displays this text dynamically in the selected language.

For more information, see: $乡 \nu$ Using a Text List
When the object is assigned to an alarm group and is located below the "Alarm Configuration" object, CODESYS adds the texts of the alarm group to the table. You can also add texts.

| "ID" | Unique identifier of the text |
| :--- | :--- |
| "Default" | Source text as string (example: Information A: \%i <br> options). Use the [Ctrl] + [Enter] shortcut to add a line break. <br> Double-click in the field to edit the text. |


| You can add as many <br> language columns to the <br> table as you want. A lan- <br> guage column is named <br> with a language code which <br> you already specified when <br> creating the column with <br> the "nsert Language" com- <br> mand. |  |
| :--- | :--- |
| "<language code>" | Name of the language as a language code. Example: en-US. <br> This column contains the translation of the text which is com- <br> posed in "Default". Under the condition that the language code <br> is selected as the language in the Visualization Manager, a visu- <br> alization displays the transslated text in runtime mode. If a transla- <br> tion has not been composed, then CODESYS uses the text in <br> "Default". A visualization in runtime mode can also change the <br> language if requested by a user. |
|  | Edit the line to add your own text. |
| Blank line |  |

### 6.4.6.3.2 Menu Commands

## Image Pool

Command: Insert Image
Symbol: 图
Function: The command inserts a new line into an image pool.
Call: "Image Pool" menu; context menu
Requirements: An image pool is active and a line is selected in the image pool.
For more information, see: $\Rightarrow$ Using an Image Pool

## Text List

## Command: Add Language

Symbol: 国
Function: Another language column is added to the text list.
Call: "Text List" menu; context menu
Requirement: A text list or global text list is open and active.
In the "Choose Language" dialog, specify a language abbreviation for the new language (example: "en-US"). This language code is then inserted as a column heading.

## Command: Create Global Text List

Symbol:
Function: The GlobalTextList global text list is created in the "POUs" view.
Call: "Visualization" menu; context menu
Requirement: A visualization is open.
For more information, see: ${ }^{y} \boldsymbol{y}$ GlobalTextList

## Command: Export All Unicode .txt Text List Files

Symbol: 图
Function: All text lists of the project are exported.
Call: "Text List" menu; context menu

## Requirements:

- A text list or global text list is open and active.
- The visualization encodes the characters of the texts into Unicode.
- The "Use unicode strings" option is selected in the Visualization Manager.
- The VISU_USEWSTRING compiler statement of the application is set. To check this, select the "Properties" command in the context menu. Then click the "Build" tab. VISU_USEWSTRING is specified in the input field for "Compiler definitions".
At this time, a simple text file (.txt format) is created for each text list. The name of the text list is used as the file name. The directory where the files are exported is defined in "Project $\rightarrow$ Project Settings $\rightarrow$ Visualization", on the "General" tab, in "Text list files".
A controller can read and use this format. You can copy the file to a controller, for example, and configure a setting in the Visualization Manager so that the text lists are not transferred again when the application is downloaded.
For more information, see: Using a Text List


## Command: Insert Text

Symbol: ㅍ]
Function: A new line is inserted above the selected line in the text list. Below "Default", an input field opens where you specify the output text.

Call: "Text List" menu; context menu
Requirement: A text list (not a "GlobalTextList") is open and active. A field is selected in the table.

## Command: Import/Export Text Lists

Symbol:
Function: The command exports an active text list, imports a file, or aligns a text list with a file. The file is in CSV format. The "Import/Export" dialog provides options for this.
Call: "Text List" menu; context menu
Requirement: A text list or global text list is open and active.

## Dialog: Import/Export

| "Choose file to <br> compare or to import" | File which is read <br>  <br> "...: Opens the "Choose Text List File" dialog for you to select a file |
| :--- | :--- |
| "Choose export file" | File to which is written <br>  <br>  <br> file and directory <br> fil Opens the "Choose Text List File" dialog where you can select a |

Table 739: "Import/Export Type"

| "Import" | Requirement: A file is selected in "Choose file to compare or to import". <br> The file can contain text list entries for both the global text list and text lists. <br> Global text list <br> - The file is read. Then the text list entries are compared for the same source text and the differences in the translations are accepted. If necessary, the translations are overwritten in the project. <br> Text lists <br> - The file is read. Then the text list entries are compared for the same IDs and the differences in the source text and translations are accepted into the project. If necessary, the text list entries in the project are overwritten. <br> - If the file contains a new ID, then the text list entry is imported into the text list of the project and the text list is added. |
| :---: | :---: |
| "Import replacement file" | Requirement: A replacement file is selected in "Choose file to compare or to import". <br> The replacement file contains replacements for the global text list. <br> The replacement file is processed row by row and the specified replacements are performed in the global text list. <br> The structure of the replacement file is described in the " $\Leftrightarrow$ Using static text in GlobalText- List" chapter. |
| "Export" | Requirement: The file which will be written to is selected in "Select export file". <br> All texts from all text lists of the current project are exported. All languages available in the project are inserted as columns in the export file. The file can be used for the external translation of the language-dependent texts. |
| "Export only text differences" | Requirement: An import file is selected for the comparison in "Choose file to compare or to import", and an export file to be written to is selected in "Select export file". <br> The import file is read and the lines of the active text list are compared with it. When lines match, they are ignored. <br> When lines differ, the line is written to the export file. Translations from the text list are accepted if necessary. The translations from the import file are accepted and overwritten if necessary. |
| "Use qualified path" | Text lists are exported and imported with full and qualified path name. <br> Example: Device.Application.TextList <br> As a result, text lists are found uniquely, even if they occur multiple times in the project with the same name. |

For more information, see: ${ }^{y} y$ Using a Text List

## Command: Remove Language

Symbol: 国
Function: The selected language column is removed from the text list.
Call: "Text List" menu; context menu
Requirement: A text list or global text list is open and active. A field is selected in the language column which you want to remove.
For more information, see: ${ }^{4} y$ Using a Text List

## Command：Rename Language

Symbol：国
Function：A dialog opens for specifying a new name for a language which is displayed in the text list as a column heading．

Call：＂Text List＂menu；context menu
Requirement：A text list or global text list is open and active．A field is selected in the language column which you want to rename．
For more information，see：$\Longleftrightarrow$ Using a Text List

## Command：Remove Unused Text List Entries

Symbol：图
Function：The command checks whether or not a text list entry in the project is used as a static text．If this is the case，then the text list entry is removed from the text list．
Call：＂Text List＂menu；context menu
Requirement：The＂GlobalTextList＂is open and active．A field is selected in the table．
For more information，see：${ }^{\leftrightarrows}$ Using a Text List

## Command：Check Visualization Text IDs

Symbol：
Function：The command checks whether or not the ID of a text list entry in the project is correct and reports the result．

Call：＂Text List＂menu；context menu
Requirement：The＂GlobalTextList＂is open and active．A field is selected in the table．
If the check determines that the global text list and the static texts of the visualizations do not match，then the reason for this may be that the global text list is or was write－protected．The requirement for this is that you have configured a user management in the project．
For more information，see：${ }^{\aleph}$ Using a Text List

## Command：Update Visualization Text Ids

## Symbol：㨼

Function：The command updates all inconsistent IDs in a static text list．
Call：＂Text List＂menu；context menu
Requirement：The＂GlobalTextList＂is open and active．A field is selected in the table．The object is write－protected．
If the check determines that the global text list and the static texts of the visualizations do not match，then the reason for this may be that the global text list is or was write－protected．The requirement for this is that you have configured a user management in the project．

For more information，see：${ }^{\Downarrow}$ Using a Text List

## Command：Remove text list support

Symbol：䓍
Function：The text list support from the selected enumeration object is removed．

Call: Context menu of an object of an enumeration with text list support (祭)
Text list support allows for the localization of the enumeration component identifier and the display of the symbolic component value in a text display of a visualization.

## Command: Add text list support

Symbol:
Function: A text list support is added to the selected DUT object of type Enumeration.
Call: Context menu of a standard DUT object of type Enumeration (
Text list support allows for the localization of the enumeration component identifier and the display of the symbolic component value in a text display of a visualization.

### 6.4.6.3.3 Dialogs

## Dialog: Properties - Image Pool

Function: The basic properties of the selected image pool are set here.
Call: "View $\rightarrow$ Properties" command of an "ImagePool" object; context menu of an "ImagePool" object

| "Download only <br> used images" | $\boxed{\checkmark}:$ Instead of loading all images from the image pool, only the images <br> which are actually used in the application are downloaded to the con- <br> troller. |
| :--- | :--- |
| "Download by <br> visualization" | $\boxed{\checkmark}:$ The image pool is downloaded with the visualization to the controller. |
| "Internal" | $\checkmark:$ The image pool is not provided in the "ToolBox" view. You cannot <br> drag these images to the visualization. |

Table 740:"Symbol Library Settings"

| "Mark library as |  |
| :--- | :--- |
| symbol library" | Identifies the image pool as a symbol library for use in a visualization <br> The symbol library gets the VisuSymbolLibrary = TRUE key as a file <br> property in the project information. The VisuElements library is automat- <br> ically inserted as a placeholder library in the "POUs" pool of the Library <br> Manager. <br> Requirement: A library project is open. <br> The symbol libraries installed in the repository under "Project Settings", <br> "Visualization" category, "Symbol Libraries" tab are displayed. |
| "Text list for <br> symbol <br> translation" | Select the text list from the list box which contains the translated texts for the <br> image pool. |

For more information, see: ${ }^{4}$ Using an Image Pool and $\left.{ }^{4}\right\rangle$ Using a Text List

## Dialog: Properties - Text List

Function: The basic properties of the selected text list are set here.
Call: "View $\rightarrow$ Properties" command of a "Text List" object; context menu of a "Text List" object

| "Download by <br> visualization" | $\boxed{\square}:$ The text list is downloaded with the visualization to the controller. |
| :--- | :--- |
| "Internal" | $\boxed{\square}:$ The text list can be used only in a library. It is not available in an <br> ordinary CODESYS project. |

For more information, see: ${ }^{\Downarrow}$ Using a Text List

### 6.5 Libraries and solutions

### 6.5.1 Information on libraries

AC500 V3 library handling with CODESYS V3 is different from the library handling with CODESYS V2. In AC500 V3 a coexistence of library versions in the library manager is possible and users can change a version within the project. How to install a new library and how to choose a library version in the project is described in the application note Add New Library Version To an AC500 V3 Project.

System libraries When upgrading Automation Builder or an existing project, new AC500 V2 system libraries are installed automatically. Older library versions will be removed as coexistence of a new library version and an older library version is not possible. Check the available library version in the Library Manager.

> Usually, when upgrading Automation Builder or an existing project, new AC500 V2 system libraries are installed automatically and older library versions are removed.

> As an exception, for the CANopen device CM598-CN both library versions are available in the Library Manager due to compatibility reasons. However, coexistence of a new library version and an older library version is not possible. In order to avoid compile errors remove the older library version.

Chapter 6.3.7 "Converting an AC500 V2 project to an AC500 V3 project" on page 1799

Part of the system libraries are the group of CMP libraries, i.e. the App library, IEC task library and the log library. How to use CMP libraries is described in an application example.

## Customer libra- Target change from AC500 V2 to AC500 V3 ries <br> After a target change from AC500 V2 to AC500 V3 the customer libraries have to be converted manually using the Library Converter ${ }^{*}>$ Chapter 6.3.1.3 "Later change-over of a target system" on page 1414.

Some Standard CODESYS libraries are automatically converted during the target change.

## Documentation

for libraries

- Description for the use of and information about selected libraries
- Reference for function blocks, functions, structures etc.

』 Chapter 6.5.14 "Reference, function blocks" on page 4086

### 6.5.2 Reference to CODESYS (V3)

Note that CODESYS V3 libraries are used.
Chapter 6.4.1.17.3 "Information for Library Developers" on page 2035

### 6.5.3 Library Manager functionality

### 6.5.3.1 General

The Library Manager contains descriptions of libraries and function blocks. In the Automation Builder the Library Manager is located under the node "Application".
这 PLC_PRG (PRG)
$\pm$ Task Configuration
The Library Manager offers a wide array of functionality for the user.

Use cases and how to handle the function blocks of a certain library is described in sample projects. After the respective library has been installed the corresponding sample projects are available in the default path AutomationBuilder Examples. If the default path is inaccessible, click "Help $\rightarrow$ Project examples" in the Automation Builder menu.

StringUtils With the help of the StringUtils library, strings in the AC500 PLC application can be handled library and modified. How to use the available functions is demonstrated in the application example StringUtils library.

### 6.5.3.2 Search for libraries and add libraries

In the Library Manager the search function allows you to quickly find any library, user defined library or function.

1. Select "Add Library".

| $\dagger$ Add Library | $\times$ Delete Library |  | fifl Library Repository (0) Icon legend... |  |
| :---: | :---: | :---: | :---: | :---: |
| Name |  |  | Namespace | Effective version |
| $\pm$ 3SLicens | e $=$ 3SLicense, 3.5 .1 | 4.0 (3S - Smart Software Solutions GmbH) | _3S_LICENSE | 3.5.14.0 |
| $\pm$ A AC500_ | DiagCpu = DiagCpu, | 1.2.0.4 (ABB) | AC500_DiagCpu | 1.2.0.4 |

$\Rightarrow$ The Add Library Window opens and a list of all available libararies is displayed.
Libraries in folder "ABB - AC500" are created by ABB and tested in combination with Automation Builder.

We recommend to use libraries of subfolder "Use Cases" for your project.
Libaries in subfolder "Intern" are necessary for internal procedures.
All 3S libraries distributed with Automation Builder are required by ABB libraries and have been tested in combination with AC500 and Automation Builder. Additional 3S libaries that are not distributed with Automation Builder can easily be added. There are no known major issues with using them, however, be aware that they are not tested by ABB.

2. Enter the name of the library or function you are searching for.
3. Choose the library you want to add

### 6.5.3.3 View embedded documentation of all libraries

In the Library Manager you can view embedded documentation of any ABB and 3S libraries.
Precondition: Library must be available in Library Manager.
ⓨ Chapter 6.5.3.2 "Search for libraries and add libraries" on page 3774

1. Select a library.
$\Rightarrow$ The contents of the library are shown.

2. From the contents select an object.
$\Rightarrow$ The corresponding documentation is opened.

|  | DiagTypes, 1.2.4.1 (ABB) | - |
| :---: | :---: | :---: |
|  | \% history |  |
|  | + Enums |  |
|  | $\pm$ Types |  |

Documentation
DiagTypes Library Documentation
Company: ABB
Title: DiagTypes
Version: 1.2.4.1
Categories: ABB - AC500|Use Cases|Diagnosis
Namespace: AC500_DiagTypes
Author:
Placeholder: AC500_DiagTypes

### 6.5.3.4 Access version history

The Library Manager allows you to access the version history of ABB libraries. The version history is not available for non ABB libraries.

1. Select a library.
$\Rightarrow$ The contents of the library are shown.

2. Select "history".
$\Rightarrow$ The version history is shown.


Inputs/Outputs Documentation
history (STRUCT)
TYPE history : STRUCT
Change history

| version | date | id | comment |
| :--- | :--- | :--- | :--- |
| 1.2 .0 .1 | $2019-$ | - | build system maintenance |
|  | $08-28$ |  |  |

### 6.5.3.5 Download missing libraries

The Library Manager allows you to automatically download missing 3 libraries from the project that are not available from the library repository.

1. Select "Download missing libraries".
$\Rightarrow$ The 'Download missing libraries' window is opened.
2. Select which libraries you want to download.

### 6.5.4 ACS/DCS drives libraries

### 6.5.4.1 Introduction

### 6.5.4.1.1 Scope of the document

The purpose of system technology document is to give an overview and explain the overall concepts of the Drives library in V3. The library contains function blocks to establish communication, to control the ABB ACS / DCS drives from AC500 V3 PLCs.

### 6.5.4.1.2 Safety instructions and preconditions to use drives library

The user has to read the following instructions and documents before using the libraries:

- All pertinent state, regional, and local safety regulations must be observed when installing and using this product. When functions or devices are used for applications with technical safety requirements, the relevant instructions must be followed.
- Read the complete safety instructions of the user's manuals for the devices you are using, before installation and commissioning.
- Read all safety instructions of the AC500 PLC. See System description AC500 in the online help in Automation Builder.
- Read the user information of the devices and functions you are using, see online help in Automation Builder.
- Installation and commissioning of the drive(s) is not part of this document nor the online help of Automation Builder. Installation and commissioning of the drive(s) must be done according to the related drives manuals and safety instructions.
The library package has been released for the software and firmware versions listed in the readme file of the package only.

In no event will ABB or its representatives be liable for loss of data, profits, revenue or consequential, incidental or other damage that may result from the use of other versions of product, software or firmware versions. The error-free operation of the Drives V3 Library with other devices, software or firmware versions should be possible but cannot be guaranteed and may need adaptations e.g. of example programs.
The user must follow all applicable safety instructions and the guidelines mentioned in the user documents of the ABB products.
Read the complete safety instructions for the AC500 before installation and commissioning.


## CAUTION!

Generally, the user in all applications is fully and alone responsible for checking all functions carefully, especially for safe and reliable operation.

The function blocks contained in the library can only be executed in RUN mode of the PLC, but not in simulation mode.

### 6.5.4.1.3 Comparison of V2 and V3 drives library

The below table compares the FBs in the V2 library package and corresponding adapted FBs in the V3 library package.

The V2 package (PS553-Drives) has different library files for each protocol and the same is replaced with a single library in V3 (ABB_Drives_AC500).

| PS553-Drives Library package (V2) |  | PS5605-Drives Library package (V3) |  |
| :---: | :---: | :---: | :---: |
| Library Name | Function Block | Library | Function Block |
| ACSDrives- <br> Base_AC500_V2 | $\begin{aligned} & \text { ACSSXXX_DRIVES_CTRL_ } \\ & \text { BASIC } \end{aligned}$ | Not supported - use DrvControlModbusACS |  |
|  | ACS_DRIVES_CTRL_EN G | ABB_Drives_AC500 | DrvControlModbusEng |
|  | ACS_DRIVES_CTRL_STA NDARD |  | DrvControlModbusACS |
|  | ACS_DRIVES_CTRL_STA NDARD GEN |  | DrvControlACS |
|  | MCS_MOD_READ_N_PR |  | DrvModbusRead |
|  | ACS_MOD_WRITE_N_PR M |  | DrvModbusWrite |
|  | ACS_REF_SCALING |  | DrvScaling |
| ACSDrivesCom- <br> ModRTU_AC500_V20 | ACS3XX_COM_MOD_RT | Not supported |  |
|  | ACS_COM_MOD_RTU | ABB_Drives_AC500 | DrvModbusRtu |
|  | $\begin{aligned} & \text { ACS_COM_MOD_RTU_E } \\ & \text { NHANCED } \end{aligned}$ |  | DrvModbusRtu |
|  | $\begin{aligned} & \text { ACS_COM_MOD_RTU_G } \\ & \text { EN } \end{aligned}$ | ABB_ModbusRtu_AC500 | ModRtuToken |
|  | ACS_COM_MOD_RTU_G EN_READ_N_PRM |  | ModRtuRead |
|  | ACS_COM_MOD_RTU_G EN_WRITE_N_PRM |  | ModRtuWrite |
|  |  |  | ModRtuReadWrite23 |
| ACSDrivesCom- <br> ModTCP_AC500_V22 | ACS_COM_MOD_TCP | ABB_Drives_AC500 | DrvModbusTcp |
|  | ACS_COM_MOD_TCP_E NHANCED |  | DrvModbusTcp |
| ACSDrivesComModTCP_Ext_AC500_V24 | ACS_COM_MOD_TCPx | ABB_Drives_AC500 | DrvModbusTcp |
|  | $\begin{aligned} & \text { ACS_COM_MOD_TCPx_E } \\ & \text { NHANCED } \end{aligned}$ |  | DrvModbusTcp |
| DCSDrives_AC500_V24 | DCS_DRIVES_CTRL | ABB_Drives_AC500 | DrvControlModbusDCS |
|  | $\begin{aligned} & \text { DCS_DRIVES_CTRL_GE } \\ & \mathrm{N} \end{aligned}$ |  | DrvControlDCS |
| ACSDrivesComPN_AC500_V24 | $\underset{\text { DPV1 }}{\substack{\text { ACS_PN_WRITE_N_PRM }}}$ | ABB_Drives_AC500 | DrvPnWrite |
|  | ACS_PN_READ_N_PRM_ DPV ${ }^{-1}$ |  | DrvPnRead |
| ACSDrivesComPB_AC500_V24 | $\begin{aligned} & \text { ACS_PB_READ_N_PRM_ } \\ & \mathrm{DPV}_{1} \end{aligned}$ | Will be supported in next Release |  |
|  | ACS_PB_WRITE_N_PRM _DPV1 |  |  |
|  | ACS_COM_PB | Not supported |  |
|  | ACS_COM_PB_PZD |  |  |
|  | $\begin{aligned} & \text { ACS_PB_READ_PRM_DP } \\ & \text { V0 } \end{aligned}$ |  |  |


| PS553-Drives Library package (V2) |  | PS5605-Drives Library package (V3) |  |
| :--- | :--- | :--- | :--- |
| Library Name | Function Block | Library | Function Block |
|  | ACS_PB_WRITE_PRM_D <br>  <br> PV0 |  |  |
|  |  | ABB_Drives_AC500 | DrvModbusReadWrite23 |
|  |  | ABB_Drives_AC500 | DrvModbusRtuBroadcast |
|  | ABB_Drives_AC500 | DrvControlCANCiA402 |  |

### 6.5.4.1.4 PLCs and drives: communication and control

## General

This document will briefly explain about communication settings between PLCs with drives, how to control the drives from PLC using the control function blocks.
Each input and output of the function blocks are explained in the integrated documentation in the library.

This library is released for the following products:

- AC500 V3 CPU
- ABB Drives:
- ACS380, ACS480, ACS580, ACH580, ACQ580, ACSM1, ACS880, DCS550, DCS800, DCS880. Other drives may still work, but are not tested.
- To use the control blocks the Communication Profile must be "ABB Drives Profile" or "ABB Drives Profile enhanced"
- Fieldbus Adapters: FENA-01, FENA-11, FENA-21, FSCA-01, FCAN-01, FECA-01, RETA-01, RETA-02, RCAN-01, FPNO-21, FMBT-21.
Fieldbus adapter support is dependent on the drive and for more details refer the corresponding drive manual.

Drives Library in V3 will support following protocols for the communication:

- Modbus TCP (onboard ETH1 and ETH2 ports)
- Modbus RTU (onboard COM1 port)
- PROFINET (using communication module CM579-PNIO)
- EtherCAT (using communication module CM579-ETHCAT)
- CANopen (onboard CAN port)


## Modbus TCP

## Preconditions

The following hardware components must be available:

- AC500 V3 PLC with ETH option. Configure onboard ETH1 or ETH2 for Modbus TCP.
- Drive with fieldbus adapter module
- ACS Drives and DCS880: FENA-01 or FENA-11 or FENA-21 or FMBT-21
- DCS550 and DCS800: RETA-01
- RJ45 Ethernet cable


## ACS drives



Fig. 274: FB - Overview of Modbus TCP connection with ACS drives

To exchange only status word, actual speed, control word and speed reference:

- Communication profile in drive parameters: ABB Drives classic
- Communication function block in AC500 program:

Use function block 'DrvModbusTcp'. \&y Chapter 6.5.4.4.1.8 "DrvModbusTcp" on page 3805

- DrvModbusTcp
- Control function block in AC500 program:

Use function block 'DrvControlModbusACS'. « Chapter 6.5.4.4.1.4 "DrvControIModbusACS" on page 3801

- Scaling of the speed or torque (optional):

Use function block ‘DrvScaling'. ${ }^{\text {B }}>\mathrm{C}$ Chapter 6.5.4.4.1.1 "DrvScaling" on page 3796

To exchange status word, actual value1 (speed), actual value2 (torque), control word, reference1 (speed), reference value2 (torque) and up to 12 more values read from drive and up to 12 more values write to the drive:

- Communication profile in drive parameters: ABB Drives enhanced
- Communication function block in AC500 program:

Use the function block 'DrvModbusTcp' with input EnhancedProfile = TRUE. $\Longleftrightarrow$ Chapter 6.5.4.4.1.8 "DrvModbusTcp" on page 3805

- DrvModbusTcp
- Control function block in AC500 program:

Use function block 'DrvControlModbusACS'. \& Chapter 6.5.4.4.1.4 "DrvControIModbusACS" on page 3801

- Scaling of the speed or torque (optional): Use function block 'DrvScaling'. $y^{\star y}$ Chapter 6.5.4.4.1.1 "DrvScaling" on page 3796

To exchange more than above mentioned values use additionally the following blocks:

- Read the values:
 on page 3804
- Write the values:

Use the function block 'DrvModbusWrite’. \& Chapter 6.5.4.4.1.7 "DrvModbusWrite" on page 3805

- Read Write the values:

Use the function block ‘DrvModbusReadWrite23'. « Chapter 6.5.4.4.1.11 "DrvModbusReadWrite23" on page 3825

DCS drives


Fig. 275: FB - Overview of Modbus TCP connection with DCS drives

DCS550 and DCS800 drives

To exchange only status word, actual speed, control word and speed reference:

- Communication function block in AC500 program:

Use function block "DrvModbusTcp". « Chapter 6.5.4.4.1.8 "DrvModbusTcp" on page 3805

- "DrvModbusTcp"
- Control function block in AC500 program:

Use function block "DrvControlModbusDCS". « Chapter 6.5.4.4.1.5 "DrvControlModbusDCS" on page 3803

- Scaling of the speed or torque (optional):

Use function block "DrvScaling". ${ }^{\wedge}>$ Chapter 6.5.4.4.1.1 "DrvScaling" on page 3796

To exchange more than above mentioned values use additionally the following blocks:

- Read the values:

Use the function block "DrvModbusRead". $\Rightarrow$ Chapter 6.5.4.4.1.6 "DrvModbusRead" on page 3804

- Write the values:

Use the function block "DrvModbusWrite". «̌ Chapter 6.5.4.4.1.7 "DrvModbusWrite" on page 3805

- Read write the values:

Use the function block "DrvModbusReadWrite23". « Chapter 6.5.4.4.1.11 "DrvModbusReadWrite23" on page 3825

DCS880 drives To exchange only status word, actual speed, control word and speed reference:

- Communication profile in drive parameters: ABB Drives classic
- Communication function block in AC500 program:

Use function block "DrvModbusTcp". $\Longleftrightarrow$ Chapter 6.5.4.4.1.8 "DrvModbusTcp" on page 3805

- "DrvModbusTcp"
- Control function block in AC500 program:

Use function block "DrvControlModbusDCS". « Chapter 6.5.4.4.1.5 "DrvControlModbusDCS" on page 3803

- Scaling of the speed or torque (optional):

Use function block "DrvScaling". « ${ }^{\Downarrow}$ Chapter 6.5.4.4.1.1 "DrvScaling" on page 3796

To exchange status word, actual value1 (speed), actual value2 (torque), control word, reference1 (speed), reference value2 (torque) and up to 12 more values read from drive and up to 12 more values write to the drive:

- Communication profile in drive parameters: ABB drives enhanced
- Communication function block in AC500 program:

Use the function block "DrvModbusTcp" with input EnhancedProfile = TRUE. ${ }^{\#}$ chapter 6.5.4.4.1.8 "DrvModbusTcp" on page 3805

- "DrvModbusTcp"
- Control function block in AC500 program:

Use function block "DrvControlModbusDCS". \& Chapter 6.5.4.4.1.5 "DrvControlModbusDCS" on page 3803

- Scaling of the speed or torque (optional): Use function block "DrvScaling". $\Leftrightarrow$ Chapter 6.5.4.4.1.1 "DrvScaling" on page 3796

To exchange more than above mentioned values use additionally the following blocks:

- Read the values:

Use the function block "DrvModbusRead". $\triangleq$ Chapter 6.5.4.4.1.6 "DrvModbusRead" on page 3804

- Write the values:

Use the function block "DrvModbusWrite". „ Chapter 6.5.4.4.1.7 "DrvModbusWrite" on page 3805

- Read Write the values:

Use the function block "DrvModbusReadWrite23". \& Chapter 6.5.4.4.1.11 "DrvModbusReadWrite23" on page 3825

## Modbus RTU

## Preconditions

The following hardware components must be available:

- AC500 V3 PLC. Configure onboard COM1 for the Modbus RTU communication.
- Drive with
- ACS Drives and DCS880: Embedded fieldbus or FSCA-01
- DCS550 and DCS800: Embedded fieldbus or RMBA-01
- Twisted pair serial cable


## ACS drives



Fig. 276: FB - Overview of Modbus RTU connection with ACS drives

To exchange only status word, actual speed, control word and speed reference:

- Communication profile in drive parameters: ABB Drives classic
- Communication function block in AC500 program:

Use function block "DrvModbusRtu". ${ }^{\leftrightarrows}$ Chapter 6.5.4.4.1.9 "DrvModbusRtu" on page 3812

- DrvModbusRtu
- Control function block in AC500 program:

Use function block "DrvControlModbusACS". 范 Chapter 6.5.4.4.1.4 "DrvControlModbusACS" on page 3801

- Scaling of the speed or torque (optional):

Use function block "DrvScaling". « Chapter 6.5.4.4.1.1 "DrvScaling" on page 3796

To exchange status word, actual value1 (speed), actual value2 (torque), control word, reference1 (speed), reference value2 (torque) and up to 12 more values read from drive and up to 12 more values write to the drive:

- Communication profile in drive parameters: ABB Drives enhanced
- Communication function block in AC500 program:

Use the function block "DrvModbusRtu". 乡 Chapter 6.5.4.4.1.9 "DrvModbusRtu" on page 3812

- "DrvModbusRtu"
- Control function block in AC500 program:

Use function block "DrvControlModbusACS". 乡 Chapter 6.5.4.4.1.4 "DrvControlModbusACS" on page 3801

- Scaling of the speed or torque (optional): Use function block "DrvScaling". $\triangleq$ Chapter 6.5.4.4.1.1 "DrvScaling" on page 3796

To exchange more than above mentioned values use additionally the following blocks:

- Read the values:

Use the function block "DrvModbusRead". $\triangleq$ Chapter 6.5.4.4.1.6 "DrvModbusRead" on page 3804

- Write the values:

Use the function block "DrvModbusWrite". « Chapter 6.5.4.4.1.7 "DrvModbusWrite" on page 3805

- Read Write the values:

Use the function block "DrvModbusReadWrite23". « Chapter 6.5.4.4.1.11 "DrvModbusReadWrite23" on page 3825

DCS drives


Fig. 277: FB - Overview of Modbus RTU connection with DCS drives

DCS550 and DCS800 drives

To exchange only status word，actual speed，control word and speed reference：
－Communication function block in AC500 program：
Use function block＂DrvModbusRtu＂．\＆Chapter 6．5．4．4．1．9＂DrvModbusRtu＂on page 3812
－＂DrvModbusRtu＂
－Control function block in AC500 program：
Use function block＂DrvControlModbusDCS＂．\＆Chapter 6．5．4．4．1．5＂DrvControlMod－ busDCS＂on page 3803
－Scaling of the speed or torque（optional）：
Use function block＂DrvScaling＂．侊 Chapter 6．5．4．4．1．1＂DrvScaling＂on page 3796

To exchange more than above mentioned values use additionally the following blocks：
－Read the values：
Use the function block＂DrvModbusRead＂．参 Chapter 6．5．4．4．1．6＂DrvModbusRead＂ on page 3804
－Write the values：
Use the function block＂DrvModbusWrite＂．$\gg$ Chapter 6．5．4．4．1．7＂DrvModbusWrite＂ on page 3805
－Read Write the values：
Use the function block＂DrvModbusReadWrite23＂．«y Chapter 6．5．4．4．1．11＂DrvModbus－ ReadWrite23＂on page 3825

DCS880 drives To exchange only status word，actual speed，control word and speed reference：
－Communication profile in drive parameters：ABB Drives classic
－Communication function block in AC500 program：
Use function block＂DrvModbusRtu＂．« Chapter 6．5．4．4．1．9＂DrvModbusRtu＂on page 3812
－＂DrvModbusRtu＂
－Control function block in AC500 program：
Use function block＂DrvControlModbusDCS＂．乡y Chapter 6．5．4．4．1．5＂DrvControlMod－ busDCS＂on page 3803
－Scaling of the speed or torque（optional）：
Use function block＂DrvScaling＂．«̌ Chapter 6．5．4．4．1．1＂DrvScaling＂on page 3796

To exchange status word，actual value1（speed），actual value2（torque），control word， reference1（speed），reference value2（torque）and up to 12 more values read from drive and up to 12 more values write to the drive：
－Communication profile in drive parameters：ABB Drives enhanced
－Communication function block in AC500 program：
Use the function block＂DrvModbusRtu＂．品 Chapter 6．5．4．4．1．9＂DrvModbusRtu＂ on page 3812
－＂DrvModbusRtu＂
－Control function block in AC500 program：
Use function block＂DrvControlModbusDCS＂．\＆Chapter 6．5．4．4．1．5＂DrvControlMod－ busDCS＂on page 3803
－Scaling of the speed or torque（optional）：Use function block＂DrvScaling＂．$\triangleq$ Chapter 6．5．4．4．1．1＂DrvScaling＂on page 3796

To exchange more than above mentioned values use additionally the following blocks:

- Read the values:

Use the function block "DrvModbusRead". $\Downarrow$ Chapter 6.5.4.4.1.6 "DrvModbusRead" on page 3804

- Write the values:

Use the function block "DrvModbusWrite". " Chapter 6.5.4.4.1.7 "DrvModbusWrite" on page 3805

- Read Write the values:

Use the function block "DrvModbusReadWrite23". « Chapter 6.5.4.4.1.11 "DrvModbusReadWrite23" on page 3825

## PROFINET

## Preconditions

The following hardware components must be available:

- AC500 V3 PLC with CM579-PNIO (PROFINET Master communication module)
- Drive with fieldbus adapter module
- ACS Drives and DCS880: FENA-01 or FENA-11 or FENA-21
- DCS550 and DCS800: RETA-02
- RJ45 Ethernet cable

The following values should be mapped in the fieldbus configuration of the drive and the configuration of AC500. These settings must be done in the Automation Builder hardware configuration.

- Drive $\rightarrow$ AC500: Status word and actual value 1 (speed) and optional actual value 2 (torque).
- AC500 $\rightarrow$ Drive: Control word and reference value 1 (speed) and optional reference value 2 (torque).

The following function blocks can be configured in the AC500 program.

- Communication profile: ABB Drives Profile
- Control block:
- ACS Drives: Use function block ‘DrvControIACS’. \& Chapter 6.5.4.4.1.2 "DrvControIACS" on page 3797.
- DCS Drives: Use function block ‘DrvControIDCS’. \& Chapter 6.5.4.4.1.3 "DrvControIDCS" on page 3799.
- Scaling of the speed or torque (optional): Use function block 'DrvScaling'. $y^{*}$ Chapter 6.5.4.4.1.1 "DrvScaling" on page 3796.
- PROFINET read function block. $\#$ Chapter 6.5.4.4.1.14 "DrvPNRead" on page 3829
- PROFINET write function block. 出 Chapter 6.5.4.4.1.15 "DrvPnWrite" on page 3830


## ACS drives



Fig. 278: FB - Overview of PROFINET connection with ACS drives

## DCS drives



Fig. 279: FB - Overview of PROFINET connection with DCS drives

## EtherCAT

## Preconditions

The following hardware components must be available:

- AC500 V3 PLC with CM579-ETHCAT (EtherCAT Master communication module)
- Drive with fieldbus adapter module
- ACS Drives and DCS880: FECA-01
- DCS550 and DCS800: RECA-01
- RJ45 Ethernet cable

The following values should be mapped in the fieldbus configuration of the drive and the configuration of AC500. These settings must be done in the Automation Builder hardware configuration.

- Drive $\rightarrow$ AC500: Status word and actual value 1 (speed) and optional actual value 2 (torque).
- AC500 $\rightarrow$ Drive: Control word and reference value 1 (speed) and optional reference value 2 (torque).


#### Abstract

A direct Ethernet cable from CM579-ETHCAT to FECA-01 module is recommended, connection through switch is not recommended since it will slow down the connectivity. Also, the drives need to be connected in the same sequence as they are added in the Automation Builder when multiple drives are connected.


The following function blocks can be configured in the AC500 program.

- Communication profile: ABB Drives Profile
- Control block:
- ACS Drives: Use function block "DrvControlACS". 出 Chapter 6.5.4.4.1.2 "DrvControIACS" on page 3797
- DCS Drives: Use function block "DrvControIDCS". « Chapter 6.5.4.4.1.3 "DrvControIDCS" on page 3799
- Scaling of the speed or torque (optional): Use function block "DrvScaling". $\Leftrightarrow$ Chapter 6.5.4.4.1.1 "DrvScaling" on page 3796


## ACS drives



Fig. 280: FB - Overview of EtherCAT connection with ACS drives

## DCS drives



Fig. 281: FB - Overview of EtherCAT connection with DCS drives

## CANopen

## Preconditions

The following hardware components must be available:

- AC500 V3 PLC. Configure onboard CAN port for CANopen communication.
- Drive with fieldbus adapter module
- ACS Drives and DCS880: FCAN-01
- DCS550 and DCS800: RCAN-01
- CANopen communication cable with $120 \Omega$ resistor.

The following values should be mapped in the fieldbus configuration of the drive and the configuration of AC500. These settings must be done in the Automation Builder hardware configuration.

- Drive $\rightarrow$ AC500: Status word and actual value 1 (speed) and optional actual value 2 (torque).
- AC500 $\rightarrow$ Drive: Control word and reference value 1 (speed) and optional reference value 2 (torque).

The following function blocks can be configured in the AC500 program.

- Communication profile: ABB Drives Profile
- Control block:
- ACS Drives: Use function block "DrvControIACS". « Chapter 6.5.4.4.1.2 "DrvControIACS" on page 3797
- DCS Drives: Use function block "DrvControIDCS". \& Chapter 6.5.4.4.1.3 "DrvControlDCS" on page 3799
- Scaling of the speed or torque (optional): Use function block "DrvScaling". $\triangleq$ Chapter 6.5.4.4.1.1 "DrvScaling" on page 3796


## ACS drives

AC500 program


Fig. 282: FB - Overview of CANopen connection with ACS drives

## DCS drives



Fig. 283: FB - Overview of CANopen connection with DCS drives

## CANopen with CAN CiA402 Profile for generic Drives

## Preconditions

The following hardware components must be available:

- AC500 V3 PLC. Configure onboard CAN port for CANopen communication.
- Any drive with CAN fieldbus adapter module and CAN CiA402 profile.
- CANopen communication cable with $120 \Omega$ resistor.

The following values should be mapped in the fieldbus configuration of the drive and the configuration of AC500. These settings must be done in the Automation Builder hardware configuration.

- Drive $\rightarrow$ AC500: Status word and actual speed.
- AC500 $\rightarrow$ Drive: Control word and reference speed.

The following function blocks can be configured in the AC500 program.

- Communication profile: CANopen device profile CiA402
- Control block: Use function block "DrvControICANCiA402". \#y Chapter 6.5.4.4.1.13 "DrvControICANCiA402" on page 3828


## General drives with CAN CiA402 interface



Fig. 284: FB - Overview of CANopen CiA402 with any drives

### 6.5.4.1.5 Compatibility

To check the compatibility of the drives and their communication modules please refer to the following table, it shows the tested combinations.

| Communication | PLC communication modules |  | Drive fieldbus adapter module |  |  | Drive |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | PLC com-munication module | Firmware version | FBA | FBA comm sw ver | FBA appl sw ver | Drive | Firmware version | Drive rating ID |
| Modbus RTU <br> Classic | Onboard |  | FSCA-01 |  | 1.63 | ACS580 | $\begin{aligned} & \hline 1.70 .4 .0 \\ & (\mathrm{CCON}-11) \end{aligned}$ | $\begin{aligned} & \text { ACS580-0 } \\ & 1-12 A 6-4 \end{aligned}$ |
|  |  |  | Embedded |  |  | ACS380 | 2.04.0.3 | $\begin{aligned} & \text { ACS380-0 } \\ & 4- \\ & X X-01 A 8-4 \end{aligned}$ |
|  |  |  | Embedded |  |  | ACS480 | 2.06.255.5 | $\begin{aligned} & \text { ACS480-0 } \\ & 4-02 A 7-4 \end{aligned}$ |
| Modbus RTU Enhanced | Onboard |  | FSCA-01 |  | 1.63 | ACS880 | 2.8.2 | $\begin{aligned} & \text { ACS880-0 } \\ & \text { 1-04A0-3 } \end{aligned}$ |
| Modbus TCP | Onboard ETH1 / ETH2 |  | RETA-01 | 1.30 | 3.05 | DCS800 | 3.7 |  |
| Modbus TCP Enhanced | Onboard ETH1 / ETH2 |  | FENA-21 |  | 3.20 | ACH580 | 2.06.0.2 | $\begin{aligned} & \text { ACH580-0 } \\ & 1-02 A 6-4 \end{aligned}$ |
| $\begin{aligned} & \text { PROFINE } \\ & \mathrm{T} \end{aligned}$ | CM579- <br> PNIO | 2.8.6.21 | FENA-21 |  | 3.20 | ACS880 | 2.82 | $\begin{aligned} & \text { ACS880-0 } \\ & 1-04 A 0-3 \end{aligned}$ |
|  |  |  | FENA-21 |  | 3.20 | ACSM1 | $\begin{aligned} & \hline \text { UMFI2000 } \\ & \text { (N2020) } \\ & \hline \end{aligned}$ | $\begin{aligned} & \text { ACSM1-03 } \\ & \text { A0-4 } \end{aligned}$ |
| EtherCAT | $\begin{aligned} & \text { CM579- } \\ & \text { ETHCAT } \end{aligned}$ | 4.4.3.21 | FECA-01 |  | 1.31 | ACQ580 | 2.05.0.4 | $\begin{aligned} & \text { ACQ580-0 } \\ & \text { 1-02A6-4 } \end{aligned}$ |
| CANopen (ABB Profile) | Onboard |  | FCAN-01 |  | 1.16 | ACSM1 | $\begin{aligned} & \text { UMFI2000 } \\ & \text { (N2020) } \end{aligned}$ | $\begin{aligned} & \text { ACSM1-03 } \\ & \text { A0-4 } \end{aligned}$ |
| CANopen (CiA402) | Onboard |  | FCAN-01 |  | 1.16 | ACS380 | 2.04.0.3 | $\begin{aligned} & \text { ACS380-0 } \\ & 4- \\ & \text { XX-01A8-4 } \end{aligned}$ |

### 6.5.4.2 Installation

The library is part of the Automation Builder 2.2. or higher. Use the Library manager to add the library into project.

For more details on the package, refer to the release notes of the latest Automation Builder.

### 6.5.4.3 Hardware and software requirement

| Hardware | Software |
| :--- | :--- |
| AC500 V3 PLCs: | Automation Builder 2.2. or higher |
| PM5630-2ETH, |  |
| PM5650-2ETH, |  |
| PM5670-2ETH and |  |
| PM5675-2ETH |  |
| AC500 V3 eCO: |  |
| PM5012-x-ETH |  |
| PM5032-x-ETH |  |
| PM5052-x-ETH and | Drive Composer Pro, |
| PM5072-T-2ETH | Drive Studio, |
| PM5082-T-2ETH *) | Drive Window or |
| ABB drive: | Drive Window Light |
| ACS380, ACS480, ACS580, ACH580, |  |
| ACQ580, |  |
| ACS880, ACSM1, DCS550, DCS800, |  |
| DCS880. |  |
| (Other drives may work, but are not tested.) |  |
| Fieldbus adapter module: |  |
| FSCA-01, RMBA-01 |  |
| FENA-01 / FENA-11 / FENA-21, RETA-01, |  |
| FPNO-21, FMBT-21 |  |
| FECA-01, RETA-02 |  |
| FCAN-01, RCAN-01 |  |
| (Other fieldbus adapter modules may work, |  |
| but are not tested.) |  |
| Remarks: |  |
| *) from $\geq$ AB 2.6.1 and CPU FW $\geq 3.6 .2 ~$ |  |

Before adding a drive to the device tree, the correct device description file has to be installed into Automation Builder's device repository. For ABB drives the corresponding device descriptions are available for download from our webside. For ABB drives connected via CANopen or EtherCAT there is also the option to generate the device descriptions via Drive Composer Pro ("Tools $\rightarrow$ EDS Export").

Drive configuration tool and fieldbus adapter module support is dependent on the drive used, for the compatible tool details refer to the drive manual.

### 6.5.4.4 Description of the library

### 6.5.4.4.1 Function blocks

## DrvScaling

| DrvScaling |  |
| :---: | :---: |
| Enable $B O O L$ | BOOL Busy |
| SpeedScaleMax UINVT | B001 Error |
| SpeedRefMax USMT | ERROR_D ErrorID |
| SpeedRef REAL | BOOL WarnAct |
| ActSpeedFb INT | WARNUMG_ID WarnIDLast |
| TorqueScaleMax UINT | INT SpeedRefFb |
| TorqueRefMax UINVT | REAL ActSpeed |
| TorqueRef REAL | SVT TorqueRefFb |
| ActTorquefb $\mathrm{INT}^{\text {I }}$ | REAL ActTorque |

Fig. 285: DrvScaling


DrvScaling function block is used to scale the speed or torque reference to the drive based on the maximum values defined.

Function block "DrvScaling" can be used to scale the variables from fieldbus equivalent values to values used in the program. Fieldbus variables are given in fieldbus equivalent values as INT values. With the scaling a conversion from INT (fieldbus) to REAL (program) and vice versa is performed. Reference1 and Actual Value1 (speed) are mostly given in the range of $-20000 \ldots+20000$. Reference2 and Actual Value2 (torque) are mostly given in the range of $0 \ldots+10000$.

## DrvControIACS

| Enable $B O O L$ | DrvControlACS |
| :--- | ---: |
| - |  |
| SwitchOn BOOL | BOOL Busy |
| Run $B O O L$ | $B O O L$ Error | -



This function block can be used to control ACS drives with ABB drives profile using direct input of status word (SW) from drive via any supported fieldbus communication like PROFINET, EtherCAT, CANopen.

Control word (CW) will be built by the function block according to the ABB drives profile state machine. Output CW has to be send to the drive via any fieldbus communication supported. Function block provides standard start/stop signals to control the drive and standard diagnosis signals are read from the drive.

| Drive Parameter | ACS380/ ACS480/ <br> ACS580/ ACH580/ <br> ACQ580/ ACS880 | ACSM1 | Comment |
| :--- | :--- | :--- | :--- |
| EXT1 COMMANDS | 20.01 = Fieldbus A | 10.01 = FBA | Fieldbus interface as <br> source for start and stop |
| EXT1 / EXT2 SEL | $19.11=$ MCW Bit11 (06.01) | $34.01=$ P02.12 bit 15 | Fieldbus interface as <br> source to switch to EXT2 <br> control place |
| REF1 SELECT | 22.11 = FBA ref1 | 24.01 = FBA ref1 | Fieldbus interface as <br> source to speed reference |
| FAULT RESET SELECT | $31.11=$ P06.01 bit 7 | $10.08=$ P02.12 bit 8 | Fieldbus interface as <br> source for fault reset |
| PROFILE | $51.02=$ Drives Classic / <br> Enhanced | $51.02=$ Drives Classic / <br> Enhanced | Control profile to ABB <br> Drives profile classic or <br> enhanced |

## DrvControlDCs




This function block can be used to control DCS drives with ABB drives profile using direct input of status word (SW) from drive via any supported fieldbus communication like PROFINET, EtherCAT, CANopen.

Control word (CW) will be built by the function block according to the ABB drives profile state machine. Output CW must be sent to the drive via any fieldbus communication supported. Function block provides standard start/stop signals to control the drive and standard diagnosis signals are read from the drive.

| Drive Parameter | DCS550 | DCS800 | DCS880 | Comment |
| :--- | :--- | :--- | :--- | :--- |
| EXT1 COMMANDS | 10.01 = Main Ctrl <br> Word | $10.01=$ Main Ctrl <br> Word | $20.01=$ Main Ctrl <br> Word | Fieldbus interface as <br> source for start and <br> stop |
| EXT1 / EXT2 SEL | 10.07 (HandAuto) <br> MCW: Bit11 <br> 11.02 (Ref1Mux) <br> MCW: Bit11 <br> 11.12 (Ref2Mux) <br> Invert 11.02 | 10.07 (HandAuto) <br> MCW: Bit11 <br> 11.02 (Ref1Mux) <br> MCW: Bit11 <br> $11.12 ~(R e f 2 M u x) ~$ <br> Invert 11.02 | $19.11=$ MCW Bit11 <br> $(06.01)$ | Fieldbus interface as <br> source to switch to <br> EXT2 control place |
| REF1 SELECT | $11.03=$ <br> SpeedRef2301 | $11.03=$ <br> SpeedRef2301 | $22.11=$ FBA ref1 | Fieldbus interface as <br> source to speed <br> reference |
| FAULT RESET <br> SELECT | NA | NA | Fieldbus interface as <br> source for fault reset |  |
| PROFILE | NA | NA | NA <br> Classic / Enhanced | Control profile to <br> ABB Drives profile <br> classic or enhanced |

## DrvControIModbusACS

|  | DrvControlModbusACS |
| :--- | ---: |
| Enable $B O O L$ |  |
| - SwitchOn $B O O L$ | $B O O L$ Busy | -



This function block can be used to control ACS drives with ABB Drives profile or ABB Drives enhanced profile using Modbus communication block like DrvModbusTcp or DrvModbusRtu.
Status Word (SW) is read from drive through Modbus communication block using "DriveData" interface. © Chapter 6.5.4.4.3 "Structure: DrvDataType" on page 3831

Control Word (CW) will be built by the function block according to the ABB drives profile state machine. CW will be sent via DriveData and the used communication block to the drive. Function block provides standard start/stop signals to control the drive and standard diagnosis signals are read from the drive.

The function block should be used for ACS drives using ABB drive (Classic/ Enhanced) profile for Modbus protocol only. The data transfer to the ACS drive is realized via the "IN_OUT" variable DriveData, which must be connected to "DrvModbusTcp" or "DrvModbusRtu" function block.

## DrvControIModbusDCS

| DrvControlModbusDCS |  |
| :---: | :---: |
| Enable BOOL | BOOL Busy |
| Switchon BOOL | BOOL Error |
| Run BOOL | ERROR_ID Errorid |
| StopEmcyCoast BOOL | BOOL WarnAct |
| -StopEmcyRamp BOOL | WARNING_ID WarnIDLast |
| Reset BOOL | BOOL NoConToDrive |
| Autostart BOOL | BOOL Ready |
| ControlWordB11 BOOL | BOOL Operating |
| ControlWordB12 BOOL | BOOL OnInhibited |
| ControlWordB13 BOOL | BOOL AtSetpoint |
| ControlWordB14 BOOL | BOOL Tripped |
| ControlWordB15 BOOL | BOOL Alarm |
| SpeedRef INT | BOOL LocalCtri |
| RefValue2 INT | INT ActSpeed |
| DriveData DrvDataType | INT ActValue2 |
|  | WORD ActStatusWord |
|  | WORD UsedControlWord |
|  | STRING(65) Message |



This function block can be used to control DCS drives with ABB Drives profile using Modbus communication block like "DrvModbusTcp" or "DrvModbusRtu".
Status Word (SW) is read from drive through Modbus communication block using DriveData interface. ${ }^{*}$ ) Chapter 6.5.4.4.3 "Structure: DrvDataType" on page 3831

Control Word (CW) will be built by the function block according to the ABB drives profile state machine. CW will be sent via DriveData and the used communication block to the drive. Function block provides standard start/stop signals to control the drive and standard diagnosis signals are read from the drive.

## The function block should be used for DCS drives using ABB drive profile for Modbus protocol only. The data transfer to the DCS drive is realized via the "IN_OUT" variable DriveData, which must be connected to "DrvModbusTcp" or "Dr̄ModbusRtu" function block.

## DrvModbusRead



The function block 'DrvModbusRead' reads one or more parameters / values of the drive. The number of data to be read is specified at the input 'Nvar'. The first parameter number is specified at the input 'PrmNum'. All parameters must be accessible from consecutive Modbus registers in the drive. The values of the parameters are stored in the PLC memory area, defined at the input 'Data'.
The values in the PLC memory area are updated when the read job was performed without error. This is indicated by JobDone $=$ TRUE and ModMastErrorAct $=$ FALSE.

If the Modbus job was finished with an error, the output ModMastErrorAct is set for one cycle. The Error ID returned by the Modbus job is shown at the output ModMastErrorIDLast. The output ModMastErrorIDLast will show that last Error ID until the input Enable is set from TRUE to FALSE.
As long as the Enable = TRUE a new read job is requested automatically one cycle after the further read job was terminated. The Modbus job is started from the Communication Block which is connected to the same 'DriveData' variable. It uses the Modbus function code 03 (read n words). The drive (Modbus device) from which the parameter is read is specified at this Communication Block. The Communication Blocks are available from the library e.g. DrvModbusTcp or DrvModbusRtu.
The function block is activated (Enable = TRUE) or deactivated (Enable = FALSE) via input Enable. If the block is active, the current values are available at the outputs. To start a new read job the input Enable must be set to TRUE. If the input values are valid, a request to perform a Modbus job is send to the Communication Block via the 'DriveData' variable. If at least 1 input is invalid, no job is generated, and the error is displayed at the outputs Error and ErrorID instead.

## DrvModbusWrite



Function block 'DrvModbusWrite' writes ' $n$ ' parameters to the drive. The number of parameters to be written must be available in the PLC memory area, defined at the input Data. The write job has been performed without error if JobDone $=$ TRUE and ModMastErrorAct $=$ FALSE .

If the Modbus job was finished with an error, the output ModMastErrorAct is set for one cycle. The Error ID returned by the Modbus job is shown at the output ModMastErrorIDLast. The output ModMastErrorIDLast will show that last Error ID until the input Execute is set from TRUE to FALSE.

To start a new write job the input Execute must be set from FALSE to TRUE (edge sensitive). The Modbus job is started from the Communication Block which is connected to the same DriveData variable. It uses the Modbus function code 16 (write n words). The drive (Modbus device) to which the parameter is written is specified at the Communication Block.

Drive parameters are only saved temporarily, if changed via fieldbus. To make these changes permanent in the drive the special parameter "PARAMETER SAVE" must be set.

Please see drive manuals for the parameter details.

## DrvModbusTcp

## Introduction

| DrvModbusTcp |  |
| :---: | :---: |
| Enable BOOL | BOOL Busy |
| Eth BYTE | BOOL Error |
| IPAdrServer STRING(15) | ERROR_ID Errorid |
| EnhancedProfile BOOL | BOOL ModMastErrorAct |
| NvarRead BYTE | WORD ModMastErrorIDLast |
| NvarWrite BYTE | BOOL WarnAct |
| WriteValues ARRAY [1..12] OF INT | WARNING_ID WarnIDLast |
| EnableWrite BOOL | BOOL JobDone |
| SteadyWrite BOOL | BOOL JobBusy |
| DriveData DrvDataType | ARRAY[1..12] OF INT Readvalues |

Function block DrvModbusTcp controls the Modbus TCP communication to ACS/DCS drives and provides the basic values (CW, Ref1, Ref2, SW, Act1, Act2) which are used for the basic control of drives with ABB Drives Profile or ABB Drives Enhanced Profile.

## ABB drives classic profile

With input parameter EnhancedProfile = FALSE, the function block works for ABB Drives Classic Profile.

Reading status The function block continuously reads data from the drive starting at Modbus register 400004. information from drives
(Actual Value 2) are continuously read from the drive and written to the DriveData variable.
These values are stored in DriveData.StatusWord, DriveData.ActValue1 and DriveData.ActValue2.

The following table shows the performed Modbus read job and the needed mapping in the drive as well as the area where the data is stored in the AC500.

| Modbus register address in drive | Mapping configuration in drive |  |  | Written in AC500 | Condition at function block |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  | ACS380, <br> ACS480, <br> ACS580, <br> ACH580, <br> ACQ580, <br> ACS880, ACSM1 | $\begin{array}{\|l} \text { DCS550, } \\ \text { DCS800 } \end{array}$ | DCS880 |  |  |
| Communication module | FENA-01/ 11/21 FMBT-21 | RETA-01 | FENA-01/11/21 FMBT-21 |  |  |
| 400004 | Status Word (SW) | Status Word (SW) | Status Word (SW) | DriveData.wStatusWord | Enable = TRUE |
| 400005 | Actual Value 1 | Actual Value 1 | Actual Value 1 | DriveData.iActValue1 | Enable = TRUE |
| 400006 | Actual Value 2 | Actual Value 2 | Actual Value 2 | DriveData.iActValue2 | Enable = TRUE |

Writing control To write the Control Word (CW), Reference Value 1 (Speed Reference) or
word and reference value to drives

Reference Value 2 (Reference Value 2) from the DriveData variable (DriveData.ControlWord, DriveData.Reference1, DriveData.Reference2) to the drive, the input EnableWrite has to be TRUE (default).

If the input SteadyWrite $=$ TRUE (default $=$ FALSE) these values are written steadily.
If the input SteadyWrite = FALSE (default) these values are only written if there was a change on any of those values.

These 3 values are written to the ACS drive starting at Modbus register 400001.
The function block checks if there are changes of the Control Word (wControlWord), Reference Value 1 (iRefValue1) or Reference Value 2 (iRefValue2) on the DriveData variable. If there is a change a write job is requested to send these 3 values to the ACS/DCS drive starting at Modbus register 400001.

The following table shows the performed Modbus write job and the needed mapping in the drive as well as the area where the data is stored in the AC500.

| Modbus register address in drive | Mapping configuration in drive |  |  | Written from AC500 | Condition at function block |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  | ACS380, ACS480, ACS580, ACH580, ACQ580, ACS880, ACSM1 | $\begin{aligned} & \text { DCS550, } \\ & \text { DCS800 } \end{aligned}$ | DCS880 |  |  |
| Communication module | $\begin{aligned} & \text { FENA-01/ 11/21 } \\ & \text { FMBT-21 } \end{aligned}$ | RETA-01 | FENA-01/11/21 FMBT-21 |  |  |
| 400001 | Control Word (CW) | Control Word (CW) | Control Word (CW) | DriveData.wControlWord | Enable = TRUE |
| 400002 | Reference Value 1 | Reference Value 1 | Reference Value 1 | DriveData.iRefValue1 | Enable = TRUE |
| 400003 | Reference Value 2 | Reference Value 2 | Reference Value 2 | DriveData.iRefValue2 | Enable = TRUE |

## ABB drives enhanced profile

With input parameter EnhancedProfile = TRUE, the function block works for ABB Drives Enhanced Profile.

Reading status information from drives

The function block continuously reads data from the drive starting at Modbus register 400051. So at least the Status Word (SW), Actual Value 1 (Speed Reference), Actual Value 2 (Actual Value 2) are continuously read from the drive and written to the DriveData variable.

These values are stored in DriveData.StatusWord, DriveData.ActValue1 and DriveData.ActValue2.

Apart from these three parameters there is also an option to read 12 additional drive parameters.
Using the input NvarRead the function block can be configured to read between 0 and 12 parameters from the drive. All read data is then written to the array at the ReadValue output array. Configuration in ACS drive is depending on configured parameters in group FBA DATA IN.

| Modbus register address in drive | Mapping configuration in drive |  |  | Written in AC500 | Condition at function block |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  | ACS380, <br> ACS480, <br> ACS580, <br> ACH580, <br> ACQ580, <br> ACS880, ACSM1 | $\begin{array}{\|l\|} \hline \text { DCS550, } \\ \text { DCS800 } \end{array}$ | DCS880 |  |  |
| Communication module | FENA-01/ 11/21 FMBT-21 | RETA-01 | FENA-01/11/21 <br> FMBT-21 |  |  |
| 400051 | Status Word (SW) | Status Word (SW) | Status Word (SW) | DriveData.wStatusWord | Enable = TRUE |
| 400052 | Actual Value 1 | Actual Value 1 | Actual Value 1 | DriveData.iActValue1 | Enable = TRUE |
| 400053 | Actual Value 2 | Actual Value 2 | Actual Value 2 | DriveData.iActValue2 | Enable = TRUE |
| 400054 | FBA Data IN 1 | FBA Data IN 1 | FBA Data IN 1 | ReadValues[1] | Enable = TRUE NVarRead >= 1 |
| 400055 | FBA Data IN 2 | FBA Data IN 2 | FBA Data IN 2 | ReadValues[2] | Enable = TRUE NVarRead >= 2 |
| ... | ... | ... | ... | ... | ... |
| 400064 | FBA Data IN 11 | FBA Data IN 11 | FBA Data IN 11 | ReadValues[11] | Enable = TRUE NVarRead >= 11 |
| 400065 | FBA Data IN 12 | FBA Data IN 12 | FBA Data IN 12 | ReadValues[12] | Enable = TRUE NVarRead >= 12 |

## If 32-bit parameters are mapped to DATA IN,

- The following field in DATA IN must be left open (= 0)
- The word order of the High-Word (HW) and Low-Word (LW) can be configured in the drive. (using FENA-X1: Par. 51.22)
- To retrieve the original 32-bit value from the drive in AC500 the HW and LW from ReadValues fields must be recombined in the program.

Function block DATA IN has to be configured in drive in the following groups see also FENA-x1 manual.

| Drive | Parameter Group |
| :--- | :--- |
| ACS355 | $54.01 \ldots 54.10$ |
| ACS380, ACS480, ACS580, ACH580, | $52.01 \ldots 52.1252 .01 \ldots 52.12$ if installed as <br> ACQ580, ACS880, ACSM1 |

Writing control word and reference value to drives

To write the Control Word (CW), Reference Value 1 (Speed Reference) or
Reference Value 2 (Reference Value 2) from the DriveData variable (DriveData.ControlWord, DriveData.Reference1, DriveData.Reference2) to the drive, the input EnableWrite has to be TRUE (default).

If the input SteadyWrite = TRUE (default = FALSE) these values are written steadily.
If the input SteadyWrite $=$ FALSE (default) these values are only written if there was a change on any of those values.

These 3 values are written to the ACS drive starting at Modbus register 400001.
Apart from these three there parameters there is also an option to write 12 additional drive parameters.
Using the input NvarWrite the function block can be configured to write between 0 and 12 parameters to the drive. The necessary values must be present in the array connected to WriteValues input. Configuration in ACS drive is depending on configured parameters in group FBA DATA OUT.

| Modbus register address in drive | Mapping configuration in drive |  |  | Written from AC500 | Condition at function block |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  | ACS380, ACS480, ACS580, ACH580, ACQ580, ACS880, ACSM1 | $\begin{array}{\|l\|l} \hline \text { DCS550, } \\ \text { DCS800 } \end{array}$ | DCS880 |  |  |
| Communication module | FENA-01/ 11/21 FMBT-21 | RETA-01 | FENA-01/11/21 <br> FMBT-21 |  |  |
| 400001 | Control Word (CW) | Control Word (CW) | Control Word (CW) | DriveData.wControlWord | Enable = TRUE |
| 400002 | Reference Value 1 | Reference Value 1 | Reference Value 1 | DriveData.iRefValue1 | Enable = TRUE |
| 400003 | Reference Value 2 | Reference Value 2 | Reference Value 2 | DriveData.iRefValue2 | Enable = TRUE |
| 400004 | FBA Data OUT 1 | FBA Data OUT 1 | FBA Data OUT 1 | WriteValues[1] | Enable = TRUE <br> NVarWrite >= 1 |
| 400005 | FBA Data OUT 2 | FBA Data OUT 2 | FBA Data OUT 2 | WriteValues[2] | Enable = TRUE NVarWrite >= 2 |
| ... | ... | ... | ... | ... | $\ldots$ |
| 400014 | FBA Data OUT 11 | FBA Data OUT 11 | FBA Data OUT 11 | WriteValues[11] | Enable = TRUE <br> NVarWrite >= 11 |
| 400015 | FBA Data OUT 12 | FBA Data OUT 12 | FBA Data OUT 12 | WriteValues[12] | Enable = TRUE <br> NVarWrite >= 12 |

If a Modbus TCP job tries to access a register in the drive which has no valid mapping information then job is aborted with an error.
Therefore, the drive parameters in FBA DATA IN group and FBA DATA OUT must be configured according to the used 'NvarRead' and 'NvarWrite' input number respectively.

## If 32-bit parameters are mapped to DATA OUT,

- The next/following field in DATA OUT must be left open (= 0)
- The word order of the High-Word (HW) and Low-Word (LW) can be configured in the drive. (using FENA-X1: Par. 51.22)
- To retrieve the original 32-bit value from the drive in AC500 the HW and LW from WriteValues fields must be recombined in the program.

ACS drive parameters are only saved temporarily, if changed via fieldbus.
To make these changes permanent in the drive the special parameter
"PARAMETER SAVE" must be set.
Please see also drive manuals which parameter must be set.
For ACS380, ACS480, ACS580, ACH580, ACQ580, ACS880 and DCS880 Par $96.07=1$

For ACSM1, DCS800 and DCS550 - Par 16.06 = 1

## Diagnosis

If a Modbus TCP job tries to access a register in the drive which has no valid mapping information the job is aborted with an error.
The output ModMastErrorAct reflects that an actual error occurred. This output is only TRUE for one cycle. At that cycle the output ModMastErrorIDLast reflects the actual ErrorID from the ModTcpMast job. The ModMastErrorIDLast will keep this Error ID until a new rising edge of the Enable is given.

However, there are internal diagnosis variables available, which are not shown at any output, but can be accessed from the function block instance.

These additional diagnosis variables can be accessed by opening the function block instance or through the block visualization "VisuDrvModbusTcp".

- iWriteErrCnt: number of errors in write jobs since Enable = TRUE.
- wLastWriteErno: holds the error number of the last executed write job.
- iReadErrCnt: number of errors in read jobs since Enable = TRUE.
- wLastReadErno: holds the error number of the last executed read job.
- iReadWriteErrCnt: number of errors in read write jobs since Enable = TRUE.
- wLastReadWriteErno: holds the error number of the last executed read write job.



## Drive parameter settings

| Settings in the drive <br> according to AC500 <br> configuration | ACS380, ACS480, <br> ACS580, ACH580, <br> ACQ580, ACS880, <br> ACSM1 | DCS550, <br> DCS800 | DCS880 |
| :--- | :--- | :--- | :--- |
| Communication module | FENA-01 /11/21 <br> FMBT-21 | RETA-01 | FENA- 01/11/21 <br> FMBT-21 |
| Fieldbus activation = <br> EXT FBA / ENABLE | 50.01 | 98.02 | 50.01 |
| FBA A Comm loss func | 50.02 | 30.35 | 50.02 |
| Comm Rate = Auto (0) | 51.03 | 51.02 | 51.03 |
| IP configuration = Static IP | 51.04 | 51.03 | 51.04 |
| IP address1 $\ldots$ IP <br> address2 | $51.05 \ldots 51.08$ | $51.04 \ldots 51.07$ | $51.05 \ldots 51.08$ |
| Subnet CIDR $=24$ <br> (eg: 255.255.255.0) | 51.09 | $51.08 \ldots 51.11$ | 51.09 |
| Gateway Address <br> (normally $=0.0 .0 .0)$ | $51.10 \ldots 51.13$ | $51.12 \ldots 51.15$ | $51.10 \ldots 51.13$ |
| Protocol / Profile $=$ <br> MB/TCP ABB E or <br> MB/TCP ABB C | 51.02 | 51.16 | 51.02 |
| Word order for <br> 32-bit parameter | 51.22 | No 32-bit access | 51.22 |
| Modbus timeout. <br> Depending on timeout <br> mode. Value in 100 ms | 51.20 | 51.17 | 51.20 |
| Modbus timeout mode: <br> If input "SteadyWrite" is <br> false set to "Any message" | 51.21 | 51.27 | 51.21 |
| If input "SteadyWrite" is <br> true can also be set to <br> "Control RW" |  |  | 51.27 |
| Refresh settings in drive | 51.27 |  |  |

- Please refer the respective drive / fieldbus module manual for the parameter settings if the drive setting is not mentioned in above table.
- For RETA-01/-02 IP address could also be set via hardware Dip-Switches. If any switch is set (192.168.0.xxx) with $x x x=$ Dip-Switches setting
- ACS drive parameters are only saved temporarily, if changed via fieldbus.

To make these changes permanent in the drive the special parameter "PARAMETER SAVE" must be set.
Please see also drive manuals which parameter must be set. For ACS380, ACS480, ACS580, ACH580, ACQ580, ACS880 and DCS880

- Par $96.07=1$

For ACSM1, DCS800 and DCS550 - Par 16.06 = 1

## DrvModbusRtu

## Introduction



Function block DrvModbusRtu controls the Modbus RTU communication to ACS/DCS drives and is used for the basic control of drives with ABB Drives Profile or ABB Drives Enhanced Profile.

## ABB drives classic profile

With input parameter EnhancedProfile = FALSE, the function block works for ABB Drives Classic Profile.

Reading status information from drives

The function block continuously reads data from the drive starting at Modbus register 400004. So at least the Status Word (wStatusWord), Actual Value 1 (iActValue1), Actual Value 2 (iActValue2) are continuously read from the drive and written to the DriveData variable.

These values are stored in DriveData.wStatusWord, DriveData.iActValue1 and DriveData.iActValue2.

With input NvarRead the function block can be configured to read in the same job between 0 ... 24 data more from the drive. These additional data are written to the array at the 'ReadValues' output. These data must be configured in the drive and are only accessible if the embedded Modbus is used.

The following table shows the performed Modbus read job and the needed mapping in the drive as well as the area where the data is stored in the AC500.

| Modbus register address in drive | Mapping configuration in drive |  |  | Written in AC500 | Condition at function block |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  | ACS380, ACS480, ACS580, ACH580, ACQ580, ACS880, ACSM1 | $\begin{array}{\|l\|} \hline \text { DCS550, } \\ \text { DCS800 } \end{array}$ | ACS380, <br> ACS480, <br> ACS580, <br> ACH580, <br> ACQ580, <br> ACS880,DCS880 |  |  |
| Communication module | FSCA-01 | RMBA-01 | Embedded fieldbus |  |  |
| 400004 | Status Word (SW) | Status Word (SW) | Status Word (SW) <br> $58.104=4$ | DriveData.wStatusWord | Enable = TRUE |
| 400005 | Actual Value 1 | Actual Value 1 | Actual Value 1 $58.105=5$ | DriveData.iActValue1 | Enable $=$ TRUE |
| 400006 | Actual Value 2 | Actual Value 2 | Actual Value 2 <br> $58.106=6$ | DriveData.iActValue2 | Enable = TRUE |
| 400007 | - | - | 58.107 DATA I/O 7 | ReadValues[1] | Enable = TRUE |
| 400008 | - | - | 58.108 DATA I/O 8 | ReadValues[2] | Enable = TRUE |
| ... | ... | ... | ... | ... | ... |
| 400014 | - | - | $\begin{aligned} & \text { 58.114 DATA I/O } \\ & 14 \end{aligned}$ | ReadValues[8] | Enable = TRUE |
| ... | ... | ... | ... | $\ldots$ | ... |
| 400024 | - | - | 58.124 DATA I/O 24 | ReadValues[18] | Enable = TRUE |

More details on the limits for the data read and write is explained in Chapter
6.5.4.5 "Limits for the data read and write between AC500 and drives"
on page 3832. The value is dependent on the Drive used.

Writing control word and reference value to drives

The function block checks if there are changes of the Control Word (wControlWord), Reference Value 1 (iRefValue1) or Reference Value 2 (iRefValue2) on the DriveData variable. If there is a change a write job is requested to send these 3 values to the ACS/DCS drive starting at Modbus register 400001.
The following table shows the performed Modbus write job and the needed mapping in the drive as well as the area where the data is stored in the AC500.

| Modbus register address in drive | Mapping configuration in drive |  |  | Written from AC500 | Condition at function block |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  | ACS380, <br> ACS480, <br> ACS580, <br> ACH580, <br> ACQ580, <br> ACS880, <br> ACSM1DCS880 <br> DCS880 | $\begin{array}{\|l\|} \hline \text { DCS550, } \\ \text { DCS800 } \end{array}$ | ACS380, ACS480, ACS580, ACH580, ACQ580, ACS880DCS880 |  |  |
| Communication module | FSCA-01 | RMBA-01 | Embedded fieldbus |  |  |
| 400004 | Control Word (CW) | Control Word (CW) | Control Word (CW) $58.101=1$ | DriveData.wControlWord | Enable = TRUE |
| 400005 | Reference Value 1 | Reference Value 1 | $\begin{aligned} & \hline \begin{array}{l} \text { Reference Value } \\ 1 \\ 58.102=2 \end{array} \\ & \hline \end{aligned}$ | DriveData.iRefValue1 | Enable = TRUE |
| 400006 | Reference Value 2 | Reference Value 2 | Reference Value 2 $58.103=3$ | DriveData.iRefValue2 | Enable = TRUE |

> More details on the limits for the data read and write is explained in 4 Chapter 6.5.4.5 "Limits for the data read and write between AC500 and drives" on page 3832. The value is dependent on the Drive used.

## ABB drives enhanced profile

With input parameter EnhancedProfile = TRUE, the function block works for ABB Drives Enhanced Profile.

The ABB Drives Profile Enhanced communication profile provides register mapped access to the Control, Status, Reference and Actual Values of the ABB Drives Profile Enhanced. The mapping of the registers has been enhanced to allow additional writing of up to 12 control and reading of up to 12 additional status parameters in a single Modbus job.

Reading status information from drives

The function block continuously reads data from the drive starting at Modbus register 400051 So at least the Status Word (wStatusWord), Actual Value 1 (iActValue1), Actual Value 2 (iActValue2) are continuously read from the drive and written to the DriveData variable.

These values are stored in DriveData.wStatusWord, DriveData.iActValue1 and DriveData. iActValue2.

Apart from these three parameters there is also an option to read 12 additional drive parameters in the same job.
Using the input NvarRead the function block can be configured to read between 1 and 12 more parameters from the drive. All read data is then written to the array at the ReadValues output. Configuration in ACS drive is depending on configured parameters in group FBA DATA IN.
The following table shows the performed Modbus read job and the needed mapping in the drive as well as the area where the data is stored in the AC500.

| Modbus register address in drive | Mapping configuration in drive |  |  | Written in AC500 | Condition at function block |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  | ACS380, ACS480, ACS580, ACH580, ACQ580, ACS880, ACSM1 | $\begin{array}{\|l} \text { DCS550, } \\ \text { DCS800 } \end{array}$ | DCS880 |  |  |
| Communication module | FSCA-01 | RMBA-01 | FSCA-01 |  |  |
| 400051 | Status Word (SW) | Status Word (SW) | Status Word (SW) | DriveData.wStatusWord | Enable = TRUE |
| 400052 | Actual Value 1 | Actual Value 1 | Actual Value 1 | DriveData.iActValue1 | Enable = TRUE |
| 400053 | Actual Value 2 | Actual Value 2 | Actual Value 2 | DriveData.iActValue2 | Enable = TRUE |
| 400054 | FBA Data IN 1 | FBA Data IN 1 | FBA Data IN 1 | ReadValues[1] | Enable = TRUE NVarRead >= 1 |
| 400055 | FBA Data IN 2 | FBA Data IN 2 | FBA Data IN 2 | ReadValues[2] | Enable = TRUE NVarRead >= 2 |
| $\ldots$ | ... | ... | ... | $\ldots$ | $\ldots$ |
| 400064 | FBA Data IN 11 | FBA Data IN 11 | FBA Data IN 11 | ReadValues[11] | Enable = TRUE NVarRead >= 11 |
| 400065 | FBA Data IN 12 | FBA Data IN 12 | FBA Data IN 12 | ReadValues[12] | Enable = TRUE NVarRead >= 12 |

Writing control word and reference value to drives

The function block checks if there are changes in any of the following values since last write job:

- Control Word (wControlWord),
- Reference Value 1 (iRefValue1),
- Reference Value 2 (iRefValue2) on the DriveData variable,
- values in the input array WriteValues - WriteValues[1..NvarWrite].

If there is a change a write job is requested to send the 3 control values and the values in WriteValues array (WriteValues[1..NvarWrite]) to the ACS/DCS drive starting at Modbus register 400001. Configuration in ACS drive is depending on configured parameters in group FBA DATA OUT.
The following table shows the performed Modbus write job and the needed mapping in the drive as well as the area where the data is stored in the AC500.

| Modbus register address in drive | Mapping configuration in drive |  |  | Written from AC500 | Condition at function block |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  | ACS380, <br> ACS480, <br> ACS580, <br> ACH580, <br> ACQ580, <br> ACS880, ACSM1 <br> DCS880 | $\begin{aligned} & \text { DCS550, } \\ & \text { DCS800 } \end{aligned}$ | DCS880 |  |  |
| Communication module | FSCA-01 | RMBA-01 | FSCA-01 |  |  |
| 400001 | Control Word (CW) | Control Word (CW) | Control Word (CW) | DriveData.wControlWord | Enable = TRUE |
| 400002 | Reference Value 1 | Reference Value 1 | Reference Value 1 | DriveData.iRefValue1 | Enable = TRUE |
| 400003 | Reference Value $2$ | Reference Value 2 | Reference Value 2 | DriveData.iRef Value2 | Enable = TRUE |
| 400004 | FBA Data OUT 1 | FBA Data OUT 1 | FBA Data OUT 1 | WriteValues[1] | Enable = TRUE NVarWrite >= 1 |
| 400005 | FBA Data OUT 2 | FBA Data OUT 2 | FBA Data OUT 2 | WriteValues[2] | Enable = TRUE NVarWrite >= 2 |
| ... | ... | ... | ... | ... | ... |
| 400014 | FBA Data OUT 11 | FBA Data OUT 11 | FBA Data OUT 11 | WriteValues[11] | Enable = TRUE NVarWrite >= 11 |
| 400015 | FBA Data OUT 12 | FBA Data OUT 12 | FBA Data OUT 12 | WriteValues[12] | Enable = TRUE NVarWrite >= 12 |

## Reconnection pause

When one or more drives in the Modbus RTU lines are offline, all the other drives have to wait for the TimeOut to elapse until a line token is assigned to next drive. Reconnection pause input helps in skipping the drives which are offline from the next Modbus job and execute Modbus job operations only for the drives which are online.
"ReconnectPause" is time in seconds before next retry to connect after a timeout was detected. Timeout is detected with ModMastErrorIDLast = 16\#120 (ERR_TIMEOUT).

This feature can be used with the DrvModbusRtu function block in both ABB Drives Profile and ABB Drives Enhanced Profile. User must configure the reconnect pause input value using the input variable "ReconnectPause".

For the generic RTU block ModRtuToken (part of AC500_ModbusRtu library), also the value for the reconnect pause must be configured at input variable "ReconnectPause".

## Diagnosis

The output ErrorID which reflects an actual error number is only valid for one cycle if output Error is set to TRUE. To capture this error number an external function must be programmed.

The output ModMastErrorAct reflects that an actual error occurred. This output is only TRUE for one cycle. At that cycle the output ModMastErrorIDLast reflects the actual ErrorID from the ModRtuMast job. The ModMastErrorIDLast will keep this error ID until a new rising edge of the Enable input is given.

However, there are internal diagnosis variables available, which are not shown at any output, but can be accessed from the function block instance.

These additional diagnosis variables can be accessed by opening the function block instance or through the block visualization "VisuDrvModbusRTU".

- iWriteErrCnt: number of errors in write jobs since Enable = TRUE.
- wLastWriteErno: holds the error number of the last executed write job.
- iReadErrCnt: number of errors in read jobs since Enable = TRUE.
- wLastReadErno: holds the error number of the last executed read job.
- iReadWriteErrCnt: number of errors in read write jobs since Enable = TRUE.
- wLastReadWriteErno: holds the error number of the last executed read write job.


If several drives are used, for each drive a communication function block such as DrvModbusRtu must be programmed. Also, every other generic Modbus server device on the same Modbus RTU line must be programmed with its own ModRtuToken function block. All those communication function blocks of one Modbus RTU line must be linked together via one variable of type ModRtuTokenType, connected to the InOut LineToken. Via this variable the Modbus token is passed to the next drive/device, so only one drive/device at a time is communicating with the PLC.
ModRtuToken function block and ModRtuTokenType structure are part of AC500_ModbusRtu library. Kindly refer the same.

If the user changes drive profile while drive is online with PLC, function block outputs may give wrong indication.

If a Modbus RTU job tries to access a register in the drive which has no valid mapping information then the job is aborted with an error.
Therefore, the drive parameters in FBA DATA IN group and FBA DATA OUT must be configured according to the used 'NvarRead' and 'NvarWrite' input number respectively.

## Modbus RTU using Embedded Fieldbus:

When embedded fieldbus is used for the Modbus RTU communication, user can read maximum of 24 parameters (based on the limitation in drive) from the DATA I/O parameters in the embedded fieldbus parameter group. These parameters can only be used for reading operation and cannot be configured to write data.

## Drive parameter settings

| Settings in the drive <br> according to AC500 <br> configuration | ACS380, ACS480, <br> ACS580, ACH580, <br> ACQ580, ACS880, <br> ACSM1, DCS880 | DCS550, <br> DCS800 | ACS380, ACS480, <br> ACS580, ACH580, |
| :--- | :--- | :--- | :--- |
| ACQ580, ACS880, |  |  |  |, | DCS880 |
| :--- |, | Embedded fieldbus |
| :--- |
| Communication module |
| FSCA-01 |

- Please refer the respective drive / fieldbus module manual for the parameter settings if the drive setting is not mentioned in above table.
- ACS drive parameters are only saved temporarily, if changed via fieldbus. To make these changes permanent in the drive the special parameter "PARAMETER SAVE" must be set.
Please see also drive manuals which parameter must be set.
For ACS380, ACS480, ACS580, ACH580, ACQ580, ACS880 and DCS880
- Par 96.07 = 1

For ACSM1, DCS800 and DCS550 - Par $16.06=1$


## DrvModbusRtuBroadcast

## Introduction



Function block DrvModbusRtuBroadcast is a communication block which sends the broadcast messages via the Modbus RTU communication to all ACS/DCS drives and other Modbus devices connected to the same Modbus RTU line (physical line). The function block can be used with all drives with either ABB Drives Profile or ABB Drives Enhanced Profile but not a mix of both profiles.
As the broadcast job will be received by all devices on the same physical Modbus line it's highly recommended to use this block only in case there are no other Modbus devices connected to this line and all drives use the same profile.
This function block does not perform any Modbus read operation, hence it does not read any values such as status word, actual value 1 and actual value 2 etc., from any of the drive.

This function block should not be used along with 'DrvModbusRead' and 'DrvModbusReadWrite23' function blocks. They will be ignored showing an error. This function block should be only used independently or in combination with 'DrvModbusWrite' function block for broadcasting write operation.
A successful broadcast message for writing control word, reference values and additional mapped parameters (only in case of Enhanced Profile) is indicated by JobDone = TRUE and ModMastErrorAct = FALSE. A next broadcast job for writing these values can once again started with a fresh rising edge at 'SendCtrIValues' input.

Apart from sending control values and up to 12 additional values from WriteValues array (only in case of ABB Drives Enhanced Profile) a normal Modbus write function block "DrvModbusWrite" can be used to send broadcast write messages to specific address on all drives connected to the Modbus RTU line. The requests to process broadcast write Modbus jobs is transferred via the DriveData structure at the InOut variable DriveData which can be connected to multiple instances of write function block 'DrvModbusWrite'.

After each successful broadcast write job a fixed pause of 250 ms is implemented before any other Modbus job within the same line will be started.

## ABB drives classic profile

With input parameter EnhancedProfile = FALSE, the function block works for ABB Drives Classic Profile.

Writing control word and reference value to drives

A rising edge from FALSE to TRUE at input 'SendCtrIValues' starts sending broadcast message with Control Word and Reference Values to all the drives starting at Modbus register 400001.
Following control values: Control Word (wControlWord), Reference Value 1 (iRefValue1) or Reference Value 2 (iRefValue2) are taken from DriveData variable for sending broadcast message.
The following table shows the performed Modbus write job and the needed mapping in the drive as well as the area where the data is stored in the AC500.

| Modbus register address in drive | Mapping configuration in drive |  |  | Written from AC500 | Condition at function block and input SendCtrlvalues |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  | ACS380, ACS480, ACS580, ACH580, ACQ580, ACS880, ACSM1DCS880 DCS880 | $\begin{aligned} & \text { DCS550, } \\ & \text { DCS800 } \end{aligned}$ | ACS380, <br> ACS480, <br> ACS580, <br> ACH580, <br> ACQ580, <br> ACS880DCS880 <br> DCS880 |  |  |
| Communication module | FSCA-01 | RMBA-01 | Embedded fieldbus |  |  |
| 400001 | Control Word (CW) | Control Word (CW) | Control Word (CW) $58.101=1$ | DriveData.wControlWord | ```Enable = TRUE and Rising Edge at SendCtrlvalues``` |
| 400002 | Reference Value 1 | Reference Value 1 | Reference Value 1 $58.102=2$ | DriveData.iRefValue1 | ```Enable = TRUE and Rising Edge at SendCtrlvalues``` |
| 400003 | Reference Value 2 | Reference Value 2 | Reference Value 2 $58.103=3$ | DriveData.iRefValue2 | ```Enable = TRUE and Rising Edge at SendCtrlvalues``` |

## ABB drives enhanced profile

With input parameter EnhancedProfile = TRUE, the function block works for ABB Drives Enhanced Profile.

With the ABB Drives Profile Enhanced profile, along with 3 control values Control Word, Reference Value 1, Reference Value 2, up to 12 additional values can be sent as broadcast message in a single Modbus job.

Writing control word and reference values to drives

A rising edge from FALSE to TRUE at input 'SendCtrIValues' starts sending broadcast message with Control Word and reference values to all the drives starting at Modbus register 400001.

Following control values: Control Word (wControlWord), Reference Value 1 (iRefValue1) or Reference Value 2 (iRefValue2) from DriveData along with values in the input array WriteValues - WriteValues[1..NvarWrite] are taken for sending broadcast message.

For the additional 12 values the configuration in ACS drive is depending on configured parameters in group FBA DATA OUT.

The following table shows the performed Modbus broadcast write job and the needed mapping in the drive as well as the area where the data is taken from the AC500.

| Modbus register address in drive | Mapping configuration in drive |  |  | Written from AC500 | Condition at function block |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  | ACS380, <br> ACS480, <br> ACS580, <br> ACH580, <br> ACQ580, <br> ACS880, ACSM1 <br> DCS880 | $\begin{aligned} & \text { DCS550, } \\ & \text { DCS800 } \end{aligned}$ | DCS880 |  |  |
| Communication module | FSCA-01 | RMBA-01 | FSCA-01 |  |  |
| 400001 | Control Word (CW) | Control Word (CW) | Control Word (CW) | DriveData.wControlWord | Enable = TRUE |
| 400002 | Reference Value 1 | Reference Value 1 | Reference Value 1 | DriveData.iRefValue1 | Enable = TRUE |
| 400003 | Reference Value $2$ | Reference Value 2 | Reference Value 2 | DriveData.iRef Value2 | Enable = TRUE |
| 400004 | FBA Data OUT 1 | FBA Data OUT 1 | FBA Data OUT 1 | WriteValues[1] | Enable = TRUE NVarWrite >= 1 |
| 400005 | FBA Data OUT 2 | FBA Data OUT 2 | FBA Data OUT 2 | WriteValues[2] | Enable = TRUE NVarWrite >= 2 |
| ... | ... | ... | ... | ... | ... |
| 400014 | FBA Data OUT 11 | FBA Data OUT 11 | FBA Data OUT 11 | WriteValues[11] | Enable = TRUE NVarWrite >= 11 |
| 400015 | FBA Data OUT 12 | FBA Data OUT 12 | FBA Data OUT 12 | WriteValues[12] | Enable = TRUE NVarWrite >= 12 |

## Diagnosis

The output ErrorID which reflects an actual error number is only valid for one cycle if output Error is set to TRUE. To capture this error number an external function must be programmed.

The output ModMastErrorAct reflects an actual error occurred in Modbus job. This output is only TRUE for one cycle. At that cycle the output ModMastErrorIDLast reflects the actual ErrorID from the ModRtuMast job. The ModMastErrorIDLast will keep this error ID until a new rising edge of the Enable input is given.

However, there are internal diagnosis variables available, which are not shown at any output, but can be accessed from the function block instance.

These additional diagnosis variables can be accessed by opening the function block instance or through the block visualization "VisuDrvModbusRTUBroadcast"

- iWriteErrCnt: number of errors in write jobs since Enable = TRUE.
- wLastWriteErno: holds the error number of the last executed write job.


For all drives, which are connected to same Modbus RTU line, one instance of broadcast block DrvModbusRtuBroadcast is enough and it must be connected to same LineToken of DrvModbusRtu function blocks which are used for communication between PLC and each drive on Modbus RTU line. All those communication function blocks of one Modbus RTU line must be linked together via one variable of type ModRtuTokenType, connected to the InOut LineToken. Via this variable the Modbus token is passed to the next drive / device, so only one drive / device at a time is communicating with the PLC.

All the drives should be configured either in Classic Profile or Enhanced Profile and accordingly the function block DrvModbusRtuBroadcast should be parameterized. Mix of profile with few drives in Classic and few drives in Enhanced should not be used when using DrvModbusRtuBroadcast block, if using such configuration along with DrvModbusRtuBroadcast may lead to incorrect operation.
If the user changes drive profile while drive is online with PLC, function block outputs may give wrong indication.

The Modbus RTU broadcast job is sent to all devices on the same physical Modbus RTU line.
Therefore, if other Modbus devices than ACS / DCS drives are connected to the same line using the ModRtuToken communication block it's highly recommended not to use the DrvModbusRtuBroadcast function block.
This might only be used, if the user is aware about the behavior of the connected devices if they receive the Modbus broadcast job.

If a Modbus RTU broadcast job is sent to access a register in the drive which has no valid mapping information then Modbus broadcast job is not aborted but will just send out the broadcast message without any error in the function block. This broadcast message is ignored by drives which have no valid mapping information.

Therefore, the drive parameters in FBA DATA OUT have to be configured according to the used 'NvarWrite' input number respectively.

## Drive parameter settings

| Settings in the drive according to AC500 configuration | ACS380, ACS480, ACS580, ACH580, ACQ580, ACS880, ACSM1, DCS880 | $\begin{aligned} & \hline \text { DCS550, } \\ & \text { DCS800 } \end{aligned}$ | ACS380, ACS480, ACS580, ACH580, ACQ580, ACS880, DCS880 |
| :---: | :---: | :---: | :---: |
| Communication module | FSCA-01 | RMBA-01 | Embedded fieldbus |
| Fieldbus activation = Modbus / RS-485 comm | 50.01 | 98.02 | 58.01 |
| FBA A Comm loss func | 50.02 | 30.35 | 50.02 |
| Slave number | 51.03 | 51.02 | 58.03 |
| Transmission rate | 51.04 | 51.03 | 58.04 |
| Parity | 51.05 | 51.05 | 58.05 |
| Protocol $/$ Profile $=$ ABB Classic / ABB Enhance | $\begin{aligned} & 51.02 \\ & \text { ed } \end{aligned}$ | 51.16 |  |
| Mapping of control word, Modbus reg 400001 | Fix | Fix | 58.101 |
| Mapping of reference value 1 , Modbus reg 400002 | Fix | Fix | 58.102 |
| Mapping of reference value 2 , Modbus reg 400003 | Fix | Fix | 58.103 |
| Mapping of status word, Modbus reg 400004 | Fix | Fix | 58.104 |
| Mapping of actual value 1, Modbus reg 400005 | Fix | Fix | 58.105 |
| Mapping of actual value 2, Modbus reg 400006 | Fix | Fix | 58.106 |
| Timeout mode $=$ None (0) or Any Message (1), but not Ctrl write (2) as these values are only written after changes | 51.07 |  | 58.15 |
| Modbus timeout. Depending on timeout mode. Value in 100 ms |  | 58.17 | 58.16 |
| Refresh settings in drive | 51.27 | 51.27 | 58.06 |

> - Please refer the respective drive / fieldbus module manual for the parameter settings if the drive setting is not mentioned in above table.
> - ACS drive parameters are only saved temporarily, if changed via fieldbus. To make these changes permanent in the drive the special parameter "PARAMETER SAVE" must be set.
> Please see also drive manuals which parameter must be set.
> For ACS380, ACS480, ACS580, ACH580, ACQ580, ACS880 and DCS880
> - Par 96.07 = 1
> For ACSM1, DCS800 and DCS550 - Par $16.06=1$

## DrvModbusReadWrite23

| DrvModbusReadWrite23 |  |
| :---: | :---: |
| Execute BOOL | BOOL Done |
| Prio BOOL | BOOL Busy |
| NvarRead UINT | BOOL Error |
| PrmNumRead UINT | ERROR_ID Errorid |
| DataRead DWORD | BOOL ModMastErrorAct |
| NvarWrite UINT | WORD ModMastErrorIDLast |
| PrmNumWrite UINT | BOOL WarnAct |
| DataWrite DWORD | WARNING_ID WarnIDLast |
| DriveData DrvDataType | BOOL JobDone |
|  | BOOL JobBusy |

The function block 'DrvModbusReadWrite23' reads and writes one or more parameters of the drive via DriveData connected to Modbus TCP / Modbus RTU communication blocks with Modbus function code FCT $=23$. This function block internally calls DrvModbusWrite to execute ReadWrite job with FCT $=23$, used along with the internal structure for Fct23, DrvModFct23Type.

The number of parameters to be read is specified at the input 'NvarRead'. The first address for read operation is specified at the input 'PrmNumRead'. The values of the data are stored in the PLC memory area, defined at the input 'DataRead'.

The number of parameters to be written is specified at the input 'NvarWrite'. The first address for write operation is specified at the input 'PrmNumWrite'. The values of the data that should be written must be stored in the PLC memory area, defined at the input 'DataWrite'.
To start a new ReadWrite job the input Execute must be set from FALSE to TRUE (edge sensitive). The Modbus job is started from the communication block DrvModbusTcp or DrvModbusRtu which is connected to the same DriveData variable. It uses the Modbus function code 23 (Read and write n words). The drive (Modbus device) to which the parameter is written is specified at the Communication Block.

The values in the PLC memory area are updated when the ReadWrite job was performed without error. The ReadWrite job has been performed without error if JobDone = TRUE and ModMastErrorAct $=$ FALSE .

If the Modbus job was finished with an error, the output ModMastErrorAct is set for one cycle. The Error ID returned by the Modbus job is shown at the output ModMastErrorIDLast. The output ModMastErrorIDLast will show that last Error ID until the input Execute is set from TRUE to FALSE.

After termination of this job, even if it was not successful, a next ReadWrite job can once again only be started with a rising edge at 'Execute' input.

Drive parameters are only saved temporarily, if changed via fieldbus.
To make these changes permanent in the drive the special parameter "PARAMETER SAVE" must be set.
Please see drive manuals for the parameter details.
For ACS380, ACS480, ACS580, ACH580, ACQ580, ACS880 and DCS880 -
Par $96.07=1$
For ACSM1, DCS800 and DCS550 - Par 16.06 = 1

## DrvControIModbusEng

| DrvControlModbusEng |  |
| :---: | :---: |
| Enable BOOL | BOOL Busy |
| Off1 BOOL | BOOL Error |
| Off2 BOOL | BOOL NoConToDrive |
| Off3 BOOL | BOOL ReadyOn |
| -InhibitOp BOOL | BOOL ReadyRun |
| RampOutZero BOOL | BOOL ReadyRef |
| RampHold BOOL | BOOL Tripped |
| RampInZero BOOL | BOOL Off2state |
| Reset BOOL | BOOL Off3State |
| ControlWordB8 BOOL | BOOL SwitchOnInhibit |
| ControlWordB9 BOOL | BOOL Alarm |
| RemoteCmd BOOL | BOOL AtSetpoint |
| ExtCtriloc BOOL | BOOL Remote |
| ControlWordB12 BOOL | BOOL AboveLimit |
| ControlWordB13 BOOL | BOOL ExtCtrilocact |
| ControlWordB14 BOOL | BOOL ExtRunEnable |
| ControlWordB15 BOOL | BOOL StatusWordB13 |
| UseControlWord BOOL | BOOL StatusWordB14 |
| ControlWord WORD | BOOL StatusWordB15 |
| RefValue1 INT | WORD ActStatusWord |
| RefValue2 INT | INT ActValue1 |
| DriveData DrvDataType | INT ActValue2 |
|  | WORD UsedControlWord |



The function block "DrvControlModbusEng" is designed for user specific control of the drive by setting the control word (CW) by the user itself in the program.
Therefore, the user should have a detailed knowledge of the ABB drives profile handling.
The reference and actual values must be given in fieldbus equivalent, e.g. range -20000 ... +20000.

Inputs "RefValue1", "RefValue2" and the generated control word are written to the "DriveData" variable which transfers these values to a communication function block, e.g. "DrvModbusRtu", "DrvModbusTcp" or "DrvModbusRtuBroadcast" communication function block writes to the drive. In the same way "ActValue1", "ActValue2" and the status word are transferred from the communication function block to the "DrvControlModbusEng" block, where they are written to the outputs.
The control word can be generated in 2 ways.
First way is to set the single bits of the control word separately at the inputs "Off1", "Off2" ... "ControlWordB15" while the input "UseControlWord" = FALSE.
Second way is to set the input "UseControlWord" = TRUE and write the control word as a whole word directly to the input control word. The generated control word is written to the "DriveData" variable and for diagnosis purpose also available at output "UsedControlWord".
The input and output names of the bits in control word and status word reflect the functions used with ABB Drive Profile. So the block should be used with ABB Drives Profil setting in the drive.

The function block does not execute any functionality expect data transfer to and from the "DriveData" variable. There is no special drive parameter setting necessary to use this block.
The programmer using this block should have a detailed understanding of how to set the control word according to the status word and the description of the used drive.
For standard speed and torque control application it is recommended to use the "DrvControlModbusACS" instead.

## DrvControlCANCiA402

| DrvControlCANCiA402 |  |
| :---: | :---: |
| Enable BOOL | BOOL Busy |
| -SwitchOn BOOL | BOOL Error |
| EnableOperation BOOL | ERROR_ID Errorid |
| - EnableVoltage BOOL | BOOL NoConToDrive |
| QuickStop BOOL | BOOL ReadyOn |
| ControlWordBit4 BOOL | BOOL SwitchedOn |
| ControlWordBit5 BOOL | BOOL OperationEnabled |
| ControlWordBit6 BOOL | BOOL Fault |
| Reset BOOL | BOOL VoltageEnabled |
| Halt BOOL | BOOL QuickStopped |
| ControlWordBit11 BOOL | BOOL OnDisabled |
| ControlWordBit12 BOOL | BOOL Warning |
| ControlWordBit13 BOOL | BOOL StatusWordBit8 |
| - ControlWordBit14 BOOL | BOOL Remote |
| - ControlWordBit15 BOOL | BOOL TargetReach |
| StatusWord WORD | BOOL IntLimitActivate |
|  | BOOL StatusWordBit12 |
|  | BOOL StatusWordBit13 |
|  | BOOL StatusWordBit14 |
|  | BOOL StatusWordBit15 |
|  | WORD ControlWord |
|  | STRING(80) Message |



The function block "DrvControlCANCiA402" is used for the control of ABB ACS Drive or nonABB drives from AC500 using CANCiA402 Profile. The CANopen CiA402 function block and visualization from the library can also be used for the 3rd party drives which comply to the CANopen CiA402 Profile.

## DrvPNRead



Function block "DrvPnRead" reads maximum 37 parameters from the drive in a single DPV1 query. The number of parameters to be read is specified at the input Nvar.
Parameters to read from the drive are specified at the Data input. "DrvPbPnPrmDpv1DataType" structure must be declared to a variable and connected to Data input using ADR. This structure contains the group, index, which must be given to the variable. Read parameter type and values are stored in the same variable.
"DrvPdPrmDpv1DataType" structure has the following array elements:

## ⓨ Chapter 6.5.4.4.4 "Structure: DrvPdPrmDpv1DataType" on page 3832

- "abyPrmGroup": Array of 37 WORD for specifying parameter group.
- "abyPrmIndex": Array of 37 WORD for specifying parameter index.
- "abyPrmType": Array of 37 DRV_PDRIVE_PRM_TYPE. READ parameter data type will be available here. For details refer to DRV_PDRIVE_PRM_TYPE. If a type is set here at the start, it can be compared with the type read from the drive if the compare input is TRUE.
- "adwPrmValue": Array of 37 DWORD. Read parameter value will be available here. If a value is set at the start, it can be compared with the value read from the drive if the compare input is TRUE.


## Read errors:

If the drive rejects to read a specific parameter, it returns the error code DRV_ERROR_PRM (16\#44) in the corresponding abyPrmType element and a more specific error value in the corresponding adwPrmValue element.

The number of elements with errors from the drive are given at the output "NumPrmErrors". The output "PrmErrCmpValues" gives an array, which contains the more specific error values in the elements (index). This can be used to quickly identify the erroneous elements.

## Compare input:

As the "DrvPnWrite" function block does not return an error in case a parameter in the drive could not be written correctly it is recommended to verify the writing.
This can be done with the call of this function block "DrvPnRead" if the same parameters, types and values are connected to the DATA input (use the same struct as for the writing), and the input "Compare" is set to TRUE.

Then the types and values of the connected struct are copied at the rising edge of execute inputs and compared with the returned types and values from the drive.

In case of a difference this is set into the corresponding element of the output array "PrmErrCmpValues" with the possible three error codes DRV_CMP_DIFF_TYPE, DRV_CMP_DIFF_VALUE or DRV_CMP_DIFF_TYPE_AND_VALUE

## Mode input:

- Mode $=16 \# 00=>$ Read direct variables and parameters via an Fxxx module, e.g. FENA-21 or FPNO-21. Group and Index have to be used as in the "Data.awPrmGroup" and "Data.awPrmIndex" array. (Number of Elements in the PN Data block is set to 16\#01)
- Mode $=16 \# 01 \Rightarrow>$ Read direct variables and parameters via an Rxxx module, e.g. RETA-21 for ACS800 or DCS500. Group and Index have to be used as in the "Data.awPrmGroup" and "Data.awPrmIndex" array. (Number of Elements in the PN Data block is set to 16\#01)
- Mode $=16 \# 1 \mathrm{x}=>$ to be used to access "PROFIDrive" parameters with Attribute $=16 \# 10$ (Value) and Number of Elements $=x$.
- Mode $=16 \# 2 x=>$ to be used to access "PROFIDrive" parameters with Attribute $=16 \# 20$ (Description) and Number of Elements = x. (Not supported with Fxxx or Rxxx modules)
- Mode $=16 \# 3 x=>$ to be used to access PROFIDrive parameters with Attribute $=16 \# 30$ (Text) and Number of Elements = x. (Not supported with Fxxx or Rxxx modules)
For "PROFIDrive" parameters using Mode $=16 \# 1 \mathrm{x}, 16 \# 2 \mathrm{x}$ or $16 \# 3 \mathrm{x}$ the Number of Elements $=$ $x$ is used for all the parameters in the Data array.


## DrvPnWrite

| DrvPnWrite |  |
| :---: | :---: |
| [Execute BOOL := FALSE] | BOOL Done |
| CmDevice AC500_IoDrvCM579Profinet.IDeviceCM579Profinet | BOOL Busy |
| DevName STRING(INT\#240) | BOOL Error |
| [DevSlot UIVTT : $=1$ ] | ERROR_ID ErrorID |
| [DevSubSlot UIVT := 1] | AC500_CM579Profinet.ERROR_ID AddErrorID1 |
| [Mode BYTE :=0] | UDINT AddErrorID2 |
| [MaxRetries BYTE :=4] | DWORD Status |
| [WaitTimeRetry WORD $:=200$ ] | WORD AddVal1 |
| Nvar BYTE | WORD AddVal2 |
| Data DWORD | UINT PackageSize |

Function block "DrvPnRead" reads maximum 37 parameters from the drive in a single DPV1 query. The number of parameters to be written is specified at the input Nvar.
Another limit while using the "DrvPnWrite" function block is, it can process only up to 240-byte data in one request or 37 drive parameters whichever is lower. If the write data length is more than 240 bytes, the function block generates an error code WRITE_PACKAGE_SIZE_TOO_LONG 16\#0004. At the output "PackageSize" the precalculated size of the request is shown.
Parameters to write to the drive are specified at the data input. "DrvPdPrmDpv1DataType" structure must be declared to a variable and connected to data input using ADR.
"DrvPdPrmDpv1DataType" structure has the following array elements:
② Chapter 6.5.4.4.4 "Structure: DrvPdPrmDpv1DataType" on page 3832

- "abyPrmGroup": Array of 37 WORD for specifying parameter group.
- "abyPrmIndex": Array of 37 WORD for specifying parameter index.
- "abyPrmType": Array of 37 DRV_PDRIVE_PRM_TYPE for specifying parameter type, refer the respective drives manual for parameter data type and enter the respective enumeration. For details about enumeration refer DRV_PDRIVE_PRM_TYPE in the library.
- "adwPrmValue": Array of 37 DWORD for specifying parameter value that should be written.

The values in the structure area are updated when the write job was performed without error. This is indicated by Done=TRUE.

## Mode input:

- Mode $=16 \# 00=>$ Write direct variables and parameters via an Fxxx module, e.g. FENA-21 or FPNO-21. Group and Index have to be used as in the "Data.awPrmGroup" and "Data.awPrmIndex" array. (Number of Elements in the PN Data block is set to 16\#01)
- Mode $=16 \# 01=>$ Write direct variables and parameters via an Rxxx module, e.g. RETA-21 for ACS800 or DCS500. Group and Index have to be used as in the "Data.awPrmGroup" and "Data.awPrmIndex" array. (Number of Elements in the PN Data block is set to 16\#01)
- Mode $=16 \# 1 \mathrm{x}=>$ to be used to access "PROFIDrive" parameters with Attribute $=16 \# 10$ (Value) and Number of Elements $=x$.
- Mode $=16 \# 2 x=>$ to be used to access "PROFIDrive" parameters with Attribute $=16 \# 20$ (Description) and Number of Elements $=x$. (Not supported with Fxxx or Rxxx modules)
- Mode $=16 \# 3 x=>$ to be used to access PROFIDrive parameters with Attribute $=16 \# 30$ (Text) and Number of Elements = x. (Not supported with Fxxx or Rxxx modules)

For "PROFIDrive" parameters using Mode $=16 \# 1 \mathrm{x}, 16 \# 2 \mathrm{x}$ or $16 \# 3 \mathrm{x}$ the Number of Elements $=$ x is used for ALL the parameters in the Data array.

### 6.5.4.4.2 Function: DrvModPara32Bit



Creates the Modbus address for 32-bit parameters of the ACSxxx drives.
To access 32-bit parameters in ACSxxx drives using Modbus a special address calculation must be performed.
This block calculates the 6-digit address out of the 5-digit address used for 16-bit parameters.
Input is the 5-digit address: GGii, where GG = parameter group and $\mathrm{ii}=$ the index.
E.g. Par $12.02 \Rightarrow$ address $=1202$.

Output is the calculated address for 32-bit parameters according to the following rule:
DrvModPara32Bit $=20000+(200$ * GG $)+(2$ * ii $)$ e.g. Par. $14.54 \Rightarrow$ output $=22908$
This output can be connected directly to the input "PrmNum" of one of the blocks DrvModbusRead or DrvModbusWrite or inputs "PrmNumRead" and "PrmNumWrite" of the block DrvModbusReadWrite23.

### 6.5.4.4.3 Structure: DrvDataType

Structure DrvDataType is used for the DriveData variable to exchange the data for one drive.
Structure DrvDataType is used for the DriveData variable which must be connected to all function blocks related to the same drive.

Besides the element "sName" all variables should not be written by the user directly. They are read and written within the function blocks. The DrvDataType contains some more internal, invisible variables which are used for internal functionality and not meant for user access.

The following table shows the visible variables of DrvDataType.

| Variable | Data Type | Default value | Description |
| :--- | :--- | :--- | :--- |
| wStatusWord | WORD | 0 | Actual status word from drive |
| iActValue1 | INT | 0 | Actual value1 from drive - mostly equal speed |
| iActValue2 | INT | 0 | Actual value2 from drive - mapping is made in drive <br> configuration |
| wControlWord | WORD | 0 | Control word to drive |
| iRefValue1 | INT | 0 | Reference value1 to drive - mostly speed reference |
| iRefValue2 | INT | 0 | Reference value2 to drive - mapping is made in drive <br> configuration |
| xOnline | BOOL | FALSE | Connection established - set in Modbus communication <br> function block after successful reading and writing one <br> Modbus job |
| xCtrlBlockUsedf | BOOL | FALSE | A control block is used to generate the control word, <br> ref1 and ref2 values |
| sName | STRING | 'Default <br> Drive Name' | Name for drive, which can be set by user directly to <br> DriveData variable |

### 6.5.4.4.4 Structure: DrvPdPrmDpv1DataType

Structure "DrvPdPrmDpv1DataType" is required to exchange the data between AC500 and drives using the PROFINET communication. "DrvPdPrmDpv1DataType" structure must be declared to a variable and connected to data input using ADR in the "DrvPnRead" or "DrvPnWrite" function blocks.

| Variable | Data Type | Default value | Description |
| :--- | :--- | :--- | :--- |
| awPrmGroup | ARRAY [1..37] <br> OF WORD | $[37(0)]$ | ABB drive Group number from where the parameter to <br> read or write - Profidrive Parameter Index. |
| awPrmIndex | ARRAY [1..37] <br> OF WORD | $[37(0)]$ | ABB drive index number from where the parameter to <br> read or write - Profidrive Parameter Subindex. |
| abyPrmType | ARRAY [1..37] <br> OFDRV_PDRI <br> VE_PRM_TYP <br> E | $[37($ DRV_EMP <br> TYPRM) $]$ | Parameter Type in array. While using read block it will <br> act as an output (and input ar start of block with Com- <br> pareTRUE). While using write block it will act as an <br> input. |
| adwPrmValue | ARRAY [1..37] <br> OR DWORD | $[37(0)]$ | Parameter Value in array. While using read block it will <br> act as an output (and input at start of block with Com- <br> pare=TRUE). While using write block it will act as an <br> input. |

### 6.5.4.5 Limits for the data read and write between AC500 and drives

The below table defines the limits for the reading of data from the drive and limits for writing data to drives from AC500 with cyclic data exchange.
If fieldbus adapter Plug (FBA) is used, then parameter group FBA DATA IN (e.g. 52) and group FBA DATA OUT (e.g. 53) is accessed in the drive. For the embedded fieldbus (EFB) parameters are used in EFB group (e.g. 58).
According to the table below, limits are defined for the variables 'NVarRead', 'NVarWrite' in DrvModbusRtu and DrvModbusTcp blocks.

| Drive | Fieldbus Adapter (FBA) |  | Embedded Fieldbus <br> (EFB) |
| :--- | :--- | :--- | :--- |
|  | Data In (Group 52) | Data Out (Group 53) | Data I/O (group 58) <br> Configurable as input <br> or output. |
| ACS380 | 12 | 12 | 14 |
| ACS480 | 12 | 12 | 14 |
| ACS580 | 12 | 12 | 14 |
| ACQ580 | 12 | 12 | 14 |
| ACH580 | 12 | 12 | 14 |
| ACS880 | 12 | 12 | 24 |
| ACSM1 | 12 | 12 | Not supported |
| DCS550 |  |  |  |
| DCS800 |  |  |  |
| DCS880 | 12 | 12 | 24 |

### 6.5.5 Application libraries

Application libraries can be used in AC500 V3 PLCs. The requirements for the use of the function blocks of the application libraries and information and prerequisites for the general handling of application libraries are described in the application examples:

HTTP library In order to be able to use the PLC as a client for web services, the HTTP function block library can be used. Setup and use of the HTTP library are described in the application example AC500 Webservices - HTTP Function Block Library.

MySQL library With the help of the MySQL function block library, MySQL databases can be used to store and access AC500 V3 data. Setup and use are described in an application example.

MSSQL library With the help of the MSSQL function block library, MSSQL databases can be used to store and access AC500 V3 data. Setup and use are described in an application example.

### 6.5.6 BACnet-BC

### 6.5.6.1 Introduction to BACnet

BACnet is a standardized data communication protocol for Building Automation and Control networks as defined in the ANSI/ASHRAE standard 135 and ISO 16484-5.

The advantage is interoperability between devices of different vendors.
The BACnet protocol defines services to allow communication between devices. Examples include 'Who is', 'I am', 'Who has' and 'I have' for device and object search and identification, "Read Property" and "Write Property" for the exchange of data, up to more complex services for alarm and event management, scheduling and trending.

The BACnet protocol defines a number of object types on which the services operate. Each object is characterized by its properties.

The BACnet objects are combined in a BACnet device. A BACnet device represents the functionality of a physical device.
More background information and introduction can be found here:
http://www.bacnet.org
http://www.bacnet.org/Bibliography

### 6.5.6.2 AC500 and BACnet

A BACnet device can be described by its "BACnet Interoperability Building Blocks" (BIBB)s, which are needed to establish services. They are grouped in different areas:

- "Data Sharing" (DS)
- "Alarm and Event Management"(AE)
- "Scheduling" (SCHED)
- "Trending" (T)
- "Device and Network Management" (DM)
"Data Sharing" for example contains two BIBBs which are needed for the "Service Read Property":
- Client side: DS-RP-A (Data Sharing - Read Property - A)
- Server side: DS-RP-B (Data Sharing - Read Property - B)

The BACnet standard defines profiles by the minimum required BIBBs, see table below. "BACnet Simple Sensor" (B-SS) is the simplest one, only containing one BIBB. More complex devices contain more BIBBs (from right to left).

|  |  | CP600 | AC500 V3 |  | AC500 V2 |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Interoperability Areas (IA) | BACnet Devicep-profiles |  |  |  |  |  |  |
|  | B-OWS | 'B-OD | B-BC | B-AAC | B-ASC | B-SA | B-SS |
| Data Sharing | DS-RP-A, B | \|DS-RP-A, B | DS-RP-A, B | DS-RP-B | DS-RP-B | DS-RP-B | DS-RP-B |
|  | DS-RPM-A | IDS-RPM-A, B | DS-RPM-A, B | DS-RPM-B |  |  |  |
|  | DS-WP-A | iDS-WP-A, B | DS-WP-A, B | DS-WP-B | DS-WP-B | DS-WP-B |  |
|  | DS-WPM-A | 1 | DS-WPM-B | DS-WPM-B |  |  |  |
|  |  | DS-V-A |  |  |  |  |  |
|  |  | ,DS-M-A |  |  |  |  |  |
|  <br> Event <br> Management | AE-N-A | , AE-N-A | AE-N-I-B | AE-N-I-B |  |  |  |
|  | AE-ACK-A | - | AE-ACK-B | AE-ACK-B |  |  |  |
|  | AE-INFO-B | 1 | AE-INFO-B | AE-INFO-B |  |  |  |
|  | AE-ESUM-A |  | AE-ESUM-B |  |  |  |  |
|  |  | 1 AE , VN-A |  |  |  |  |  |
| Scheduling | SCHED-A | 1 | SCHED-E-B | SCHED-E-B |  |  |  |
| Trending | T-VMT-A | 1 | T-VMT-I-B |  |  |  |  |
|  | T-ATR-A | 1 | T-ATR-B |  |  |  |  |
| Device \& Network Management | DM-DDB-A, B | \|'DM-DDB-A, B | DM-DDB-A, B | DM-DDB-B | DM-DDB-B |  |  |
|  | DM-DOB-A, B | 1DM-DOB-B $\quad 1$ | DM-DOB-B | DM-DOB-B | DM-DOB-B |  |  |
|  | DM-DCC-A | 1 | DM-DCC-B | DM-DCC-B | DM-DCC-B |  |  |
|  | DM-TS-A | 1-1\||r | DM-TS-B or DM-UTC-B | DM-TS-B or DM-UTC-B |  |  |  |
|  | DM-UTC-A | 1 |  |  |  |  |  |
|  | DM-RD-A | 1 | DM-RD-B | DM-RD-B |  |  |  |
|  | DM-BR-A | 11  <br> 1 1 <br> 1  | DM-BR-B |  |  |  |  |

The AC500 V2 supports BIBBs qualifying it as "BACnet Application Specific Controller" (BASC), by installing the BACnet B-ASC library.
AC500 V3 supports many more BIBBs qualifying it as "BACnet Building Controller" (B-BC), which contains a server (all BIBBs ending with -B) and a client (all BIBBs ending with -A). In fact, the AC500 contains some more BIBBs. All BIBBs under B-BC in the table above, plus:

| DS-COV-A, -B | (Change of Value-A, -B) |
| :--- | :--- |
| DS-COVP-A, -B | (Change of Value of Properties-A, -B) |
| AE-N-E-B | (Alarm and Event-Notification External-B) |
| AE-ASUM-B | (Alarm and Event-Alarm Summary-B) |
| SCHED-I-B | (Scheduling-Internal-B) |
| T-VMT-E-B | (Viewing and Modifying Trends External-B) |
| DM-TS-B | (Time Synchronization-B) |
| DM-UTC-B | (UTC Time Synchronization-B) |
| DM-MTS-A | (Manual Time Synchronization-A) |
| DM-LM-B | (List Manipulation-B) |
| DM-OCD-B | (Object Creation and Deletion-B) |
| NM-BBMDC-B | (BBMD Configuration-B) |

A list with all details can be found in the Automation Builder pdf document ABB-B-BC-PICSAC500_V3.pdf. Direction: Help/Project examples/Examples.
The figure below shows a typical application for an AC500 V3, acting as B-BC.


A drive with several actuators and sensors is acting as B-ASC, for example providing a temperature value as "Analog Input" (AI) object on the MS/TP network.
\& Chapter 6.5.6.3.2 "Supported BACnet networks" on page 3835
AC500 B-BC as client can read this temperature value, perform some processing (scaling, limit check) and on the server side provide the processed value as "Analog Value" (AV) object and as "Trend" object on the IP network. Higher level clients like BACnet Operator Workstation (B-OWS) can access the processed objects "Analog Value" and "Trend" for supervision.

The following chapters describe the possible applications and how to configure an AC500 V3 as B-BC.

### 6.5.6.3 AC500 V3 as BACnet Building Controller (B-BC)

### 6.5.6.3.1 General

The BACnet integration into CODESYS implements the ANSI/ASHRAE standard 135-2012 (ISO 16484-5) protocol revision 14 and is based on the AMEV AS-A and AS-B standards. Integration allows access to the properties of BACnet objects and the configuration parameters of a BACnet device by means of an IEC application. You can program a dynamic BACnet configuration and have access to the BACnet functions in the BACnet network by reading and writing BACnet object properties.

### 6.5.6.3.2 Supported BACnet networks

BACnet can run on different local area network types. The AC500 B-BC supports the following ones:

- MS/TP (Master Slave / Token Passing), based on serial RS-485
- BACnet IP, based on Ethernet / UDP / IP


Different networks can be combined to one common "BACnet internetwork". The figure above shows an example of some BACnet devices in one "BACnet internetwork". Each device has a device ID (10 to 15) which must be unique on application level. Services on application level (e.g. read or write request) are working with these device IDs and need no addressing information of the lower levels.

The example "BACnet internetwork" consists of different BACnet networks:

- BACnet MS/TP network connecting device 10, 11 and 12
- BACnet IP network (UDP port 47808), consisting of one IP subnets with IP range 192.168.0.x, connecting device 12, 13 and 14
- BACnet IP network (UDP port 47809), consisting of one IP subnet with IP range 192.168.2.x, connecting device 14 and 15

Addressing in a BACnet network is done through datalinks which must have a unique BACnet MAC address (which is different to an Ethernet MAC address).

- In a MS/TP network the BACnet MAC address is just one octet (1, 2,3 in the example). ${ }_{y}{ }^{\circ}$ Chapter 6.5.6.3.5.5 "Configuration of datalinks" on page 3844
- In an IP network the BACnet MAC address is the combination of the IP address and the UDP port number (for example 192.168.0.130.47808 for device 13). The following 16 UDP ports are reserved for BACnet: BAC0 (=47808 decimal) to BACF.
\# ${ }^{\sharp}$ Chapter 6.5.6.3.5.5 "Configuration of datalinks" on page 3844
To form a common "BACnet internetwork" the single BACnet networks must be combined by BACnet routers. AC500 can act as a BACnet router between BACnet MS/TP and IP networks (device 12 in the figure above) or between two different BACnet IP networks (device 14).
Two IP subnets using the same UDP ports can be combined to one BACnet IP network with an internet router.

|  | BACnet Internetwork |  |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Layer |  |  |  |  |  |  |  |  |  |
| Application | $\begin{gathered} \hline \text { Device } \\ \text { ID } 10 \\ \hline \end{gathered}$ | $\begin{aligned} & \hline \text { Device } \\ & \text { ID } 11 \end{aligned}$ | $\begin{gathered} \hline \text { Device } \\ \text { ID } 12 \\ \hline \end{gathered}$ |  |  |  | $\begin{aligned} & \text { Device } \\ & \text { ID } 14 \end{aligned}$ |  | $\begin{gathered} \hline \text { Device } \\ \text { ID } 15 \\ \hline \end{gathered}$ |
| Network |  |  | BACnet router |  | Internet Router |  | BACnet router |  |  |
|  | MSTP | MSTP | MSTP | IP | IP | IP | IP | IP | IP |
| Datalink | BACnet-mAC | BACnet-MAC | BACnet-MAC 3 | BACnet MAC UPD port .47808 | BACnet MAC UPD port .47808 | BACnet MAC UPD port .47808 | BACnet MAC UPD port .47808 | BACnet MAC UPD port .47809 | BACnet MAC UPD port .47809 |
| Physical | $\begin{gathered} \text { RS485 } \\ 19200 \text { baud } \end{gathered}$ | $\begin{gathered} \text { RS485 } \\ 19200 \text { baud } \\ \hline \end{gathered}$ | $\begin{gathered} \text { RS485 } \\ 19200 \text { baud } \end{gathered}$ | $\begin{gathered} \hline \text { +IP address } \\ .0 .120 \\ \hline \end{gathered}$ | +1P address .0 .130 | +IP address .1 .130 | +1P address <br> .1.140 | +IP address <br> .2.140 | +IP address <br> .2 .150 |
|  | BACnet network \#1 (MSTP) |  |  | IP subnet 1: 192.168.0.x |  | IP subnet 2: 192.168.1.x |  | IP subnet 2: 192.168.2.x |  |
|  |  |  |  | BACnet network \#2 (IP) |  |  |  | BACnet network \#3 (IP) |  |

The problem is that internet routers block local broadcast messages, which are required for BACnet communication. This can be solved by "Broadcast Management Devices" (BBDM). AC500 V3 can be configured as BBDM. In the figure above the devices 12 and 14 should be configured as BBDM in order to enable the BACnet communication across the internet router.
An alternative is to configure AC500 V3 as foreign BACnet device if an IP subnet contains no BBDM device to pass broadcast messages over internet routers.

Configuring the AC500 as BBDM or foreign device is described in ${ }^{\mu} \Rightarrow$ Chapter 6.5.6.3.5.5 "Configuration of datalinks " on page 3844.

### 6.5.6.3.3 Supported objects and properties

Communication with BACnet is done through objects and properties.
The AC500 B-BC server of the figure below is represented as a BACnet device object with "ID 12". The device contains more objects like the Analog Input object, representing the input of a temperature measurement device. An object contains several properties, like "ID, Description, Present Value, Unit" etc.

Further possible objects of an AC500 B-BC are:

- "Binary Input" for example from connected to a switch
- "Analog / Binary Output" for actuators
- "Analog / Binary Values" for local variables
- "Calender"
- "Schedule"
- "Trend Log"
- ...
- A list with all details can be found in the Automation Builder pdf document ABB-B-BC-PICSAC500_V3.pdf. Help/Project examples/.


Fig. 286: BACnet objects, properties, services and BIBBs

### 6.5.6.3.4 Supported BIBBs and services

While objects and properties describe which data are communicated, the communication itself is done with services between clients and servers. A certain service can only be executed if client and server have the related BIBBs. The Fig. 286 BACnet objects, properties, services and BIBBs shows a simple "Service Read Property" which is possible because the client on the right supports DS-RP-A and the server on the left supports DS-RP-B. The service is executed in two steps:

1. The client initiates a confirmed request "Read Property", asking for the present value of the "Analog Input" of object with "ID 1010".
2. The server answers with an acknowledge, sending the present value which is $21,89^{\circ} \mathrm{C}$ in the example.

A list of all supported BIBBs and services of AC500 V3 is given in the Automation Builder pdf document ABB-B-BC-PICS-AC500_V3.pdf. Help/Project examples/Examples.

## 6．5．6．3．5 BACnet configuration in Automation Builder

## General

To act as a BACnet server or client，the AC500 must be configured accordingly．The figure below shows the basic configuration of a BACnet server（left）and a BACnet server with client functionality（right）．It is also possible to have server and client functionality in parallel．

```
-17) Device_12 (PMS650-2ETH - TB5620-2ETH)
    * 副 PLClogic
        道 Io_8us
    - 目っ Interfaces
```

 Following objects need to be created：
1 ＂BACnet Server＂root object．This is the root object for the server functionality，as well as for the client functionality．It is mandatory，even if only client functionality is required．$\Leftrightarrow$ Chapter 6．5．6．3．5．2＂Configuration of BACnet server root object＂on page 3838
2 BACnet server objects，for example＂BACnet Analog Input＂Temperature．The properties of the objects must be controlled（written or read）by the PLC logic．${ }^{\wedge} \Rightarrow$ Chapter 6．5．6．3．5．3 ＂Adding BACnet server objects＂on page 3840
3 BACnet client objects，represented by a different symbol．For example，＂BACnet Client Read Property＂．The functionality of the client objects must be programmed in the PLC logic． Inserting the client objects below the server is optional．It is also possible to instantiate the objects only in a PLC logic．${ }^{*}>$ Chapter 6．5．6．3．5．4＂Adding BACnet client functionality＂ on page 3841
4 Datalink for the physical layer．This object links the physical interface（Ethernet IP or serial MS／TP）to the＂BACnet Server＂object．In the example above the IP address of ETH1 is automatically retrieved by inserting the＂BACnet IP datalink＂below the ETH1 port．\＆＂Con－ figuration of an IP datalink＂on page 3845．For MS／TP refer to ${ }^{*}$＂Configuration of an MS／TP datalink＂on page 3844.

## Configuration of BACnet server root object

1．Create an empty project with an AC500 V3 CPU type and call it fpr example＂Device＿12＂．
2．Insert a＂BACnet Server＂object below the interfaces object in the device tree．

| Devices | Add object below ：Interfaces |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  |  |
|  | Object path： |  |  |  |  |  |
|  | Device＿12IInterfaces |  |  |  |  |  |
|  | Object name：BACnet＿Server |  |  |  |  |  |
|  | Categories | $\checkmark$ | Search object name |  |  |  |
|  | $\begin{aligned} & \text { Scripting } \\ & \text { Uncategorized } \end{aligned}$ |  | Name | Short Description | Version | Order Number |
|  |  |  | BACnet Server |  | 1．5．2．1 |  |

3. Set the device InstanceNumber in the "BACnet Parameters" of the "BACnet Server", e.g. to 12 and the InstanceName to Device_12 (according to Fig. 286 BACnet objects, properties, services and BIBBs).

| Devices $\quad$ - $\quad \mathrm{x}$ | BACnet_Server $\times$ |  |  |  |
| :---: | :---: | :---: | :---: | :---: |
|  | BACnet Parameters | Parameter UsedBACnetStringType |  |  |
|  |  |  | Type | Value |
|  | BACnet IEC Obje |  | Enumeration of DINT | BACNET_STRING_U... |
|  | BACnet IEC Objects | - SupportUnsolicitedCOV | BoOL | FALSE |
|  | Status | - HideToOutside | BOOL | FALSE |
|  |  | - AutoStart | BOOL | true |
|  | Information | AutoGenerateBACstackConfig | BOOL | true |
|  |  | -- Device Properties |  |  |
|  |  | $\pm$ propertyAttributes |  |  |
|  |  | - InstanceNumber | DWORD | 12 |
|  |  | - InstanceName | WSTRING(255) | 'Device_12" |
|  |  | - SystemStatus | Enumeration of DINT | DEV_OPERATIONAL |
|  |  | - VendorName | WSTRING(255) | "ABB" |
|  |  | - Vendoridentifier | UDINT | 808 |
|  |  | - ModelName | WSTRING(255) | 'PM5650-2ETH" |
|  |  | - FirmwareRevision | WSTRING(255) | ${ }^{*}{ }^{*}$ |
|  |  | - ApplicationSoftwareversion | WSTRING(255) | "1.14.1.1.5.2.1" |

4. Add a datalink, IP or MS/TP. In the example an IP datalink is inserted below ETH1. Default parameters are sufficient if only one datalink is used.
« "Configuration of an IP datalink" on page 3845
Devices
5. Build the project, download to the PLC and set it to [RUN]. The status of the "BACnet Server" should be green (running). If not, please ensure that you have installed the runtime license BACnet Protocol B-BC Runtime, verifiable by right-click on the PLC node and select [Show license information] from the runtime licensing menu. The project is scanned for required licenses. If you are logged in to a PLC, then the licenses available on the PLC are displayed. A missing required license is highlighted.
«y Chapter 6.3.2.2.2 "PLC runtime licensing" on page 1446

6. Start any BACnet client to find the server, for example Inneasoft BACnet Explorer.


## Adding BACnet server objects

Goal is to publish an analog value as BACnet server object. This example is according to Fig. 286 BACnet objects, properties, services and BIBBs, left part containing a temperature value.

1. Configure a "BACnet Server" root object according to ${ }^{\wedge}>$ Chapter 6.5.6.3.5.2 "Configuration of BACnet server root object " on page 3838.
2. Add a "BACnet Analog Input" object below the "BACnet Server".

3. Rename it to Temperature, adjust the parameters: InstanceNumber: 1010, Description: Temperature, Units: UNIT_DEGREES_C.

4. The present value of the objects Temperature needs to be fed with the value from the real temperature device. Alternatively, a simple PLC program can simulate this value.

5. Download the program and observe the temperature value in the BACnet client.

| S Inneasoft BACnet Explorer (Free Edition) |  |  |  |
| :---: | :---: | :---: | :---: |
| File View ? |  |  |  |
|  |  |  |  |
| Favorites <br> Local network <br> Device_12 <br> © Alarms <br> - COV subscriptions <br> Analog Input <br> Temperature <br> Device | Id | Name | Value |
|  | 85 | Present Value | 25,36049 |
|  | 168 | Profile Name |  |
|  | 103 | Reliability | no-fault-detected (0) |
|  | 357 | Reliability Evaluation Inhibit | False |
|  | 106 | Resolution | 0,1 |
|  | 111 | Status Flags | $\{0 ; 0 ; 0 ; 0\}$ |
|  | 113 | Time Delay | 0 |
|  | 356 | Time Delay Normal | 0 |
|  | 117 | Units | ${ }^{\circ} \mathrm{C}(62)$ |
|  | 118 | Update Interval | 0 |
| Number of properties : 33 |  |  |  |

## Adding BACnet client functionality

Goal is to configure a second AC500 controller as BACnet client which reads an analog value from a server. This example is according to Fig. 286 BACnet objects, properties, services and BIBBs, right part.

1. Add a new controller and configure a "BACnet Server" root object according to ${ }_{\xi}{ }^{\circ}$ Chapter 6.5.6.3.5.2 "Configuration of BACnet server root object " on page 3838.
2. Set InstanceNumber to 14 and InstanceName to Device 14.

3. In addition to BACnet objects, BACnet clients can also be inserted as devices under a "BACnet Server". Add a "BACnet Client Read Property" below the "BACnet Server" node.

4. The created object "BACnet Client Read Property" generates a function block instance which can be used to program the client read functionality. The figure below shows a simple example.
In line 1-5 of the code part the function block is called with the following parameter:

- Device ID of the server to read from (12) $\stackrel{y}{y}$ Chapter 6.5.6.3.3 "Supported objects and properties " on page 3837
- Object ID of the object to read from (1010 for the "Analog Input")
- Object type ("Analog Input")
- Property to read ("present value")
- triggerRead to start the read operation

When the user (or another program part) sets the variable triggerRead from FALSE to TRUE the edge triggered function block BACnet_Client_Read_Property starts operation and sends the read request to the server device. After receiving the reply from the Server, the output . xDone gets TRUE (line 8) and the temperature value can be read from the output. result (line 14).

5. Download this program to another AC500 V3 controller, which is in the same IP network as the server. Set it to run and read the temperature value by setting triggerRead to TRUE. In online mode the read temperature value can be observed in line 14.


Alternative con- Unlike BACnet objects, a BACnet client does not require a complex (static) configuration, thus a figuration client function block can be used without creating a BACnet client as device.


There is no BACnet_Client_Read_Property object created below the "BACnet Server". Instead a function blōck BACnēt clīent Read Property must be declared in the PRG (line 6 in the declaration) and initially "connected" to its "BACnet Server" in IEC-code via RegisterToServer (), and thus get activated (line 2 in the code) $\left.{ }^{\circ}\right\rangle$ Chapter 6.5.14 "Reference, function blocks" on page 4086.


## Configuration of datalinks

For communication with other BACnet devices AC500 provides two different possibilities: MS/TP and IP.
${ }^{\wedge}$ Chapter 6.5.6.3.2 "Supported BACnet networks" on page 3835
For a non-routing device one MS/TP or IP datalink must be configured.
If more than one datalink is configured, routing between the datalinks is automatically enabled.

Configuration of • Add the "BACnet MS/TP COM" object below the COM port. an MS/TP datalink


In fact the empty COM port is replaced by the "BACnet MS/TP COM". By that the COM port is configured as RS-485 with fixed settings for MS/TP: No parity, 8 data bits, 1 stop bits.

- Below the "BACnet MS/TP COM" port object an "BACnet MS/TP datalink" is inserted automatically which can be configured according to the requirements.

| Devices $\quad \square \mathbf{~}$ | BACnet_MS_TP_datalink $\times$ |  |  |  |
| :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |
|  | BACnet MS/TP Parameters | Parameter | Type | Value |
|  |  | - DeviceName | STRING | '/dev/tty 00 ' |
|  | BACnet MS/IP IEC Objects | - NetworkNumber | WORD | 1 |
|  | Status | - ConnectionType | Enumeration of USINT | Master |
|  |  | - Baudrate | Enumeration of UDINT | 19200 |
|  | Information | - ReplyTimeout | UDINT | 250 |
|  |  | - TokenTimeout | UDINT | 20 |
|  |  | - DatalayerAdress | UDINT | 1 |
|  |  | - MessageQueuelength | WORD | 50 |

- NetworkNumber: Use the default value 1 if no routing is required. For routing, use a unique network number in one controller.
- ConnectionType: Use the default value Master if no routing is required. For routing, use "Master - answering always postponed".
- Baudrate can be set according to requirements in the range of from 9600 to $38400 \mathrm{bits} / \mathrm{s}$, higher values ( 57600 and 115200 bits/s) are not recommended.
- DatalayerAddress: This is the MAC address as described in $\Leftrightarrow$ Chapter 6.5.6.3.2 "Supported BACnet networks " on page 3835. The MAC address must be unique in the MS/TP network.
- For all other parameters the default values are recommended for typical applications.

Configuration of - Add a "BACnet_IP_datalink" object below the Ethernet port ETH1 or ETH2. an IP datalink


- NetworkNumber: Use the default value if no routing is required. For routing, use a unique network number in one controller.
- UPDport: Use the default value (47808 decimal) in the normal case. Range is possible from BAC0 (= 47808 decimal) to BACF. UDPport + IP address form the MAC address of the IP datalink as described in ${ }^{7}$ Chapter 6.5.6.3.2 "Supported BACnet networks " on page 3835. The IP address cannot be specified here. It is automatically taken from the parent Ethernet node (ETH1 or ETH2); its IP address is set in the communication settings of the CPU node, "Device_14" in the example.
- ForeignDevice and BBMD: Special configuration is only needed if an internet router is located between two BACnet devices.
«4 Chapter 6.5.6.3.2 "Supported BACnet networks" on page 3835
AC500 can be configured as ForeignDevice or BBMD, but not the combination of both. An example for BBDM can be found in the example folder.

Configuration of Routing enables the combination of different BACnet networks to one common "BACnet Routing internetwork".
ⓨ Chapter 6.5.6.3.2 "Supported BACnet networks" on page 3835
BACnet devices from different BACnet networks can communicate with each other.
If more than one datalink is configured in one CPU, routing between the different networks is automatically enabled. It must only be ensured that the network number is unique in one controller.

* Chapter 6.5.6.3.2 "Supported BACnet networks" on page 3835

For MS/TP the ConnectionType must be set to "Master - answering always postponed". An example for routing can be found in the example folder.

## Time syncronisation

The BACnet clients expect to receive the local time. Currently the AC500 V3 does not distinguish between UTC time and local time and its time zone is set to 0 . This will be improved in the near future. In the meantime, it is recommended to store the local time (green color in the following figure) in the AC500 as a workaround.


Using this workaround, the following time sync mechanisms can be used:

- Set local time from Automation Builder Tab "PLC Shell":

Set the time by the command "time hh:mm:ss"


- Read the local time from the Automation Builder Tab "Statistics":
"Current PLC Date and time" shows the PLC time as local time without conversion, if the tab "Show PLC time in UTC" is enabled.


For storing the local time in AC500, do not use the button [Set PLC to PC Time] (Tab "Statistics"), since this is always converting from local time to UTC time.

- BACnet clients can read local or UTC time, both requests will deliver the same (local) time information, since the timezone is 0 .
- If an SNTP time sync is required (for example with a Meinberg clock), UTC times are exchanged. For conversion of UTC to local time in AC500 a proprietary STNP client must be programmed.
Please contact the PLC support for more information.


### 6.5.6.3.6 Package content

## General

The BACnet package PS5607-BACnet-BC can be installed with the Installation Manager and contains the following components:

- BACnet runtime component, part of AC500 firmware.
- Automation Builder package: CODESYS BACnet
- BACnet plug-in component
- Device descriptions for "BACnet Server", BACnet objects, BACnet client and datalinks
- Libraries: BACnet, BACnetDefaultImpl and CmpBACnet.乡 Chapter 6.5.6.3.6.2 "BACnet libraries" on page 3847

Example folder - Example folder

- Examples and example documentation \# ${ }^{\text { }}$ Chapter 6.5.6.3.6.3 "Application examples" on page 3848
- Datasheet and FAQ BACnet Protocol Implementation Conformance Statement (PICS), acting as a datasheet, describing all BACnet objects, services and communication capabilities. BACnet Conformance Certificate FAQ - Frequently Asked Questions, including AC500 specific information, performance and limit


## BACnet libraries

The IEC library CmpBACnet represents the integration of the BACnet stack into a CODESYS IEC environment and provides the BACnet data types as well as the BACstack methods. The sole use of the IEC library CmpBACnet (without the BACnet and BACnetDefaultImpl libraries) would result in complex and lengthy IEC application code.

The BACnet library simplifies BACnet application development considerably as compared to the sole use of CmpBACnet, especially in the following areas:

- Starting and stopping the BACnet stack
- Using BACnet server objects and their properties
- Triggering asynchronous requests (mainly client service requests) and processing the request transaction
- Processing of callbacks from the BACnet stack (see IBACnetEventConsumer) and distributing the callbacks to multiple receivers in the application
Furthermore, the BACnet library provides a plug-in mechanism (BACnetServerPlugin) for extending certain aspects of the BACnet library. BACnetServerPlugin is the basis for the BACnetDefault Impl library.

The BACnetDefault Impl library is used for the additional simplification of BACnet application development. The BACnet standard ASHRAE 135 leaves some aspects of the practical use of BACnet open. The most notable examples include the following:

- Persistence of server objects
- Storage and persistence of Trend Log, Trend Log Multiple, and Event Log entries
- Update of the date/time information of the device object

The IEC library BACnet is intended as a layer over the IEC library CmpBACnet. However, the layer does not hide the library because this would require the BACnet library to have "facade" functions for CmpBACnet functions. These facade functions would result in larger application code and increased runtime requirements. This is difficult for the PLC to accept. For this reason, it is necessary to know when elements from the BACnet library or CmpBACnet library are to be used.

General rules:

- Starting and stopping the BACnet stack

Always use BACnetServer. StartBACnetStack and
BACnetServer.StopBACnetStack or AutoStart. Never directly use the corresponding
functions of the CmpBACnet library, such as CmpBACnet. BACnetServerInit.

- Using BACnet server objects and their properties

Always use the specified function blocks in IEC-lib-BACnet, such as BACnetAnalogValue. Never directly use the corresponding functions of the BACnet library, such as CmpBACnet. BACnetStorePropertyInstance.

- Triggering of asynchronous requests Always use the specified client function blocks of the BACnet library, such as BACnetClientReadProperty. Never directly use the corresponding functions of the CmpBACnet library, such as CmpBACnet. BACnetReadProperty. All functions of the CmpBACnet library that require a BACnetAsyncTransactionToken belong to this category and should never be used directly.
- Processing of callbacks from the BACnet stack and distributing the callbacks to multiple receivers in the application
Always use IBACnetEventConsumer and BACnetServer. RegisterHook/ UnregisterHook/RegisterCallback/UnregisterCallback. Never directly use the corresponding functions of the CmpBACnet library, such as CmpBACnet. BACnetSetHook or CmpBACnet. BACnetSetCallback.
When is it appropriate and safe to directly call the functions of the CmpBACnet library?
Basically, it is only necessary to call functions of CmpBACnet directly when a corresponding functionality is not provided in the BACnet library. Check the BACnet library first before trying to use CmpBACnet directly. It is possible to use blocking functions in CmpBACnet, such as BACnet*CbCompletion, BACnetIam(Ex), or BACnetIHave(Ex), BACnetUnconf*.

Most often, you will use BACnet*CbCompletion to implement your specific
IBACnetEventConsumer. BACnetEventCallbacks. But first check whether or not the BACnetDefaultImpl library already contains an appropriate standard implementation.

## Application examples

- AC500_V3_BACnet_B-BC_Example_ABxxx.project including simple read and write

- Use case 1: AC500 as BACnet client, read and write (with priority)
- Use case 2: AC500 as "BACnet Server", publish the analog value
- AC500_V3_BACnet_B-BC_Example_Routing_ABxxx.project
- Examples from 3S, including
- Read and write operations with more options, notification class, calendar, scheduler, etc.
- Device discovery
- BBMD
- Persistence
- Logging
- Routing


### 6.5.7 CAA library guidelines

With the help of the CAA library, different use cases for dealing with AC500 PLCs can be programmed. Function block descriptions for the CAA library can be found in the Library Manager.
The guidelines for the CAA libraries correspond to the general guidelines for library development. For a detailed description see help chapter Guidelines for Library Development.

By using the CSV read function, information from CSV or other formatted DAT files can be read into Structured Text. This use case is described in an application example.

Procedures how to handle files with the CAA library in order to write, read and append files are given in an application example.

### 6.5.8 Data Logger Library

### 6.5.8.1 Overview

The AC500 "Data Logger Library" package (PS5609-Log) contains the:

- "DataLogger" and "DataLoggerEco"library (licensed)
- "Multilogger" library (technology preview, documentation in the examples folder)

The data logging function block library ("DataLogger" and "DataLoggerEco") contains 5 function blocks for the purpose of advanced time-stamped data logging for different use cases.
In the most challenging use case, it also can be called buffering:
The AC500 application program generates or receives data which are normally transmitted to a telecontrol system for storage (historian databases) and further processing or displaying to the end user (SCADA, reporting). Typically, these may be remote applications like wateror oil-pumping or electrification stations or solar power plants. The connection between these remote stations with an AC500 and a central SCADA/telecontrol station is not always stable (mobile or wired internet) or only sporadically connected. Sporadically connected can be by intention, e.g. to save communication costs or open ports/connections to be used with a control station only in a limited way. Still owners or even authorities often require gapless data for compliance/prove of correct operation, billing or other purposes.

- Then the data logging function blocks buffer or store data in case of a broken or intentionally interrupted connection between AC500 and the telecontrol system and automatically retrieves and resends the data once online again and/or requested, while already newer data has to be buffered in parallel.


Fig. 287: Overview
1 AC500 application (remote substation)
2 Telecontrol (control station)

- The "Data Logger Library" can be also used as an event recorder. In this special mode data is continuously recorded in a ring buffer which can be read out after a certain event $x$ (e.g. outage) in order to analyze the values especially before but also after the event $x$.
OR
- Data can be logged only (and on command transferred to the FTP area to be analyzed offline or taken out via the memory card).

The following figure gives an overview of the described interaction of the data logging function blocks. There is always an input function block ("Logxx_Input") needed which transfers the input data into data sets with timestamp for use by the data logger ("LogHandling"). An output function block (Logxx_Output) receives the current or retrieved data from the data logger in case of communication or further processing. The input function blocks "Logxx_Input", the function block "LogHandling" and the output function blocks "Logxx_Output" communicate via SRAM FIFOin and FIFOout areas in the memory. The SRAM FIFOin is power-fail-safe intermediate buffer and help in decoupling time wise and speeding up the necessary write/read operations on the logging file structures significantly. These read/write operations on the files are done in blocks of data sets, enabling a comparably fast interaction with the otherwise slow file system.


Fig. 288: Overview function blocks
Each data logger application requires the main function block "LogHandling", one of the input function blocks to provide data to be logged and one output function block to retrieve the data and send the data to telecontrol (SCADA).
As input and output function blocks 2 different types exist:

- For data logging of an interrupted IEC 60870 communication, the function blocks "Loglec60870_Input" and "Loglec60870_Output" are provided. The IEC 60870 data logging function blocks support the IEC data types and work internally with the standard AC500 IEC 60870 library. The IEC data logging output function block does not need special handling or control/inputs.
- For other types of general data "LogGeneric_Input" and "LogGeneric_Output" are provided. The generic data logging function blocks support an even larger variety of data types. The generic output function block needs to be handshake with for each data set, in order to retrieve the data from the data logging files. Therefore the generic function blocks can also be used to integrate the data logging into any other protocol, e.g. Modbus.
The function block "LogHandling" ensures that also several consecutive and fast interruptions can be handled without losing data. While the log file is replayed, arriving new data is stored in the SRAM FIFOin and added to the data logging files in parallel (File FIFO) if the SRAM FIFOin becomes full (during that short time the log file replay is paused). Nevertheless any data send to a control station via a communication is always with the oldest data first (FIFO = "First In First Out").

As for normal Ethernet default setting it takes up to 30 seconds before a communication break is detected (e.g. with TCP/IP protocols the following has to be done):

- The data rate at which data might have to be buffered in case of a communication break has to be estimated and possibly minimized, especially the threshold option has to be checked carefully if used. A worst case test is advised with the application if in doubt.
- Ping mechanism should be used in the substation. This is done in the example program for the IEC logger. With this ping the interruption is detected much faster e.g. here after 1-2 seconds (can be configured in the example program - the configurable "SecureReadTime" must be considered in this context. This ensures that the time delay - before a loss of connection is detected and is compensated).

As the SRAM FIFO has to store data during this time its size limits the data rate. The SRAM FIFO size is 160 data sets (or 60 data sets on AC500-eCo). If data rate is too high, FIFO will overflow. The maximum data rate is depending on the CPU type, storage media (memory card/flash disk) and cycle time configured, and must be determined by trial and error if high.

The data rates for storing only without this detection can be much higher and depends on the CPU and memory type chosen. The data is always logged in directly readable csv format.

② Chapter 6.5.8.6 "CSV file formats" on page 3856
Depending on the input function block and data type, the log file contains only one or up to 32 data variables per timestamped data set. The data logging files can be configured (up to 65 k data sets per file, up to 999 consecutive log files, file name extension).

### 6.5.8.2 Operating modes

This chapter describes the different operating modes of the data logging and their behavior.

- Mode 0/1: Buffer and disposal in chronologic order.
- Mode 0: Limited storage (keeps oldest, but stops if full).
- Mode 1: Endless (ring buffer) operation modes (deletes oldest).
- Mode 2: Buffer and disposal via FTP, Log file(s) copied to FTP server area for further use.
- Mode 3: Events recorder, logs data before and after an event.

Mode 0/1: Buffer and disposal in chronologic order


Fig. 289: Overview Mode 0/1
Mode $0 / 1$ is for buffering the values from the AC500 application in case of a broken or intentionally interrupted connection between AC500 and telecontrol. In the normal state 1 the values are directly sent from the FIFOin (input values from application) to FIFOout (telecontrol connection). As soon as the connection is interrupted, the data logging changes to working state 2 . The values are sent to the file FIFO instead. When the file FIFO is full, the data logging is stopped (Mode 0) or the oldest data will be overwritten (Mode 1 = ringbuffer). When the connection is established again and the "ReleaseHistory" pin is triggered, the data logging changes to working state 3. It cares for disposal of the values in chronological order. The buffered values
are written to FIFOout (working state 3a). This may take some time during which new values are coming from the application and stored into FIFOin. Before the FIFOin overflows the data logging switches to working state 3b and buffers the new values. After that it can continue with working state 3a. Only if the File FIFO is empty (all files deleted) the data logging changes back to normal state 1.
The advantage of Mode $0 / 1$ is that all values (directly and buffered) are sent to telecontrol in strictly chronological order which is expected by most control stations (SCADA systems/historians).
If a historical value is sent to the SCADA after a current value has already been sent, the historical value is normally rejected by the SCADA.

As it takes up to 30 seconds before a communication break is detected (e.g. with TCP/IP protocols by the AC500 hardware/firmware), the data rate at which data should be logged in case of a communication break has to be calculated and limited. It therefore makes sense to use PING to detect a possible interruption in the connection. This enables an earlier detection of the connectionless state.

Mode 2: Buffer and disposal via FTP


Fig. 290: Overview Mode 2
Mode 2 is also used for buffering the values from AC500 application in case of a broken connection between AC500 and telecontrol. State 1 and state 2 are similar to Mode 0/1. The difference is the disposal. When the connection is established again the data logging changes directly back to state 1 and the input values in FIFOin are directly sent to FIFOout (telecontrol connection). The buffered values in File FIFO are internally moved from disk 1 to disk 2 which can then be accessed or used by FTP (client or server). This move action can also be triggered by the command "MoveFile", or when file 1 is full. The advantage of Mode 2 is the immediate availability of the latest and all current values after an outage.

## Use data logging for permanent logging

To use data logging for normal, permanent logging, set Mode $=2$ and make input "BreakConnect" = "TRUE" or "Connected" = "FALSE".

Mode 3: Events recorder


Fig. 291: Overview Mode 3
Mode 3 is used to record data values around an event, before and after the event $X$, e.g. outage of a part of the plant. The values are continuously recorded into the File FIFO file system independent of the connection status to telecontrol. If the File FIFO is full the oldest values are overwritten (ring buffer). Thus the file FIFO always contains the values from the past period $n$, which is depending on the number of values per second and on the size of the File FIFO. When a certain event $x$ occurs, the command "MoveFile" can be given directly or after the period $m$. With the command "MoveFile" the values in File FIFO are internally moved from disk1 to disk 2 and can be read out by an FTP action (client or server) when required.


Fig. 292: The buffered values represent the time before the event ( $n-m$ ) and after the event ( $m$ ).
The advantage of Mode 3 is that the values from the time period before the event ( $n-m$ ) and after the event ( m ) are recorded and can help to reconstruct the cause and effect of the event.

### 6.5.8.3 Technical details

| Parameter | Value |
| :--- | :--- |
| IEC 60870-5-104 protocol inte- <br> grated in data logging, IEC data <br> types | SinglePoint SP1/16, DoublePoint DP, IntegratedTotal <br> IT1/16, MeasurementValue ME1/16 |
| Generic logging to file(s); AC500 <br> data types | BIN, BYTE, INT, UINT, DINT, UDINT, REAL |
| Trigger | Cyclic, event, tolerance |


| Parameter | Value |
| :---: | :---: |
| File format | CSV, including local timestamp. Generic logging with separate ID (max. 8 characters), IEC logging with IEC addresses. <br> Timestamped data sets contain 1-16 values (IEC) depending on type logged. Generic contain different number of values, depending on type logged. <br> BINARY: max. 58 <br> BYTE: max. 88 <br> INT: max. 50 <br> UINT: max. 58 <br> DINT: max. 29 <br> UDINT: max. 31 <br> REAL: max. 27 |
| Data logging target | Flash disk, userdisk or memory card, power fail input for memory card (from UPS) |
| Data logging file sizes and storage depth | FIFO storage in file system, data logging depth only limited by memory size |
| Configurable file FIFO | Number of files (max. 999); number of data sets per file (max. 65535) |
| Internal SRAM FIFO size | 160 data sets (Standard AC500 PLC) <br> Maximum 60 data sets (AC500-eCo PLC) |
| Block write mode into files | Up to 50 data sets/second per max. 88 values |
| Operation modes | Mode 0: Buffer and disposal in chronologic order limited storage (keeps oldest, but stops if full) <br> Mode 1: Buffer and disposal in chronologic order endless (ring buffer) operation modes (deletes oldest) <br> Mode 2: Buffer and disposal via FTP, log file(s) copied to FTP server area for further use <br> Mode 3: Events recorder, logs data before and after an event. |
| Supported software/firmware | V3.4 or higher |
| Current restrictions | One data logger per PLC <br> One IEC 60870 connection only: While log file is replayed, no other current information via IEC available Usable solutions: <br> - Delay replay of log file after connection returned to allow a "general inquiry" <br> - Use of Mode 2 |

## Logging capacity:

- Data set: One data set always has 400 bytes.
- FIFOin: Has a maximum capacity of 161 data sets (a 400 bytes $=64400$ bytes).
- FIFOout: Has a maximum capacity of 161 data sets (a 400 bytes $=64400$ bytes).
- File: One file store up to 65535 data records, which are copied block by block from the FIFOin in case of a communication lost.
- A maximum of 999 files can be saved.

In purely mathematical terms, that would be 999 files * 65535 data sets * 400 bytes/data set $=$ $26,187,786,000$ bytes $=26 \mathrm{~GB}$.

Since neither the flash disk nor the usable memory card have such a capacity, the user has to find a sensible compromise. The flash disk as a storage medium is fail-safe, i.e. in the event of a sudden power failure, data in the possibly currently open file is reconstructed when the power is restored. This is not the case with a memory card. There the file is destroyed. It is therefore advantageous if such a variant is operated with a power supply that keeps the PLC alive for at least a few seconds after the supply voltage failure ("Input ExternalPower" on the "LogHandling" function block).

## Time synchronization:

Currently the AC500 V3 does not distinguish between UTC time and local time and its time zone is set to 0 . This will be improved in near future. In the meantime, it is recommended to store the local time in the AC500 as a workaround.
\#y Chapter 6.5.6.3.5.6 "Time syncronisation" on page 3845

### 6.5.8.4 File names

File names are renamed according to storing time with an accuracy of 100 ms . The files are renamed from "filename.csv" to a file name with timestamp and with or without file extension, according to input "Disk2Extension" is applicable only in Mode 2.

```
File name with 02281448.593 = February 28th, 2:48pm (14:48),59s, 300ms
timestamp
```

File name with $\quad 02281448 . c s v=$ February 28th, 2:48pm (14:48) timestamp and file extension

### 6.5.8.5 Preconditions

## CAUTION!

To avoid failure in processing of the function blocks.

- The function blocks LOGxxxxxx_Input, LogHandling and LOGxxxxxx_Output must be executed in the same task.

The "Data Logger Library" is supported in standard AC500 V3 PLCs, use the library "ABB_Datalogger_AC500".
The "Data Logger Library" can be used with AC500-eCo V3 - PM50x2-T-2ETH using the library "ABB_DataloggerEco_AC500".

- CPU firmware must be V 3.4 .0 or higher.
- Use memory card from ABB with sufficient free space, at least $1.5 \times$ file size as configured (file size is depending on input "MaxNumDatasetFile" from "LogHandling" function block).
- Maximum number of files (input of "LogHandling") is limited to 999. ABB memory card is formatted with FAT by default.
\& Chapter 5.2.9 "Storage devices" on page 1332


## Runtime License:

- To use the "Data Logger Library", runtime license must be activated in the PLC. Runtime license is separate for standard AC500 and AC500-eCo V3 PLC.
If using the:
- "ABB_Datalogger_AC500" library with standard PLC, use the PS5609-Log runtime license.
- "ABB_DataloggerEco_AC500" library with AC500-eCo V3 (PM50x2-T-2ETH), use the PS5609-Log-e runtime license (AC500-eCo library will not run on AC500 Standard), but PS5609-log license can also be used on AC500-eCo library).


### 6.5.8.6 CSV file formats

## Generic data

## logging

| ${ }^{4}$ | 8 | $c$ | D | E | F | 6 | H | 1 | 1 | K | เ | m | N | - | - | $Q$ | R | 5 | T | $u$ | v | w |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 10 | timestamp | Msec | TYPC(NUM) | TYP(TX) | UENGTM | OATA 1.max |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| B00: | DT*2015-05-18-14:54:33 | 491. |  | B00L | 58 | FALSE | false | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | false | FALSE | FALSE | FALSE | FAISE | false | FALSE |
| BITE | OT/2015-05-18.14:34-33 | 491 |  | bite | 88 | 2 | 18. | 5 | 233 | 14. | 54 | 33 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| INT | OTw2015-05-18-10:54-33 | 491 |  | INT | So | 3 | 18. | 5 | 2015 | 14. | 54 | 33 | 7 | 8 | , | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| WORD | DT/2015-05-18-24:54-33 | 492 |  | WORD | 58 | 4 | 18. | 5 | 2015 | 14 | 54 | 33 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 4 | 15 | 16 |
| DINT | DTM2015-05-18-14:34-33 | 492 |  | dint | 29 | 5 | 18 | 5 | 2015 | 14 | 54 | 33 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| OWORD | DTw2015-05-18-14:44-33 | 492 |  | OWORD | 31 | of | 18 | 5 | 2015 | 14. | 54 | 33 | 7 | 8 | 9 | 10 | 11 | 12 | 13. | 14 | 13 | 16. |
| REAL | OTN2015-05-18-24:54:33 | 432 |  | RGAL |  | 7.0 | 13.0 | 5.0 | 2015.0 | 14.0 | 54.0 | 310 | $7 . m$ |  | 9.900 |  | 110 | 12.0 | 13.0 | 14.0 | 150 | 16.0 |

Fig. 293: Explanation of the csv file structure
One data set consists of:
ID (8 any char) + TimeStamp + msec + Datatype (num) + Datatype (txt) + Length (following data) $+\max 32$ data

| Parameter | Value |
| :--- | :--- |
| ID = | ID of LogGeneric_Input (max 8 any characters) |
| DataType = | DataType of LogGeneric_Input (1...7) |
| Length $=$ | Length of LogGeneric_Input (max 88) <br> BINARY (58); BYTE (88); INT (50); UINT (58); DINT (29); UDINT (31); <br> REAL (27) |

## Example

|  | A | 8 | $c$ | 0 | 1 | 9 | H | $t$ | 1 | K | $t$ | M | N | 0 | p | $Q$ | R | 5 | $\dagger$ | $u$ | $v$ | w | $\times$ | $\checkmark$ |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 1 | \$006 |  | an | 18006 | 38 | fust | Falst | falst | Falst | fals | Fmist | facs | Ealst | taise | Faist | trase | Falst | salst | Faist | fals | faist | faisf | Falst | tals |
| 2 | BTE | DTw2015 CSS 18.1654:39 | 491 | 2 8vte | 5 | 2 | 18 | 5 | 23 | 24 | 54 | 33 | 7 | 8 | , | 10 | 12 | 12 | 13 | 24 | 15 | ${ }^{26}$ | 17 | ${ }^{18}$ |
| 3 | ws | DTreoss-e5-31-265:313 | an | 3 lnt | so | 3 | 18 | 5 | 2015 | 14 | 54 | 3 | 7 | 8 | , | 10 | 11 | 12 | 13 | 3 | 13 | 36 | 17 | 18 |
|  | woso | DTw2015 es-18-14.4633 | 491 | 4 WORD | 58 | 4 | 18 | 5 | 2015 | 14 | 54 | 33 | 7 | 8 | , | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | ${ }^{18}$ |
|  | Dint | DTw2015-65-38-1654:33 | 492 | S Dowt | 29 | 5 | 18 | s | 2015 | 14 | 54 | 33 | 7 | 8 | , | 10 | 11 | 12 | 13 | 34 | 13 | 16 | 17 | 18 |
| 6 | OWORD | DT20015 CS -28-2454:33 | 432 | 60 WCat | 3 | 6 | 18 | 5 | 2015 | 14 | 54 | 33 | 7 | 8 | , | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | ${ }^{18}$ |
|  | nsal | DTE2015 CS-18.1654:33 | 492 | 7 REaL | 27 | 7.0 | 18.0 | 50 | 20150 | 14.0 | 54.0 | 33.0 | 7 m | 8.888 | 299 | 10.0 | 11.0 | 12.0 | 13.0 | 14.0 | 13.0 | 160 | 17. am | 18.0 |

Fig. 294: File opened directly with Excel

IEC 60870 data
logging

| 4 | A | B | C | D | E | F | G | H | 1 | J | K | L | M | N | 0 | P | Q |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 1 | TimeStamp | msec | IEC_TYPE | TypText | Slot | Con | $1 d x$ | NoDP | Quality Bits(Byte) | Quality (SQ) | GADU | [AD3/211(1) | Var1 | [AD3/2/1(2) | Var 2 | LAD3/2/1/3) | Var3 |
| 2 | DT\#2012-02-08-10:52:04 | 566 |  | SP1 | 0 | 0 | 198 | 1 | 0 | 0 | 65535 | 66816 | TRUE |  |  |  |  |
| 3 | DT\#2012-02-08-10:52:04 | 566 |  | SP16 | 0 | 0 | 0 | 16 | 0 | 0 | 65535 | 1114369 | TRUE | 1179905 | FALSE | 1245441 | FALSE |
| 4 | DT\#2012-02-08-10:52:04 | 566 |  | IT1 | 0 | 0 | 80 | 1 | 0 | 0 | 65535 | 768 | 4991880 |  |  |  |  |
| 5 | DT\#2012-02-08-10:52:04 | 566 | 5 | IT16 | 0 | 0 | 85 | 16 | 0 | 0 | 65535 | 1049344 | 8 | 1114880 | 2 | 1180416 | 220 |
| 6 | DT\#2012-02-08-10:52:04 | 567 | - 6 | ME1 | 0 | 0 | 203 | 1 | 0 | 0 | 65535 | 512 | 374391.0 |  |  |  |  |
| 7 | DT\#2012-02-08-10:52:04 | 567 |  | ME16 | 0 | 0 | 117 | 16 | 0 | 0 | 65535 | 1049088 | 8.0 | 1114624 | 2.0 | 1180160 | 220.0 |

Fig. 295: Explanation of the csv file structure

Table 741: One data set consists of the following parameters

| Parameter | Description |
| :--- | :--- |
| lecType $=$ | lecType of Loglec60870_Input (1...7) |
| Slot/Con/Idx/NoDP $=$ | Pin group of Loglec60870_Input (1...7) |
| Quality_Bits(Byte) = | IV/NT/SB/BL/CA/CY/QOV (packed in 1 byte) of <br> Loglec60870_Input (1...7) |
| Quality (SQ) = | SQ of LOG_IEC60870_INPUT (1...7) |
| GADU $=$ | Calculated internally, from Automation Builder configurator <br> (Gadu1+Gadu2) |
| IAD3/2/1(n) = | Calculated internally, for every datapoint separately, from <br> Automation Builder configurator (IAD1+IAD2+IAD3) |
| $\mathrm{n}=$ | 1 or 16, in case of DP is n=2 |
| VAR $(\mathrm{n})=$ | Variable |


| IEC type | Values | Meaning |
| :--- | :--- | :--- |
| SP1 | - | SinglePoint 1 |
| SP16 | - | SinglePoint 16 |
| DP | - | DoublePoint |
| IT1 | - | IntegratedTotal 1 |
| IT16 | - | IntegratedTotal 16 |
| ME1 | - | MeasurementValue 1 |
| ME16 | - | MeasurementValue 16 |
| Quality_Bits(Byte): | Quality.0: = IV; | Quality with quality invalid |
|  | Quality.1: = NT; | Quality not topical |
|  | Quality.2: = SB; | Quality substituted |
|  | Quality.3: = BL; | Quality blocked |
|  | Quality.4: = CA; | Quality with quality carry |
|  | Quality.5: = CY; | Quality with quality counter <br> adjusted |
|  | Quality.6: = QOV; | Quality Overflow Quality |
|  | Quality.7: = Reserve; | (*Reserve - Quality*) |
| Quality SQ(Byte) | SQ | Quality sequence number <br> (Range: 0 to 31) |
|  |  |  |

## Example



Fig. 296: File opened directly with Excel

### 6.5.8.7 Comparison V2 and V3 Data Logger Library package

List of differences between V2 and V3 "Data Logger Library".

| Feature | V2 data logger | V3 data logger |
| :--- | :--- | :--- |
| Package | Part of PS563-Water | PS5609-Log |
| Supported in AC500-eCo <br> PLCs | No | Yes (PM50x2-T-2ETH) |
| Buffer size | 160 data sets | AC500 - 160 data sets <br> AC500-eCo - Maximum 60 <br> data sets (default 50) |
| Storage area | Memory card, flash disk | Memory card, flash disk, user- <br> disk |

While updating the library from V2 to V3 platform some of the inputs and output names are adapted in the function blocks. List of adapted names as below, not mentioned variables remains unchanged.

| Function block | Variable type | V2 name | V3 name |
| :---: | :---: | :---: | :---: |
| LogGeneric_Input | Output | RDY | InExecution |
|  | Output | OV | Overflow |
|  | Output | DONE | Busy |
|  | Output | NUMBER_OV | NumberOverflow |
| LogGeneric_Output | Input | SENT | PickedUp |
|  | Output | WORD_ARRAY | UintArray |
|  | Output | DWORD_ARRAY | UdintArray |
|  | Output | - | Busy |
| LogHandling | Input | RESET | CleanAll |
|  | Input | QUIT |  |
|  | Input | DISK1 | NameDisk1 |
|  | Input | DISK2 | NameDisk2 |
|  | Input | DISK2_EXTENTION | Disk2Extension |
|  | Input | MAX_DS_IN_FILE | MaxNumDatasetFile |
|  | Output | DONE | Busy |
|  | Output | RDY | - |
|  | Output | OV | Overflow |
|  | Output | ENUM_ERROR | FileError |
|  | Output | FILE_NEXTWRITE | FileNextWritePosition |
|  | Output | FILE_NEXTREAD | FileNextReadPosition |
|  | Output | USED_FILES | NumberUsedFiles |
|  | Output | NUMBER_DIRECT_S ENDS | NumberDirectSendDatasets |
|  | Output | WRITETIME_DB_IN_FILE | WriteTimeDatablocksInFile |
|  | Output | $\begin{aligned} & \text { READ- } \\ & \text { TIME_DB_FROM_FIL } \\ & \text { E } \end{aligned}$ | ReadTimeDatablocksFromFile |


| Function block | Variable type | V2 name | V3 name |
| :--- | :--- | :--- | :--- |
|  | Output | MAXWRITE- <br> TIME_DB_IN_FILE | MaxWriteTimeData- <br> blocksInFile |
|  | Output | MAXREAD- <br> TIME_DB_FROM_FIL <br> E | MaxReadTimeData- <br> blocksFromFile |
|  | Output | POSSIBLE_SECURE- <br> READ_DS | PossibleSecureRead- <br> Datasets |
| Loglec60870_Input | Output | RDY | InExecution |
|  | Output | OV | Overflow |
|  | Output | DONE | Busy |
| Loglec60870_Output | Input | NUMBER_OV | NumberOverflow |
|  | Input | - | Timeout |
|  | Input | - | Busy |

In all 5 function blocks, all the "INOUT" variables in V2 are changed as
"VAR_INPUT" variable in V3.

### 6.5.8.8 Examples

Example projects for the libraries can be found in the example folder: "Help
$\rightarrow$ Project examples" in the Automation Builder menu.
Prerequisite: Install the related library package with "Automation Builder
$\rightarrow$ Tools $\rightarrow$ Installation Manager" - [Modify].

### 6.5.9 High Availability Modbus TCP

### 6.5.9.1 HA-Modbus TCP - System technology

### 6.5.9.1.1 The AC500 High Availability system

The AC500 High Availability system is designed for the demand of automation systems that require a higher availability, which is realized by redundant devices and communications. The redundancy concept reduces the risk of losing production due to failure of parts of the automation system and thereby minimizes scheduled idle times.
For instance, control can be taken over by the secondary station automatically if the primary station fails.
AC500 High Availability system implements redundancy based on standard AC500 PLCs:

- PLC
- Field communication
- SCADA communication

General differences in high availability / redundancy systems are in which way and how fast the switchover between redundancies happens.

- Cold standby: A replacement system is there but not up and running - Process has (to allow) to completely stop for switchover - e.g. outputs may go to zero.
- Warm standby: Both CPU may be running (= warm) but e.g. communication need to be started/stopped for switch-over - Process needs to tolerate longer freeze times e.g. on outputs - e.g. several seconds.
- AC500 High Availability systems are "hot-standby":
- Redundant CPUs and all communications are always up and running (hot)
- Continuous failure detection in both CPU's and mutual exchange of status
- Continuous synchronization of critical/historical data from primary to secondary
- Automatic switch-over in very short time in case of any failure in primary CPU


Fig. 297: Principle AC500 HA-Modbus TCP architecture example based on Ethernet redundancy

Details of AC500 HA operation along the figure above:

- PLC redundancy: The two PLCs ( A and B ) are running in parallel and calculating and reading.
One is "primary" = active, which means also writing data to field devices.
The other one is "secondary" (= stand-by), also calculating but only reading data from field and receiving synchronization (or short = sync) data from the primary.
- Synchronisation data are critical internal variables with e.g. historical content, which will be transmitted from primary to secondary CPU over the sync connection, so that secondary always has the latest data and can take over immediately. Automatically synced are the historic data of the special HA library function blocks (like counters, timers, integral controllers, ...), additional Data e.g. of events and diagnosis can be synced by the user with sync blocks. The sync connection also transmits a "lifecom1" signal (back and forth) containing diagnosis data of each CPU, so that both CPU know the status of the other CPU. If secondary CPU receives no "lifecom1" anymore it assumes that primary CPU has a failure and takes over primary status. If the sync connection is broken both CPUs would try to adopt primary status, therefore, a parallel separate connection "lifecom2" is used to differential a "sync link" failure from an "other PLC" failure. The "lifecom2" should be routed via a different physical communication path than the data sync/lifecom1, e.g. the Field or SCADA network.
- The field I/O connection is performed via the Ethernet protocol ModbusTCP - connecting the CI 52 x devices (CI521 or CI522) * $乡$ Chapter 5.2.6.5.1 "CI521-MODTCP" on page 1070 \# ${ }^{4}$ Chapter 5.2.6.5.2 "CI522-MODTCP" on page 1111.

For high availability/redundancy of the field or SCADA network, proven Ethernet network redundancy mechanisms are used. (In AC500 this is assumed to be realized by at least 2 (to avoid a single point of failure) external, managed switches), which has the advantage to be able to use AC500 HA with any faster redundancy mechanism / protocol.

- For the I/O communication with CI52x modules two variants exist (see online help: PLC Automation with V2 CPUs $\rightarrow$ PLC integration $\rightarrow$ Device specifications $\rightarrow$ Communication interface modules (S500) $\rightarrow$ Modbus XY)
For smaller systems, the CI 52 x modules can be directly daisy chained (as in previous figure above) if MRP (Media Redundancy Protocol) or DLR (Device Level Ring) is used. Ci52x are not actively participating in ring recovery however, a special FW allows fast ring detection and very short freeze times. Larger systems with e.g. many IO and clusters typically anyway connect to the network via a dedicated managed switch.
- SCADA connection is redundant by nature of the two Ethernet ports and can be extended with further redundancy level as well by managed switches. SCADA itself can also switch the primary PLC to ensure communication to the active PLC in case of a simple connection and a connection failure. If the redundancy mechanism of the OPC DA server is not used, SCADA level itself must be able to handle and differentiate primary and secondary PLC and IP addresses based on the HA-status bits. For CP600 a script exists to do the same for Modbus or AC500 communication protocol.

In most PLC applications the critical components to fail are, beneath PLC, typically the power supply or communication components such as wires or switches. Therefore a SPOF (Single Point Of Failure) has to be avoided by adding redundant devices or redundancy functions wherever a failure likelihood is high and failures are not tolerable.
HA core functionality typically can tolerate only a single failure in the different levels. Then, a repair of the failed part is highly advised to achieve and ensure redundancy again. As shown in the above figure, the I/O-network cabling already provides a second independent redundancy layer e.g. for cable failure by its redundancy mechanism (e.g. ring), which can keep up communication without switching the PLCs: There a second failure in the PLC level could be tolerated as long as both connecting, managed switched still work, but it is highly advised to repair immediately anyway.

The AC500 High Availability system itself only takes care of the first fault. For example, in case of a second fault the primary PLC remains primary PLC until the second fault occurs. This results in no further switchovers (manual switchovers included).

Due to the efficient data sync mechanism, which allows data sync over normal and shared ethernet networks, with a well-planned communication network, the PLCs can operate geographically separated (by many $10^{\text {th }}$ of kilometers). So even in catastrophic events with full mechanical destruction still one PLC will be available to control the process or infrastructure.

The secondary PLC or single CI52x modules can be exchanged in a running system without interruption of the primary PLC or the process. (Check document in "Examples" directory of Automation Builder if HA package was installed.)

Libraries In order to achieve high availability, the CODESYS application must be enhanced with HA function blocks, from the HA-Modbus TCP library and the CI52x library. If the bulk data manager tool (BDM) is used for configuring the System and I/O modules - this is done automatically for the basic initial configuration step by code creation resulting in a prepared user specific "template" application (see below).

- HA-Modbus TCP library contains HA control and HA utility function blocks
- HA control function blocks manage the core HA functionality by collecting diagnosis and switching if necessary.
- HA utility function blocks provide standard functions in the application program with internal sync for integral data e.g. timers, counters, PI control.
- CI52x library contains a function block to configure and communicate to the communication interface modules and ensures that only the primary PLC writes to the outputs. The inputs are read by both PLCs.
- For both PLCs the same application must be used/downloaded.

Bulk data man-
For configuration of the CI52x Modbus TCPs, a separate Bulk Data Manager tool (BDM) is ager tool (BDM)
provided. Especially in larger systems usage of BDM is recommended to comfortably engineer HA and create CI 52 x related configuration and variable data in one place:

- Configuration and parameters of the used I/O modules
- Program code creation for variable naming, configuration, communication and all basic HA functionality
The BDM tool can serve SCADA programming and documentation as well in an efficient manner.


### 6.5.9.1.2 Hardware, requirements and options overview

## Introduction

Two same type AC500 PLCs are required as central hardware components. Each PLC is equipped with at least two Ethernet ports at a processor module or at a communication module. The two PLCs, called PLC A and PLC B, are linked by Ethernet to exchange and synchronize information (Sync). Connections to the AC500 peripheral field devices (I/O) are performed via Ethernet as well.
を) Further information on which CPU type and library to be used.
The following table gives an overview of the different High Availability variants possible with AC500.

The figures are indicative, depend on chosen architectures, system size, network and CPU/ communication modules used.

Table 742: Overview of AC500 HA systems and options

|  |  | HA-CS31 | HA-Modbus TCP |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | version | V2 CPUs | V2 CPUs |  |  | V3 CPUs |  |  |
| I/O communication |  | Parallel serial | Ethernet |  |  | Ethernet |  |  |
|  | CPUs | PM573-595 | $\begin{array}{\|c} \hline \text { PM57 } \\ 3 \end{array}$ | $\begin{gathered} \text { PM59 } \\ 1 \end{gathered}$ | PM595 | PM5630 | PM5650 | PM5670 |
| Parallel serial |  |  | I/O network based on Ethernet and ext.edundancy mechanism |  |  |  |  |  |
| Max. system size CI52x ${ }^{1,6}$ |  | 3-50 ${ }^{\text {2) }}$ | 3 | $\begin{gathered} <251 \\ 50 \end{gathered}$ | $\begin{gathered} <60 / 1 \\ 92 \end{gathered}$ | <30 | < 50 | < 120 |
| I/O modules |  | CI590: S500 | CI52x: S500 and S500-eCo usable ${ }^{4)}$ |  |  |  |  |  |
| Switchover times | CPU | $25-120 \mathrm{~ms}^{3)}$ | Typically $<50 \mathrm{~ms} \sim^{6}$ |  |  |  |  |  |
|  | Field | $15-\underset{3)}{120 \mathrm{~ms}}$ | Depends mainly on network size, redundancy mechanism of external switches ${ }^{7}$ ) |  |  |  |  |  |
| SCADA | ctivity | OPC DA, IEC60870, ... | OPC DA, IEC60870, ... |  |  | OPC DA, OPC UA, IEC60870, |  |  |
|  | rfaces | Several CS31 and Ethernet | Several ETH ports, via CM597 |  |  | Several ETH ports ${ }^{5)}$ <br> (+ 1 CAN Interface ) |  |  |
|  | Sync | UDP | UDP |  |  | UDP |  |  |
|  | ecom1 |  | UDP |  |  | UDP |  |  |
|  | ecom2 | - | Modbus TCP |  |  | Modbus TCP / CAN |  |  |
| Overview of AC500 HA system |  |  |  |  |  |  |  |  |

${ }^{1)}$ Number of CI52x recommendation based on performance or max. number of sockets.
For more details of Modbus clients supported in AC500 V3 PLCs refer to 'Ethernet protocols and ports for AC500 V3' $ڭ^{*}$ Chapter 6.8.2.6.1.2.4 "Limitation of connections per protocol" on page 4484.
${ }^{2)}$ Limited by CPU performance, number of CM574 modules number of CS31 clients and process data limits.
${ }^{3)}$ Depends on system size and CPU type.

## 4) For details on certain S500-eCo modules not supported, see the Automation Builder release notes, Appendix 1.

${ }^{5)} 2 \mathrm{ETH}$ onboard ports - port extensions via CM5640-2ETH: 2 separate ETH ports/communication module for separate networks (no additional sockets or coprocessing, not switched, all ports in different subnets).
${ }^{6}$ ) Based on HA bits switchover, depending on failure case.
 configuration recommendations for HA system" on page 3884
${ }^{7}$ ) Field network: If CI 52 x are used with their 2 ports as part of a ring: In the moment of a network switchover single telegrams may be destroyed: - for V2 ETH onboard: Standard TCP delays repeats by 500 ms - for V3 CPUs onboard/CM5640 or V2 CPU using CM597: A special HA-FW ensures fast repeats of typ. $\sim 50 \mathrm{~ms}$ (settable).

## CPU choice, system size and performance indications

The diagrams below indicate the example choices of AC500 CPU's (horizontal axes) based on the number of communication interface CI-remote I/O clusters (Communication Interface modules; numbers see legend) used in a system and resulting application cycle times (vertical axes).
Further details can be found in 'Task configuration recommendations for HA system' ${ }^{\leftrightarrows}$ Chapter 6.5.9.1.5.3 "Task configuration recommendations for HA system" on page 3884. The values in below graphs base on the assumption to use max. $50-60 \%$ as CPU loading by the bare fast IO communication and HA functionality. So the application load would come on top and cycle times (especially HA, Modbus) need to be relaxed (made higher) compared to below indication.


Fig. 298: Indication of AC500 CPU's performance (horizontal axes) based on the number of communication interface Cl-remote I/O clusters (Communication Interface modules; numbers see legend) used in a system and resulting application cycle times (vertical axes).

Example: If you need a system supporting min. 25 Cl at application cycle time around 120 ms , suitable options based on above graph would be V2 PLCs - PM592 or PM595 and V3 PLCs PM5650 or PM5670. The main parameter in the application cycle determination is the amount of overall Sync data, which is assumed 160 bytes per CI for the smaller systems, up to 250 bytes per Cl for the larger ones. Sync data of the project of in total more than $\sim 1200$ byte necessitates several HA cycles to transfer within one application cycle.
If CM5640-2ETH port extender is used for the CI network, slightly lower performance/longer cycle time needs has to be expected.
The V2 or V3 PLCs types, also differ in available interfaces, protocols supported and memory size.

CI521-MODTCP or CI522-MODTCP can be used as peripheral devices which communicate via the Modbus TCP protocol with the PLCs. The HA-Modbus TCP library supports currently up to 120 CI 52 x , depending on the CPU type as listed in table 'Overview of AC500 HA systems and options' $\Rightarrow$ Further information on page 3863. Each CI52x supports up to a maximum of 10 S500-I/O modules. Nevertheless the standard Modbus TCP communication of the HA library transfers only 120 words per cycle: Therefore please check if for your module configuration matches: In case of many analog IO modules with high-density - like 16 channel AI523/AO523 or modules with fast counters - this limit might be surpassed by roughly 5-6 such modules (to help calculate exactly, there is an Excel sheet provided in the HA "Examples" subfolder of Automation Builder once installed).

For more details of Modbus clients supported in AC500 V3 PLCs refer to 'Ethernet protocols and ports for AC500 V3' ${ }^{\mu}$ Chapter 6.8.2.6.1.2.4 "Limitation of connections per protocol" on page 4484.

> Local I/O on a CPU can only signal/interact for diagnosis or service with/from this CPU. This local I/O is connected only to this CPU - cannot work with other CPU in case of this CPU failure.

## Hardware connections



Fig. 299: AC500 HA and SCADA connection
SCADA/ Engineering connection is done using ETH ports of both PLCs (or their communication module ETH modules) and one or several managed Ethernet switches depending on the redundancy requirements in the SCADA levels.

- HA communication between PLC A and PLC B must be done via two physical connections between PLC A and PLC B in order to distinguish a "sync link" failure from another PLC failure:
- Sync (including "lifecom1") over Ethernet
- "Lifecom2"
over Ethernet (Modbus TCP): Can also be combined with Field or SCADA network over CAN (only possible with AC500 V3 CPU)
(For V3 CPUs: - "Lifecom2" and sync should be directly on the CPU onboard ports, - while for Cl and SCADA network the use of CM5640 is possible to be in separate networks).
- Field devices (CI52x modules) will be connected via Ethernet switches, forming a redundant network (if requested) $\Longleftrightarrow$ Chapter 6.5.9.1.5.4 "Field I/O network topologies" on page 3886.

The following table shows possible combinations of connections for different CPU types. There must be at least two physical connections. The availability can be increased with a third physical connection, e.g. CM597 for AC500 V2 CPUs, CAN or CM5640-ETH for AC500 V3 CPUs.

|  | V2 CPU with 1 ETH onboard + CM597 |  | V2 CPU with 2 ETH onboard (+CM597) |  |  | V3 CPU | with 2 ETH onboard (+CAN) |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Scada/ Engineering | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Sync/ lifecom1 | 1 | 11 | 1 | 2 | 2 | 1 | 2 | 1 | 2 |
| lifecom2 | 11 | 1 | 2 | 1 | 11 | 2 | 1 | 3 | 3 |
| Field (Modbus TCP) | 11 | 11 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| \# of physical connections | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 3 | 3 |


|  | V2 CPU with 1 ETH onboard + CM597 ETH |  | V2 CPU with 2 ETH onboard (+CM597-ETH) |  |  | V3 CPU with 2 ETH onboard (+ CAN) |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Scada / Engineering | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Sync / lifecom1 | 1 | 11 | 1 | 2 | 2 | 1 | 2 | 1 | 2 |
| lifecom2 | 11 | 1 | 2 | 1 | 11 | 2 | 1 | 3 | 3 |
| Field (Modbus TCP) | 11 | 11 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| \# of physical connectiol | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 3 | 3 |

For V3 CPU's with CM5640-2ETH, following as of Automation Builder 2.6.1 new port options are available for SCADA and field connection.


1 ETH1 (orange)
2 ETH2 (green)
3 CAN (light blue, applicable only in V3)
11 CM597 communication module at slot1 (grey)
X (Dark Blue), Y (Yellow): For CM5640-2ETH: Any one of the both ETH ports can be used.
V3 ETH interface internal numbering for each port of CPU and CM5640-2ETH communication module (ETH1, ETH2).

| Port print of <br> device | CPU onboard | Communica- <br> tion module <br> $\mathbf{1}$ | Communica- <br> tion module <br> $\mathbf{2}$ | Communica- <br> tion module <br> $\mathbf{3}$ | Communica- <br> tion module <br> $\mathbf{4}$ |
| :--- | :--- | :--- | :--- | :--- | :--- |
| ETH1 | 1 | 3 | 5 | 7 | 9 |
| ETH2 | 2 | 4 | 6 | 8 | 10 |

The dotted box above indicates the example which is used in the next chapters.
"Lifecom2" connection ="3": when selected as 3-> means using CAN. It has no relation with communication module 1 ETH1 : 3 port.
At present all the CI modules in the ring must be connected either on onboard ETH or only one port of the CM5640.

The numbers in the figure above define the slot on which the connection is made. Last line \# of physical connections define how many physical interfaces are used or connected between the PLCs.

It is also possible to realize an HA system without a communication interface Cl module ${ }^{\star}$, Chapter 6.5.9.1.6.1.2 "Configuration without communication interface modules to establish redundancy" on page 3890.

## Hardware Example

HA hardware example using CM5640 for field and SCADA connection: Both ports of the CPU available for the two sync connections.


Fig. 300: Physical connection example: 2 ETH CPUs combining sync data / "lifefcom2" with SCADA / Field I/O network

Support of I/O modules (S500/S500-eCo) depends on the version of the library package. See the version details of the library in the Automation Builder release notes.

### 6.5.9.1.3 Functionality

Failures and use cases
The AC500 High Availability system performs a switch-over whenever the primary PLC is powered off, crashed or stopped or if the primary PLC loses fieldbus communication (cut of ETH or defect MRP switch) while the secondary PLC still has connection.
In the following figure the different use cases and reaction times are outlined.


Fig. 301: HA use cases - failures, assuming PLC A is primary and "lifecom2" over field network
The below use case table with reaction and diagnosis messages are based on the setup where Sync is via SCADA network, "lifecom2" over field network and PLC A is primary.

| Case | Use case | Reaction | Diagnosis message <br> on <br> *) |
| :--- | :--- | :--- | :--- |
| 1 | Primary PLC is powered off, <br> crashed or stopped. | Switchover to secondary <br> PLC. CI52x outputs are <br> frozen during switchover <br> period. | Secondary |
| 2 | Secondary PLC is powered <br> off, crashed or stopped. | No switchover, process con- <br> tinues. | Primary |
| 3 | Primary PLC loses con- <br> nection to fieldbus CI52x <br> modules while secondary <br> PLC still has a connection. | Switchover to the secon- <br> dary PLC. CI52x outputs <br> are frozen during switchover <br> period. | Primary |
| 4 | Secondary PLC loses con- <br> nection to one or more CI52x <br> modules. | No switchover, process con- <br> tinues. | Secondary |


| Case | Use case | Reaction | Diagnosis message <br> on <br> *) |
| :--- | :--- | :--- | :--- |
| 5 | Cl52x module is stopped/ <br> powered off. | No switchover, process con- <br> tinues. | Primary and secon- <br> dary |
| 6 | Connection lost in Field <br> Ethernet network. | Depending on Ethernet net- <br> work structure, and redun- <br> dancy mechanisms used a <br> reconfiguration time exists. | Lifecom2 lost and CI <br> module lost errors will <br> be generated in pri- <br> mary and secondary. |
| 7 | Sync and/ or "lifecom2" are <br> broken. | No switchover, process con- <br> tinues. | Primary and secon- <br> dary |
| 8 | Primary PLC loses connec- <br> tion to SCADA. | SCADA is responsible to <br> detect and to switch over. | - |
| 9 | Secondary PLC loses con- <br> nection to SCADA. | SCADA is responsible to <br> detect and to switch over. | - |
| 10 | SCADA is broken | SCADA is responsible to <br> detect and to switch over. | - |
| 11 | Manual switchover by the <br> user. | Switchover to the secon- <br> dary PLC. CI52x outputs <br> are frozen during switchover <br> period. | - |
| *) Diagnosis description, see function block description. |  |  |  |

## Use case descriptions

The below cases explain the behavior of the system during different use cases.
Basic diagnosis information is provided for each case ${ }_{y}{ }^{\circ}$ Chapter 6.5.9.1.8 "Diagnosis" on page 3895.

Case 1 a): Primary PLC is powered off or crashes


| Reaction | Switchover to secondary PLC. The communi- <br> cation interface modules are updated by the <br> new primary PLC. |
| :--- | :--- |
| Comment | Cl52x outputs are frozen during switchover <br> period. |
| Diagnosis message on function block | Primary PLC is powered off. <br> Secondary PLC: control block output Runtime <br> Error = 16\#001E and xHaModPrimary = TRUE |

## Case 1 b): Primary PLC is stopped



| Reaction | Switchover to secondary PLC. The communi- <br> cation interface modules are updated by the <br> new primary PLC. |
| :--- | :--- |
| Comment | Cl52x outputs are frozen during switchover <br> period. |
| Diagnosis message on function block | Primary PLC is stopped. <br> Secondary PLC: control block output Runtime <br> Error = 16\#0016 and xHaModPrimary = TRUE |

If "lifecom2" is lost and the PLC is in STOP mode RUNTIME ERROR will not be TRUE. This is because Modbus is still responding even if PLC is in STOP mode.

Case 2 a): Secondary PLC is powered off or crashes


| Reaction | No switchover |
| :--- | :--- |
| Comment | Process continues |
| Diagnosis message on function block | Primary PLC: control block output Runtime <br> Error $=16 \# 001 E$ and $x H a M o d P r i m a r y ~$$=$ TRUE |
| Secondary PLC is stopped. |  |

Case 2 b): Secondary PLC stop


| Reaction | No switchover |
| :--- | :--- |
| Comment | Process continues |
| Diagnosis message on function block | Primary PLC: control block output Runtime <br> Error = 16\#0016 and xHaModPrimary = TRUE <br> Secondary PLC is stopped. |

> If "lifecom2" is lost and the PLC is in STOP mode RUNTIME ERROR will not be TRUE. This is because Modbus is still responding even if PLC is in STOP mode.

Case 3: Primary PLC loses connection to fieldbus CI52x modules


| Reaction | Switchover to secondary PLC. The communi- <br> cation interface modules are updated by the <br> new primary. |
| :--- | :--- |
| Comment | CI52x outputs are frozen during the switch- <br> over period. |
| Diagnosis message on function block | Primary PLC: control block output Runtime <br> Error = 16\#0094 and xHaModPrimary $=$ <br> FALSE <br> Secondary PLC: control block output Runtime |
|  | Error = 16\#0015 and xHaModPrimary = TRUE |

## Case 4: Secondary PLC loses connection to fieldbus CI52x modules



| Reaction | No switchover |
| :--- | :--- |
| Comment | Process continues |
| Diagnosis message on function block | Primary PLC: control block output Runtime <br> Error = 16\#0015 and xHaModPrimary = TRUE <br> Secondary PLC: control block output Run- <br> time Error = 16\#0094 and xHaModPrimary $=$ <br> FALSE |

Case 5: Cl 52 x is powered off or stopped


| Reaction | No switchover |
| :--- | :--- |
| Comment | Process continues |
| Diagnosis message on function block | Primary PLC: control block output Runtime <br> Error = 16\#0081 and xHaModPrimary $=$ TRUE <br> Secondary PLC: control block output Run- <br> time Error = 16\#0081 and xHaModPrimary $=$ <br> FALSE |

If any $\mathrm{CI} 52 x-M O D T C P$ module is powered off and on, there is no need to power restart the complete system. The module will be recognized once the communication is reestablished.

Case 7 a): Sync connection is broken between the PLCs


| Reaction | No switchover |
| :--- | :--- |
| Comment | Process continues |
| Diagnosis message on function block | Primary PLC: control block output Runtime |
|  | Error = 16\#0014 / 16\#0094 and xHaModPri- |
| mary = TRUE |  |
|  | Secondary PLC: control block output Runtime |
|  | Error = 16\#0014 / 16\#0094 and xHaModPri- |
| mary = FALSE |  |

Case 7 b):
Lifecom2 connection is lost between the PLCs


| Reaction | No switchover |
| :--- | :--- |
| Comment | Process continues |
| Diagnosis message on function block | Primary PLC: control block output Runtime <br> Error = 16\#0008 and xHaModPrimary = TRUE <br> Secondary PLC: control block output Run- <br> time Error = 16\#0008 and xHaModPrimary $=$ <br> FALSE |

Case 8: Primary PLC loses SCADA connection


| Reaction | No switchover |
| :--- | :--- |
| Comment | Process continues, SCADA is responsible to <br> detect and switchover |
| Diagnosis message on function block | Primary PLC: control block output Runtime <br> Error = 16\#0000 and xHaModPrimary = TRUE |
| Secondary PLC: control block output Run- <br> time Error = 16\#0000 and xHaModPrimary $=$ <br> FALSE |  |

## Case 9: Secondary PLC loses SCADA connection



| Reaction | No switchover |
| :--- | :--- |
| Comment | Process continues, SCADA is responsible to <br> detect and switchover |
| Diagnosis message on function block | Primary PLC: control block output Runtime <br> Error $=16 \# 0000$ and $x H a M o d P r i m a r y ~=~ T R U E ~$ <br> Secondary PLC: control block output Run- |
| Sech |  |
| time Error = 16\#0000 and xHaModPrimary $=$ |  |
| FALSE |  |

SCADA link may be combined with sync connection or "lifecom2" connection. In that case runtime error and system behavior will be as described in the cases above (Sync connection lost / "lifecom2" connection broken).

## Case 11: Manual changeover by user



| Reaction | Changeover from primary PLC to secondary <br> PLC. |
| :--- | :--- |
| Comment | CI52x outputs will be frozen during switchover |
| Diagnosis message on function block | Primary PLC: control block output Runtime <br> Error = 16\#0000 and xHaModPrimary $=$ <br> FALSE <br> Secondary PLC: control block output Runtime |
|  | Error = 16\#0000 and xHaModPrimary = TRUE |

A manual switchover can be triggered from both PLCs. For each trigger a switchover from primary PLC to secondary PLC will take place.
6.5.9.1.4 How to get and install the AC500 High Availability system package

The PS5601- High Availability Modbus library package can be installed from the Automation Builder Installation Manager by selecting the component.


Fig. 302: Automation Builder Installation Manager
The following components are installed:

- Libraries
- AC500 V2 libraries: C:IProgram Files (x86)\Common FilesICAA-Tar-gets\ABB_AC500\AC500_V12\ibrary\PS5601-HA-MTCP CI52x_AC500_Vxx.lib, HAModbus_AC500_Vxx.lib.
- AC500 V3 libraries available in library repository: ABB_CI52x_AC500.compiled-library, ABB_HaModbus_AC500.compiled-library
- Online help: HA-CS31, HA Modbus V2 function block description
- Automation Builder Example folder: \%ALLUSERSPROFILE\%IDocuments\Automation Builder Examples\PS5601-HA-MTCP
- AC500_V2: Examples for AC500 V2 including documentation
- AC500_V3: Examples for AC500 V3 including documentation
- BulkDataManager: Bulk Data Manager (BDM) tool which helps efficient engineering in larger projects. This requires a separate installation. Further information can be found in the document: \%ALLUSERSPROFILE\%\Documents\Automation Builder Examples $1 P S 5601-H A-M T C P \backslash B u l k D a t a M a n a g e r I D o c u m e n t a t i o n . ~$
- HA-Modbus TCP System Technology.pdf (this document)


### 6.5.9.1.5 System structure

## Introduction

This chapter explains the detailed structure of the HA system in CODESYS. A HA-Modbus TCP system is characterized by two AC500 PLCs with the following features:

- Identical programs (application with additional HA and Modbus function blocks) that are loaded to both PLCs.
- Communication interface modules CI52x-MODTCP that are connected via Modbus TCP.
- Synchronization of both PLCs (sync/lifecom1 and lifecom2 logical connections).


## Programming

Each PLC contains at least three main tasks/ programs:

- HA program
- Application program
- Modbus program

The programs in one PLC communicate via internal structures of the libraries and dedicated internal memory areas for HA-Sync array and the Modbus CI52x memory(ies) CiModDataxx.


Fig. 303: Principle structure of the HA system and recommended tasks: HA, Modbus, Application

Table 743: Image description

| Layout element | Meaning |
| :--- | :--- |
| Dotted outline box | Indicates optional function block or programs. |
| Solid outline box | Indicates the mandatory function blocks or <br> programs. All mandatory blocks are called <br> when an export is created from Bulk data <br> manager. |
| Italic font | Indicates the program or functions user should <br> call in his project and not created by Bulk data <br> manager. |
| Light yellow background block / blue arrow | Indicates the operations which are handled <br> internally in the library. |
| Green solid box | Indicates the three different tasks which user <br> has to configure. |

Modbus pro- The function block CIModCI52x (V3) / CI_MOD_CI52x (V2) reads the input values from the gram CI52x modules and stores them in the structure CiModDataxx. If the CPU is primary it also writes the outputs to the CI 52 x modules. The Function block also parametrizes the Cl Module as configured in e.g. Bulk data manager tool during the first startup or when a Cl module is exchanged.
Normally the HA-Modbus TCP library takes care of communication monitoring. Nevertheless if com- munication is cut completely, the CI52x communication interfaces and its I/O modules have to react on their own to achieve a bumpless or desired behavior: The following parameters for the CI52x com- munication interfaces and I/O modules need to be considered:

-     - CI52x: parameter "Timeout" for Bus supervision: ${ }^{2)}$
Allows to detect errors from communication interface side as well and take action to ensure a fail- safe behavior if communication is cut. It can be set in 10 ms steps. If set to 0 no bus supervision is active. Proposed value: $50=$ $500 \mathrm{~ms}=$ default in Bulk data manager; this value should be increased, e.g. to value 65 if AC500 V2 CPU ports are used for field communication to take care of the larger TCP retransmit time.
- "Behaviour Outputs" at "Timeout for Bus supervision" 1), 2). This fail-safe parameter has to be con- sciously set: separate settings are possible for each module (and communication interface): "off"; "last" or "substitute": 5 s , $10 \mathrm{~s}, \infty \mathrm{~s}^{11}$.


## Remarks:

1) The parameters "Behaviour Outputs at comm. Error" is only analyzed if the Failsafe-mode is [ON].
${ }^{2)}$ Both are CI52x parameters set e.g. via Bulk data manager tool in the program.

## Application program

Example of a utility function block (with integrated sync data)

Consider the on-delay timer HA_MOD_TON (V2)/ HaModTon (V3).


Fig. 304: HaModTon utility function block with internal synchronization
Both PLCs require the same function block called in the program. Under normal operating conditions the elapsed time ET and output Q of the timer is synchronized internally from primary to the secondary CPU. ET and Q data are available and can be attached to local or global variables in the program as per application requirements. If PLC A shuts down due to a fault, the primary status switches over to PLC B.

In the event of a switchover, the moment PLC B becomes the primary, the timer on this PLC will keep running. Until the time of PLC A failure, the timer on PLC B was synchronized. This is most important in cases when one CPU was not in run or off and needs to "catch up" such integral or historic system values (timers, counters, operator settings, ...). The actual process remains then unaffected by the switchover.

## HA program <br> HA_MOD_CONTROL has two functions:

- Exchange status data (lifecom1 and lifecom2) and switch from secondary to primary PLC (or vice versa) based on the status according to the use cases described in 'Failures and use cases' ${ }^{\aleph}$ Chapter 6.5.9.1.3.1 "Failures and use cases" on page 3869.
- Send sync "HA SYNC" array from primary to secondary PLC to ensure that the secondary PLC is always in hot-stand-by and can take over immediately. UDP protocol is used for data synchronization between the CPUs.

Data synchroni- This chapter explains how the data synchronization happens between primary and secondary zation via UDP PLC via UDP.

All prepared sync data is synchronized with the secondary PLC. Typically only integral values (timers, counters, PID , ...) or settings which might have been received have to be synchronized. For example for fast start-up cases when a secondary CPU was restarted, as both PLCs are running and calculating closely in parallel and based on the same input values, synchronization will make the secondary start with current value instead of default value. For details on how to configure or use the data sync function block refer example projects.

Following steps are performed:

- HA SYNC array is transferred via UDP to the secondary CPU. This includes the exchange of lifecom1 status between primary and secondary CPU.
- In the HA program the HA_MOD_CONTROL function block collects all diagnosis, sync and lifecom2 data from the field and/ or the other PLC. Whether a switchover is necessary is decided based on a simple decision matrix.
- Lifecom2 is exchanged between CPUs over Modbus TCP every cycle.
- One task per program, see figure above.
- Status of the inputs connected to CI 52 x decentralized I/O stations is transferred to both PLCs simultaneously in every PLC cycle. They are received by the CI52x function block.
- At the end of the program, the generated output values are sent, by transferring from the primary PLC respective buffers to the $\mathrm{CI} 52 \mathrm{x}-\mathrm{MODTCP}$ module(s) via Cl 52 x function block and Modbus TCP. The secondary PLC is prepared to send but stays "silent" (not sending output values).


One ETH frame copies approx. 1412 data bytes. The number of ETH frames needed to synchronize HA Sync Array completely depends on the number of data sync bytes. Global variable iNoOfEthFrames gives the user this information, which should be used to calculate the cycle time for the application task.
y Chapter 6.5.9.1.5.3 "Task configuration recommendations for HA system" on page 3884
Up to max. 60 kB of Sync data can be synchronized.
Synchronization between the primary and the secondary PLC happens over a few cycles of HA task time depending on the total sync data bytes configured in the system. Lifecom1 is also exchanged between the primary and the secondary PLC. The primary PLC sends lifecom1 to the secondary PLC along with sync data. Backwards the secondary PLC sends lifecom1 to the primary PLC every cycle.
The following figures shows an example for V2 PLC. When in the project the sync data is equal to 4 iNoOfEthFrames then it takes 4 HA cycles to synchronize the data between the PLCs.


Here, it is assumed that 4 HA cycles are needed to transfer sync data from Primary to secondary, meaning global variable iNoOfEthFrames $=4$. Therefore system need 4 HA cycles, 2 Modbus and 1 Application cycle for one round of sync.

When sync data in the project is equal to 6 iNoOfEthFrames then it takes 6 HA cycles to synchronize the data between the PLCs.


Here, it is assumed that 6 HA cycles are needed to transfer sync data from Primary to secondary, meaning global variable iNoOfEthFrames $=6$. Therefore system need 6 HA cycles, 3 Modbus and 2 Application cycle for one round of sync.

## Task configuration recommendations for HA system

For a balanced performance of the HA system consider the following recommendations in your project task configuration:
General

- Use the real time priorities for all HA related tasks. The HA program/ task should be called at highest priority as it is responsible for the core HA functionality and should be the fastest task.
- The Modbus task contains the Modbus communication function blocks at lower priority and (depending on CPU performance) also a faster cycle time to ensure sufficient update rates on Modbus without over- loading the CPU with communication.
- The application program parts should be called in the application task with even lower priority and a larger cycle time than above tasks.
- Configuration to improve standard Modbus TCP for a fast switch over between PLCs.
- AC500 V2
- CM597ETH_SET_TCP_RTO function block from CM597_ETH_AC500_V28.lib needs to be called inside HA task. User needs to call this function block for each CM597 module connected. For recommended values see example description.
- AC500 V3
- RTO retransmission time function block "EthSetRtoMin" for the ETH port where fieldbus communication is configured. By default, minimum retransmission time configured is 15 ms .

| Task | Priority | PM57x, PM58x, PM59x | PM595-4ETH | V3 PLCs |
| :---: | :---: | :---: | :---: | :---: |
| HA | 10 (high) | 4 ms or higher | 2 ms or higher | 4 ms or higher |
| Modbus | 11 (medium) | Maximum of (HA cycle time *2), (3 ms + roundup (\#Cl/2)) | Maximum of (HA cycle time *2), (3 ms + roundup (\#CI/2)) | Onboard ETH: Max ((HA cycle time *2), (3 + roundup (\#CI modules/2))) |
|  |  |  |  | CM5640-2ETH: Max ((HA cycle time * 2 ), (\#CI modules)) |
| Application | 12 (low) | Maximum of (Modbus cycle time *2), (iNoOfEthFrames * HA cycle time) | (iNoOfEthFrames <br> * HA cycle time) | Maximum of (Modbus cycle time *2), (iNoOfEthFrames * HA cycle time *2) |

## Procedure for task configuration

1. Choose suitable CPU type according to chapter CPU choice, system size, performance indications
2. Configure task priorities according to the table
3. Set HA task to minimum according to above table
4. Calculate Modbus cycle time according formulas in the table, based on HA cycle and number of Cl modules "\#Cl"
5. Calculate Application cycle time according to formulas in the table, based on Modbus cycle time and variable iNoOfEthFrames, which is defined in the global variables of HAModbus TCP library.
6. Measure PLC and CPU load during trial operation.

V3: PLC Utilisation $\leadsto$ Chapter 6.8.2.2.1.7 "PLC utilization" on page 4428
If the PLC load is higher than $40 \%$ or CPU load higher than $60 \%$ then increase HA cycle time (e.g. to $8 \mathrm{~ms} / 12 \mathrm{~ms} / 24 \mathrm{~ms}, \ldots$ ) and go to step 4, repeat the steps until loading is within defined range.

## A new V3 CPU configuration option is introduced from Automation

 Builder 2.4.1 and onwards which allows to change the priority for Ethernet communication in PLCs.Set this configuration in the device tree of the CPU in Automation Builder double click on PLC "CPU Parameters $\rightarrow$ Communication Schema $\rightarrow$ Select "Onboard Ethernet"".

The above parameter should be set to "Onboard" Ethernet for HA systems and it will consequently increase the loading due to the higher priority. PLC Load $<50 \%$ and CPU load $<70 \%$ should be considered as guidelines here instead, while setting the task times while setting the task times.
7. Following timeout values has to be defined in the user project according to the relation defined.

| Timeout variables <br> (see defini- <br> tions in box below table) | HA in V2 | HA in V3 |
| :--- | :--- | :--- |
| timCl52xTimeOut | $1^{*}$ Modbus Task time | 50 ms or 2 * Modbus Task <br> time, whichever is higher |
| timHaModSyncTimeOut | $1^{*}$ HA Task time | $2^{*}$ HA Task time |
| timResponseTimeout | Not applicable | 50 ms or (3 * "Modbus Task <br> time"), whichever is higher |
| timCanTimeOut | Not applicable | 100 ms or increase in mul- <br> tiple of 100 |
| timeLifecom2TimeOut | 50 ms | 50 ms |

8. Add additional applications and SCADA communication: Check PLC and CPU load again vs. your requirements.

In the HA Modbus system different timeouts must be configured for the fine operation of the system as described above in the task configuration for V2 and V3 PLCs. These different timeouts meaning, and relation is explained below:

## timHaModSyncTimeOut:

Time limit to check if the new sync data is received or not in the secondary PLC. If this timeout is not defined properly, Sync lost error/ "lifecom1" lost error will be generated.

## timCanTimeOut:

Time used for the check whether "lifecom2" is received when configured via CAN. This value is applicable only in AC500 V3. Lifecom2 via CAN won't be stable between the PLCs and runtime error "lifecom2 lost" will be flickering if not the right value is configured.

## timCl52xTimeOut:

Time limit to check whether new data is received in the Modbus field modules. It is also used to check whether "lifecom2" is received when configured via Modbus TCP. If "timCI52xTimeOut" is not defined as described, "lifecom2" error / communication interface diagnosis error will not be generated as expected.

## timResponseTimeOut:

Timeout value to check whether CPU has lost the communication interface modules connected in the network. If this value is not defined as described, communication interface module lost detection will not be indicated properly.

## timeLifecom2TimeOut

Time limit to check whether "lifecom2" is received when configured via Modbus TCP. Set "timlifecom2TimeOut" value to default 50 ms , if the value is not defined correctly, runtime error "Lifecom2 lost" diagnosis error will not be generated as expected.

Field I/O network topologies

## General

Modbus TCP communication between PLC and communication interface modules CI 521 MODTCP or CI522-MODTCP can be done using different network topologies. In the following subchapters different simple combinations with their pros and cons are explained.

If a CI52x module of a daisy chain is powered off, next following modules will lose connection/ data provided there is no redundancy in the Ethernet network (e.g. ring and managed switch).

## Simple ring topology (smaller systems)

In a simple configuration, CI52x modules can be part of a ring if MRP (or DLR) protocol is used in the managed switches. Then the Cl 52 x are connected from one to another device ("daisy chained") through e.g. two network switches. The redundancy protocol detects a closed ring and opens one port of a managed switch to avoid the ring. The user has to configure the necessary ring configurations and enable the ring manager for the used ring ports in one switch.


Fig. 305: Redundant ring topology with 2 MRP switches (avoids a SPOF (Single Point Of Failure))

## Standard network topology (large systems)

In the standard redundant network, which is often done by third party dedicated telecommunication companies, managed switches are used for every connection point to this network. It's the network's (and operator's) responsibility to repair any failure fast enough so that no influence on the HA system or its outputs occur.

The network can use other fast redundancy algorithms, also having other than ring structures, if redundancy links are activated fast enough.


Fig. 306: Redundant ring topology with independent network (using any fast redundancy mechanism internally in a ring or meshed network)

## Parallel network topology (using PRP)

Each CI52x module and PLCs as single ended devices are connected by PRP switches to both networks. Here the failure of the switch which connects the primary CPU will also lead to a switchover.


Fig. 307: Parallel-redundant network, PRP switches to connect each device, CPU and communication inter- face modules

Redundancy switchover timing should match the settings in the program and communication interface modules for time-out and freeze periods. The networks for larger systems are often seen as a separate entity and done by a separate company. Make sure to have the redundancy status information of the network at least in SCADA, to repair in time. If the I/O field network responsibility is with the automation/ PLC part, the redundancy status should be also monitored by the PLC. A warning to initiate repair may be created from the managed switches in the I/O field network.

Examples - Alarm output(s) wired (e.g. to a CI52x input and related settings of the switch(es)).

- Settings of the switch(es) to send (e.g. SNMP traps, which can be received in PLC (AC500 SNMP library)).
- Use of "automation switches" which can also communicate their status directly via Modbus.

> It is also possible to connect switches in ring combination with CI modules connected to them in daisy chain. User needs to do the relevant setting based on type of switch and protocol (Ex: MRP, RSTP).
> If RSTP ring configuration is used in the system, ring reconfiguration time is slower than other ring protocols. During this reconfiguration, connection to the CI modules will be lost.

## HA Modbus system without communication interface modules in the network

It is also possible to have a HA Modbus system without connecting any field devices, CI521MODTCP / CI522-MODTCP in the network. This system can be used for establishing a redundant PLC system with data synchronization between two AC500 controllers, either without field IO or with user integration of other protocols to field-IO or "intelligent" IO: CPUs as field devices.

Secondary will be on hot standby with primary PLC, during a power off/ stop of the primary PLC. Secondary will take over the control and continue the process. Any user integrated field-IO or CPUs can establish communication mapped with the primary bit: parallel reading but prevent parallel writing.
HA without Cl modules can be also used during commissioning to check the data sync, OPC and SCADA related communications without any field devices configured. The user has to set the global variable 'xNoCiBus' to TRUE defined in the HA_GLOBAL_VARIABLES. This variable has to be set to TRUE in both PLCs. Note: It is not advised to update this variable during run time.


Fig. 308: Simple SCADA connection

### 6.5.9.1.6 Getting started

Quick start list and guidelines

## Workflow

## Engineering



Fig. 309: Engineering workflow using Automation Builder
Simple steps to engineer the HA Modbus system is explained in the following chapters.

## Configuration without communication interface modules to establish redundancy

Configuration of the HA system without communication interface modules to establish redundancy is done by the following steps (for details see the example documentations):

1．Install the hardware $\stackrel{\mu}{\leftrightarrows}$ Chapter 6．5．9．1．2＂Hardware，requirements and options overview＂ on page 3862.

2．Select the CPUs based on the requirements ${ }^{\Perp}$ Chapter 6．5．9．1．2．2＂CPU choice，system size and performance indications＂on page 3864.

3．Install Automation Builder including the latest libraries $\left.{ }^{4}\right\rangle$ Chapter 6．5．9．1．4＂How to get and install the AC500 High Availability system package＂on page 3878.

4．Create a new project in Automation Builder for the chosen CPUs．
5．Configure the required Modbus and UDP configuration in the Automation Builder device tree of the CPU．

```
= -I] HA_Without_CIModules (PM591ETH - TB521ETH)
```



```
        App
    |T0_Bus
        0.0}
    -Q Interfaces
        目且 COM1_Online_Access (COM1 - Online Access)
    10
    目[ FBP_Online_Access (FBP - Online Access)
    ETM Ethernet
        [] ETH1 (ETH1)
        -1] Protocols (Protocols)
```

            Modbus_TCP_IP_Server (Modbus TCP/IP Server)
            - in UDP_no_AC31_header (UDP (no AC31 header))
                if Connection (Connection)
    Extension_Bus
    樶迫 CM597_ETH (CM597-ETH)
            (1) IP_Settings (IP Settings)
            - Modbus_TCP_IP_Server_1 (Modbus TCP/IP Server)
    (1) Slot_2 (TA524)
    6．For UDP in AC500 V2 PLC，configure＂UDP＿no＿AC31＿header＂and set the port number to value＇3000＇．

| ［4］Connection x |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Parameter | Type | Value | Default Value | Unit | Description |
| －Port | WORD（0．65535） | 3000 | 0 |  | Port |
| －Size of receive buffer | WORD（1464．65535） | 4096 | 4096 |  | Set the size of receve buffer |
| －Size of transmit buffer | WORD（0．65535） | 4096 | 4096 |  | Set the sze of transmit buffer |
| －Receive broadcast | Enumeraton of BYTE | Dsable | Dsable |  | Disable／enable broadkast reception（data padage to al stations） |
| －Behaviour on receive buffer overfiow | Enumeration of ETTE | Overwite | Overvite |  | Behoviour on receve buffer overfiow Overurite $=$ the oldest data padages stored in the receive buffer |

7．Assign the IP addresses in $\geq 2$ different Ethernet networks：
－SCADA network：SCADA，connected PLC A and PLC B
－Field network：connected CI52x module（s）
8．Configure the mandatory HA＿MOD＿CONTROL function block for the HA task «＂HA program＂on page 3882.
9．Add Callback stop function HA＿MOD＿CALLBACK＿STOP and call it in the system event ＂stop＂．
10．Add optional HA utility function blocks or function block HA＿MOD＿DATASYNC．

11．Make the global variable $\mathrm{xNoCiBus}=$ TRUE to run the system without communication interface module configured in the system $\Leftrightarrow$ Chapter 6．5．9．1．5．4．5＂HA Modbus system without communication interface modules in the network＂on page 3889.

```
- -I] HA_Without_CIModules (PM5650-2ETH - TB5620-2ETH)
    - 瞁 PLC Logic
        = Application
        +Q Diagnosis
        -D GVL_Variables
            * GV_CP600
            * GV_OPC
            APP_GV__Variables
        -D HA_Config
                d HA_Constants_conf
                目 HA_Program_pro (PRG)
            - Programs
                鱼 APP_PRG (PRG)
                - DataSync (PRG)
                -1 HA_Utilities (PRG)
                - PRG_CALL (PRG)
                if) Library Manager
            D] CALLBACK_STOP (FUN)
            - Symbol Configuration
                Task Configuration
                # Application_Task
                #1 PRG_CALL
            - %) HA_Task
                #] HA_Program_pro
                4) VISU_TASK
            * E1 Visualization Manager
            #HAV3_Visualization
```

12．Add the task configuration $\stackrel{\text { M Chapter 6．5．9．1．5．3＂Task configuration recommendations for }}{ }$ HA system＂on page 3884.

13．Activate the runtime license if it is a V3 PLC to enable HA system $\Leftrightarrow$ Chapter 6．3．2．2．2 ＂PLC runtime licensing＂on page 1446.
14．Compile and download to both PLCs（simplified in V3 via integrated download manager）．
15．Create a boot project，restart the complete system and RUN．
16．Operation：Test use cases（e．g．by putting the primary PLC to STOP mode and observe the switchover）．For different use cases and behavior refer to ．

17．Runtime error and diagnosis function block can be used to monitor the system $\Leftrightarrow$ Chapter 6．5．9．1．8＂Diagnosis＂on page 3895.

## Configuration with communication interface modules and redundancy

For medium or large HA systems the configuration with communication interface modules and redundancy is done by the following steps．For details see the example documentations：
1．Install the hardware $\xi^{\xi}$ Chapter 6．5．9．1．2＂Hardware，requirements and options overview＂ on page 3862.
2．Select the CPUs based on the requirements ${ }^{\wedge} \Rightarrow$ Chapter 6．5．9．1．2．2＂CPU choice，system size and performance indications＂on page 3864.
3．Install Automation Builder including the latest libraries $\Leftrightarrow$ Chapter 6．5．9．1．2＂Hardware， requirements and options overview＂on page 3862.

4．Install the Bulk Data Manager tool（BDM）$\stackrel{y}{\wedge}$ Chapter 6．5．9．1．4＂How to get and install the AC500 High Availability system package＂on page 3878.
5．Create a new project in Automation Builder for the chosen CPUs．
6. Configure the required Modbus and UDP configuration in the Automation Builder device tree of the CPU. UDP settings are only required in AC500 V2 PLCs.

```
= IJ PLC_AC500_V2 (PM591ETH - TB521ETH)
    = [1] Application
        App
=-0
    IO_Bus
        0. DA501 (DA501)
-0 Interfaces
    [1] COM1_Online_Access (COM1 - Online Access)
    14] COM2_Online_Access (COM2 - Online Access)
    |! FBP_Online_Access (FBP - Online Access)
    - ETH Ethernet
        7] ETH1 (ETH1)
        =0) Protocols (Protocols)
            Modbus_TCP_IP_Server (Modbus TCP/IP Server)
        - 有 UDP_no_AC31_header (UDP (no AC31 header))
                    [i] Connection (Connection)
    Extension_Pus
    14! CM597_ETH (CM597-ETH)
        7] IP_Settings (IP Settings)
        - Modbus_TCP_IP_Server_1 (Modbus TCP/IP Server)
    | Slot_2(TA524)
```

7. For UDP in AC500 V2 PLC, configure "UDP_no_AC31_header" and define the port number as '3000'

| 53 Connection $\times$ |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Parameter | Type | Value | Defaut Value | Unit | Description |
| - Port | WORD(0..65535) | 3000 | 0 |  | Pert |
| - Size of receive buffer | WORD(1464.65535) | 4096 | 4096 |  | Set the size of receve buffer |
| - Size of traramt buffer | WORD(0..65535) | 4096 | 4096 |  | Set the size of tranmit buffer |
| - Receive broadcast | Enumeration of BYTE | Disable | Dsable |  | Disable/ensble broadcast reception (data padage to al stations) |
| - Behaviour on recelve buffer overfiow | Enumeration of BrTE | Overwite | Oververite |  | Behoviour on receve buffer overfiow Overunite $=$ the oldest data padages stored in the receive buffer |

8. In AC500 V2 PLCs for each CM597-ETH communication module added the "Send timeout" value has to be changed to 600 ms for the Modbus TCP server.

| *1 CM597_ETH | P_IP_Server_1 $\times$ |  |  |
| :---: | :---: | :---: | :---: |
| Modbus TCP/IP Server Settings |  |  |  |
| Modbus Server Settings <br> Information | Modbus TCP/IP Server |  |  |
|  | Server connections | 1 |  |
|  | Task timeout | 2000 | (ms) |
|  | OMB time | 1000 | (ms) |
|  | Send timeout | 600 | (ms) |
|  | Connect timeout | 31000 | (ms) |
|  | Close timeout | 13000 | (ms) |
|  | Byte order | Big endian $\checkmark$ |  |
|  | Set default values |  |  |

9. Assign the IP addresses in $\geq 2$ different Ethernet networks:

- SCADA network: SCADA, connected PLC A and PLC B.
- Field network: connected CI52x module(s).

10. Configure a network switch in the field network (if managed/redundant) based on network redundancy required $\xi$ Chapter 6.5.9.1.5.4 "Field I/O network topologies" on page 3886.
11. Run BDM tool to configure CI52x network.
12. Export the files. Refer for details in the document: \%ALLUSERSPROFILE\%IDocuments\Automation Builder Examples\PS5601-HA-MTCP\BulkDataManager\Documentation.
13. Import the Bulk data export files to the Automation Builder project.
14. Add Modbus TCP configuration for the ETH ports.
15. For the system with V3 PLCs, set the Communication Schema to "Onboard Ethernet" "CPU Parameters" for better performance.

16. Add Callback stop function HA_MOD_CALLBACK_STOP and call it in the system event "stop".
17. Add optional HA utility function blocks or function block HA_MOD_DATASYNC.

18. Add the task configuration $\stackrel{\Downarrow}{\Perp}$ Chapter 6.5.9.1.5.3 "Task configuration recommendations for HA system" on page 3884.
19. Activate the runtime license if it is a V3 PLC to enable HA system ${ }^{4}$ Chapter 6.3.2.2.2 "PLC runtime licensing" on page 1446.
20. Compile and download to both PLCs (simplified in V3 via integrated download manager).
21. Create a boot project, restart the complete system and RUN.
22. Operation: Test use cases (e.g. by putting the primary PLC to STOP mode and observe the switchover).
23. For different use cases and behavior refer to 'Failures and use cases' * Chapter 6.5.9.1.3.1 "Failures and use cases" on page 3869.
24. Runtime error and diagnosis function block can be used to monitor the system ${ }^{\mu}$ Chapter 6.5.9.1.8 "Diagnosis" on page 3895.

### 6.5.9.1.7 HA-Modbus TCP Limits

HA-Modbus TCP is supported as of Automation Builder 2.0 or higher and the corresponding AC500 CPUs mentioned previously. AC500 V3 PLC is currently not supporting external ETH communication modules. Therefore, onboard ETH1, ETH2 (and eventually CAN) ports are to be used for communication.

3000 sync instances can be used: Either 3000 HA_MOD_DATA_SYNC function block instances alone or together 3000 instances of HA_MOD_DATA_SYNC inclouding + HA utility function block can be used. If more than 3000 instances are configured user can see the error at


The maximum length of sync data at an instance of HA_MOD_DATA_SYNC function block would be 1412 bytes. The maximum size of sync data which can be synced between PLCA and PLCB in total can be max. 60000 bytes.
The HA-Modbus TCP system takes care of the first fault only. This fault must be visualized by the programmer and overall system (e.g. HMI, SCADA) to the operator, to plan and repair as soon as possible as redundancy might be lost. If more than one error occurs, system may not react to second or following faults.
SCADA/ HMI has to be configured/programmed to:

- Only read data from the primary PLC.
- Parameters and control data should be always written to both PLCs or has to be synchronized via the function block.
This is given automatically when using OPC DA, where the CODESYS OPC Server does this switching for the connected clients according to the primary status. For CP600 HMI a script is available to switch likewise (connected via the internal AC500 protocol or Modbus). Zenon as a SCADA also uses the AC500 protocol to automatically switchover.


### 6.5.9.1.8 Diagnosis

## General

This chapter explains the diagnosis information available to the user in the HA Modbus library and CI52x library. Diagnosis information is available at the outputs of HA control function block, HA Diagnosis function block and at the CI 52 x function block.

Depending on the use case different diagnosis information can be accessed.
\& Chapter 6.5.9.1.3 "Functionality" on page 3869
Primary CPU currently can read-out the diagnosis information (CI52x function
block outputs) from communication interface module only once, hence secon-
dary PLC will not be able to read the diagnosis information from the CI52x
module.
So if any change happens in CI52x diagnosis it is not reflected in the secondary
CPU.
This can lead to different diagnosis information of CI52x module in the primary
and the secondary CPU. Hence it is recommended to customers that diagnosis
information should be handled in the application (e.g. SCADA).

## Diagnosis in HA-Modbus TCP library

In the HA Modbus library diagnosis information is available at the control block and diagnosis block.

## Output System Configuration

This output at the HA control block gives the information of system configuration. Each bit of the word represents a different configuration.

| Bit | Description |
| :--- | :--- |
| 0 | Sync is configured via CAN |
| 1 | Sync is configured via UDP |
| 2 | Lifecom2 is configured via CAN |
| 3 | Lifecom2 is configured via UDP |
| 4 | Lifecom2 is configured via Modbus TCP |
| 5 | Initialization for Ethernet configuration |


| Output System |
| :--- |
| Configuration |
| error |


| Bit | This output at the HA control block gives the details of error in the configuration. Each bit of the <br> word represents different configuration errors. It is valid only when Error $=$ TRUE. |
| :--- | :--- |
| 0 | Description |
| 1 | Communication interface module is not configured properly |
| 2 | $1<$ SyncSlot >3. Invalid value at input sync slot |
| 3 | $1<$ SecSlot>3. Invalid value at input second slot |
| 4 | Value at IpAdrCpuASync is invalid |
| 5 | Value at IpAdrCpuBSync is invalid |
| 6 | Value at IpAdrCpuALifecom2 is invalid |
| 7 | Value at IpAdrCpuBLifecom2 is invalid |

Output Runtime This output at the HA control block gives the details of the error during run time of the system. error

| Bit | Description |
| :--- | :--- |
| 0 | Communication interface modules are lost |
| 1 | Other CPU is not active |
| 2 | Lifecom1 is lost (part of sync) |
| 3 | Lifecom2 is lost. This error will not be TRUE if the PLC is in STOP status. This is <br> because Modbus is still responding even when PLC is in STOP |
| 4 | Synchronization is lost |
| 5 | Error in synchronization |
| 6 | Ethernet status error |
| 7 | Other PLC lost communication to CI52x modules |
| 8 | CAN_HEADER function block has error |
| 9 | CAN_DATA function block has error |
| 10 | fbGetOwnIP function block has error |

Diagnosis func- Outputs at the HA Diagnosis function block, HaModDiag (V3) / HA_MOD_DIAG (V2) provides tion block the following diagnosis information of the HA system.

| Output | Description |
| :--- | :--- |
| CpuAPrimary / CPUA_PRIMARY | TRUE indicates CPU A is primary |
| CpuBPrimary / CPUB_PRIMARY | TRUE indicates CPU B is primary |
| CpuARun / CPUA_RUN | TRUE means CPU A is in RUN mode |
| CpuBRun / CPUB_RUN | TRUE means CPU B is in RUN mode |
| CpuACI52xBusActive / <br> CPUA_CI52x_BUS_ACTIVE | Modbus TCP CI52x bus active on CPU A |
| CpuBCI52xBusActive / <br> CPUB_CI52x_BUS_ACTIVE | Modbus TCP CI52x bus active on CPU B |
| CpuACi52xCfg / CPUA_CI52x_CFG | Total number of CI52x configured on CPU A |
| CpuBCi52xCfg / CPUB_CI52x_CFG | Total number of CI52x configured on CPU B |
| CpuACi52xAct / CPUA_CI52x_ACT | Total number of CI52x active on CPU A line |
| CpuBCi52xAct / CPUB_CI52x_ACT | Total number of CI2x active on CPU B line |
| SyncInstances / SYNC_INSTANCES | Number of data sync and utility blocks initial- <br> ized in the system |
| SyncDataCheckSum / SYNC_DATA_SUM | Checksum of all address pointer blocks in <br> bytes, indicates total number of bytes getting <br> synchronized. |


| Output | Description |
| :---: | :---: |
| StHACpuStatus / stHA_CPU_STATUS | HA own CPU status. It will show the status details of logged in CPU for the following parameters: <br> - HA1: CPU A is primary <br> - HA2: CPU B is primary <br> - bit_CI52x_BUS_active: $\mathrm{CI} 52 x$ bus with one or more communication interface modules active <br> - bit_CI52x_BUS_err: CI52x bus one or more communication interface modules powered off / connection lost <br> - RUN: Run status of Cl 52 x <br> - cnt: Count of data sync communication, indicates data sync between CPUs is okay. |
| StHAotherCpuStatus / stHA_OTHER_CPU_STATUS | HA other CPU status. It will show the status details of other CPU for the following parameters: <br> - HA1: CPU $A$ is primary <br> - HA2: CPU B is primary <br> - bit_CI52x_BUS_active: $\mathrm{Cl} 52 x$ bus with one or more communication interface modules active <br> - bit_CI52x_BUS_err: CI52x bus one or more communication interface modules powered off / connection lost <br> - RUN: Run status of Cl 52 x <br> - byETH_ACT_CI52x_Count: CI52x alive identification count. |

Other diagnosis Apart from the errors / diagnosis information available in the control and diagnosis block, few variables other variables can be monitored too.

| Variable | Value | Description |
| :--- | :--- | :--- |
| wHA_ER_NO_SYNC_LINK | 16\#7487 | No sync link between the <br> PLCs |
| HA_MOD_INVALID_LENGTH | 16\#2017 | Invalid length at the input of <br> the data sync block |
| HA_MOD_ERNO_TBL_OVER <br> FLOWW | $16 \# 2022$ | HA data reference table is full |
| xHaModDataErr | TRUE | IF TRUE - HA data sync is in <br> error state |
| wHaModDataErNo |  | HA data sync error code |
| xHaModErr | TRUE | HA system is in error state |
| dwHaModServerAlive |  | Life counter incremented by <br> OPC DA server |

## Diagnosis in CI52x library

In addition to the diagnosis information in the HA Modbus library, additional diagnosis information for each communication interface module can be obtained from the CI52x library.

System Config- This output at the CI52x function block gives the details of the configuration error in the Cl 52 x uration error module. Each bit of the byte represents different configuration errors:

| Bit | Description |
| :--- | :--- |
| 0 | Reserved |
| 1 | Wrong ETH port is configured at input Config ETH |
| 2 | Wrong IP address is configured for communication interface module |

$\begin{array}{ll}\text { Runtime error } & \begin{array}{l}\text { RuntimeError (v3) / RUNTIME_ERROR (v2) of the function block CiModCi52x (v3) / } \\ \text { CI_MOD_CI52x (v2). Runtime error is a combination of error bits that are described in the }\end{array} \\ & \text { following: }\end{array}$

| Runtime Error | Description |
| :--- | :--- |
| Bit 0 | Indicates communication error i.e., when CPU is not able to <br> get any response from CI52x module. This error will get reset <br> when communication is reestablished. |
| Bit 1 | Indicates parameter state is not equal to 2 <br> (PARA_STATE_PARA_DONE). If not true, then system gives <br> I/O bus error. System resets this error when parameter state is <br> equal to 2. |
| Bit 2 | Indicates the cluster error ${ }^{1}$ ) in the system, if there is an error <br> in the diagnosis buffer. ACK input is needed to reset this error. |
| Bit 3 | Indicates the hardware configuration error, mismatch between <br> configuration and actual hardware detected. System automati- <br> cally resets this error when the hardware matches. |
| S-ERR (LED on communica- <br> tion interface module) | Indicates that there is some issue with channel configuration in <br> the cluster 1 ). It is not linked with Runtime Error. User can read <br> DiagBuffer (v3) / DIAG_BUFFER (v2) from CiModDiag (v3)/ <br> CI_MOD_DIAG (v2) function block to get more information. <br> This error does not get reset using ACK. It will only reset when <br> all channel errors are removed. |
| 1) "Cluster" means a combination of one communication interface module and several I/O <br> modules attached to it. |  |

Runtime error in different scenarios:

| Error | Runtime error | PLC A: Primary |  |  |  | PLC B: Secondary |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | Bit0 comm error | Bit1 I/O bus error | Bit2 cluste $r$ error | Bit3 HW config error | Bit0 comm error | Bit1 I/O bus error | Bit2 cluster error | Bit3 HW config error |
| Wrong IP address configured | 16\#1 | TRUE | FALSE | FALSE | FALSE | TRUE | FALSE | FALSE | FALSE |
| Wrong slot address configured ${ }^{1}$ ) | 16\#0 | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE |
| Communication cable disconnected | 16\#2 | TRUE | TRUE | FALSE | FALSE | TRUE | TRUE | FALSE | FALSE |
| Wrong I/O module plugged in the CI module | 16\#B | BLINK | TRUE | FALSE | TRUE | BLINK | TRUE | FALSE | TRUE |
| Wrong hotswap I/O module plugged at the start | 16\#B | BLINK | TRUE | FALSE | TRUE | BLINK | TRUE | FALSE | TRUE |
| Wrong hotswap I/O module swapped online | 16\#4 | FALSE | FALSE | TRUE | FALSE | FALSE | FALSE | FALSE | FALSE |
| Configured I/O module not connected at start ${ }^{2}$ ) | 16\#B | BLINK | TRUE | FALSE | TRUE | BLINK | TRUE | FALSE | TRUE |
| Configured hotswap I/O module not connected at start ${ }^{2}$ ) | 16\#B | BLINK | TRUE | FALSE | TRUE | BLINK | TRUE | FALSE | TRUE |
| I/O module powered off in Cl module ${ }^{2}$ ) | 16\#4 | FALSE | FALSE | TRUE | FALSE | FALSE | FALSE | FALSE | FALSE |
| Hotswap I/O module powered off in Cl module ${ }^{2}$ ) | 16\#4 | FALSE | FALSE | TRUE | FALSE | FALSE | FALSE | FALSE | FALSE |
| Remove hotswap I/O module when online ${ }^{2}$ ) | 16\#4 | FALSE | FALSE | TRUE | FALSE | FALSE | FALSE | FALSE | FALSE |


| Error | Runtime error | PLC A: Primary |  |  |  | PLC B: Secondary |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | Bit0 comm error | Bit1 I/O bus error | Bit2 cluste r error | Bit3 HW config error | Bit0 comm error | Bit1 I/O bus error | Bit2 cluster error | Bit3 HW config error |
| Cl module is powered off | 16\#2 | TRUE | TRUE | FALSE | FALSE | TRUE | TRUE | FALSE | FALSE |
| Mismatch in Channel configuration and wiring ${ }^{3}$ ) | 16\#0 | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE |
| Regular I/O module mounted on hotswap terminal unit ${ }^{4}$ ) | 16\#0 | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE |
| During an error stage if HA system changeover is initiated 5) | 16\#0 | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE | FALSE |

[^50]Communication interface diagnosis

Table 744: Function block CiModDiag (V3) and CI_MOD_DIAG (V2)

| Output | Description |
| :---: | :---: |
| DevState / DEV_STATE | CI521 or CI522 device current status is displayed. <br> - STATE_PREOP: Device is booting <br> - STATE_OPERATION: Device is operational, no bus supervision is active <br> - STATE_ERROR: Device detected a bus error, bus supervision is active <br> - STATE_IP_ERROR: Device has an IP address error <br> - STATE_CYLIC_OPERATION: Device is operational, bus supervision is active <br> - STATE_NA: Not available |
| ParaState / PARA_STATE | CI521 or CI522 device parameter status. <br> - PARA_STATE_NO_PARA: Device has no parameters <br> - PARA_STATE_PARA_ACTIVE: Parameterization process is running <br> - PARA_STATE_PARA_DONE: Device used valid parameters and parameterization is done <br> - PARA_STATE_ERROR: Device has invalid parameters <br> - PARA_STATE_NA: Not available |
| Devicelnfo / DEVICE_INFO | CI521 or Cl522 type and extended module types. This will give the details of the module configured in the communication interface module including the I/O modules. <br> If module is with suffix $F$, then fast counter is enabled for that module. |
| DiagBuffer / DIAG_BUFFER | CI521 or CI522 module diagnosis buffer $\Rightarrow$ Chapter 6.8.4.1.2.3.3 "Diagnosis data" on page 4576. |
| ErClass / ERR_CLASS | Communication interface error class « Chapter 6.9.3 "Diagnosis messages" on page 4655. |
| ErNo / ERR_NO | Communication interface error number ${ }^{\Perp}$ Chapter 6.9.3 "Diagnosis messages" on page 4655. |
| ModMastErr / MODMAST_ERR | Latest 22 Modbus TCP error message status of the ModMastTcp (V2) / COM_MOD_MAST (V2) function block. |
| ModMastErNo / MODMAST_ERR_NO | Latest 22 Modbus TCP error numbers. Refer to the error details in Modbus library ${ }^{\wedge} \Rightarrow$ Chapter 6.5.9.1.8.2 "Diagnosis in HA-Modbus TCP library" on page 3896. <br> V3: Refer to the ERROR_ID enumeration in the Modbus TCP library (in the Library Manager) |

### 6.5.9.1.9 Library overview

Documentation CODESYS V3 libraries are described in the Library Manager as an integrated documentation Refer to the documentation section within the Library Manager.

The following function blocks are contained in the libraries:

HA－Modbus TCP
library

```
* history
Bnums
    * ERROR_ID
Function Blocks
    -G Control
        + [] HaModControl
        产 HaModDataSync
    + 吾 HaModDiag
    +
    InternalFunctionBlocks
    -G Utility
        星 HaModAIO
        䬱 HaModCtd
        言 HaModCtu
        注 HaModCtud
        星 HaModDerivative
        产 HaModDIO
        鱼 HaModIntegral
        曾 HaModPid
        並 HaModPidFixCycle
        [星 HaModRampInt
        星 HaModRampReal
        請 HaModTof
        鱼 HaModTon
F
Functions
-m
CALLBACK
    E] HaModCallbackStop
    \pm
        InternalFunctions
-m
GlobalConstants
    HA_CONSTANTS
-
GlobalVariables
    HA_GLOBAL_INTERNAL
    HA_GLOBAL_VARIABLES
Types
-
Visualization
- Control
    榎 VisuHaModControl
    㛀 VisuHaModDataSync
    睤 VisuHaModDiag
+
Utility
䀦 VisuHaOverview
```

HA_CI52x library
history
+- Internal Function Blocks
$\pm$ - CiModCi52x
$\pm$ -
-6 Functions
$\pm$ Internal Functions

- GlobalConstants
CI52x_GLOBAL_CONSTANT
- GlobalVariables
CCI52x_GLOBAL_INTERNAL
CCI52x_GLOBAL_INTERNAL
CI CI52x_GLOBAL_VARIABLES
- Types
$\pm$ DIAG
$\oplus$ Internal StructuresIO Module Parameter Structure IO Module Structures
- Visualizations
$-\operatorname{CI} 52 \mathrm{x}$
    - Function Blocks
略 VisuCiModCi52x
VisuCiModDiag


### 6.5.10 Motion Control

### 6.5.10.1 Motion wizard

This section is an introduction to the key features and tools, which will be used during the motion solution project.

### 6.5.10.1.1 Introduction to the project

Understanding the "Motion Solution Project"
The project type "Motion Solution Project" is an AC500 project with an inbuilt "Motion Solution Wizard" which helps guide the user intuitively though a step-by-step process to add and configure the motion axis in a few simple steps.

## Understanding the "Motion Solution Wizard"

The "Motion Solution Wizard" is a tool which helps the user to efficiently configure the motion axis. The motion wizard helps the user to configure the axis based on PTO or EtherCAT using 2.7.0.

The time saving functions this includes are; code writing, adding hardware and intuitively setting up PDO mapping. Afterwards the user can proceed by adding further PLCopen function blocks based on their application needs.

## Understanding the "Motion Specific Device" objects

When using the motion solution to commission the user will come across motion specific objects. These are devices which allow the user to represent:

- Real EtherCAT axes
- Real PTO axes
- Encoders (used for following or closing the position loop)
- Virtual axes (used for following)

Each of these elements contain a motion axis object for configuration.

## Understanding the motion axis objects

When using the "Motion Solution Project", motion axis objects are used to configure the axis behavior. These are device tree elements which allow the user to:

- define the motor position system type,
- mechanical units, gearing, and scaling
- control settings (PID)
- PLC Limits (Speed, acceleration etc.)
- PDO mapped data (Only for EtherCAT axis)
- "Drive Control Mode" (Only for EtherCAT axis)


### 6.5.10.1.2 Installing the latest "Motion Control Wizard" and libraries

As described earlier, during the Automation Builder installation process, or later when opening the Installation Manager the "Motion Control (PS5611)" option can be selected and installed. After this step is carried out, the latest version of libraries and motion control tools (such as the "Motion Control Wizard", Cam editor etc) are available to be used.

### 6.5.10.1.3 Introduction to the PLC capabilities

Selection of the correct AC500 PLC type
AC500-eCo CPU (type dependent)

- Supports up to 4 PTO channels - (1ms> Profiling time)
- Support onboard EtherCAT - (1ms> Profiling time)

AC500 CPU (e.g.PM56xx-2ETH)

- Supports only motion via EtherCAT using the CM579-ECAT communication module $(0.5 \mathrm{~ms}$ $>)$
Each variant has a different limitation which are explained in the following chapters.


## PTO operations

Due to differences in the performance of CPU types, there are different limits.

PTO profiling cycle time limits
On the minimum cycle times configurable in each PLC type. These are detailed below:

| PLC Type | PM5032 | PM5052 | PM5072 | PM 5082 |
| :--- | :--- | :--- | :--- | :--- |
| Min. cycle time | 5 ms | 2 ms | 1 ms | 1 ms |

## PTO hardware limits

Withing the AC500-eCo range there are different revisions hardwares available. The revision 0 hardware can only support 2 * 200 kHz or 4 * 100 kHz PTO, where in revision 1 or higher can support 4 * 200 kHz PTO channels. To know if you have a revision 1 AC500-eCo PLC, please refer to the SAP number on the side of the PLC.

- "Revision 1 CPU" has order number 1SAPxxxxxxR017x
- "Revision 0 CPU" has order number 1SAPxxxxxxR007x


## Selection of different revision CPU's

Within the wizard page there is a checkbox for selecting of hardware is revision 1 or higher (default) or older revision 0 hardware. The tick box should be made matching your revison.

| Motion axes cycle time | 5 |  | ms |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Axis name | Axis type | HW Type | HW Info | Channel/Node | Status | Simulate | Generate |
| PTO_Axis | Finite | PTO | Onboard | PTO 0 (DO00-D004) | Not generated |  | $\square$ |
| PTO_Axis_1 | Finite | PTO | Onboard | PTO 1 (D001-D005) | Not generated | $\square$ | $\checkmark$ |
| PTO_Axis_2 | Finite | PTO | Onboard | PTO 2 (D002-D006) | Not generated | $\square$ | - |
| - Hide generated code |  |  |  |  |  | $\checkmark \mathrm{V}$ | Revision 1 |

## Revision 1 or higher CPU

## Up to 4 * 200 kHz (default) PTO axis

When the axis is configured in revision 1 CPU, output channels 0 to 3 are available for 200 kHz outputs. This means "Output 0" to "Output 3" for direction (PTOx Dir) and "Output 4" to "Output 7 " for pulse/step (PTOx HS Pulse) and users need to connect the PTI drive cables accordingly.
Wiring example:


The below table shows the user an overview of the hardware channels configured based on the number of axes configured, this can be used as a reference for the pulse and direction wiring to PTI drive.


Revision 0 AC500-eCo CPU

Number of PTO axis per PLC (Below are the maximum PTO axes combinations possible based on the PTO frequency configured).

- Max 4 PTO axis -100 kHz
- Max 3 PTO axis - two 100 kHz and one 200 kHz ( 200 kHz axis must not be configured between to 100 kHz axis)
- Max 2 PTO axis - 200 kHz

Up to 4 * 100 kHz (default) PTO axis with "Revision 0 CPU"

When the axis is configured as 100 kHz , output channel "Output 0 " to "Output 3 " are configured as direction output (Value = PTOx Dir) and "Output 4" to "Output 7" are configured as pulse (Value = PTOx LS Pulse) and users need to connect the PTI drive cables accordingly.
The below table shows the user an overview of the hardware channels configured based on the number of axes configured, this can be used as a reference for the pulse and direction wiring to PTI drive.


PTO axes 100 When two PTO axes are configured as 100 kHz , the "Motion Solution Wizard" set the onboard kHz output configuration. For the first axis, direction as "Output 0 " and pulse as "Output 4" and for the second axis direction as "Output 1 " and pulse as "Output 5".


Up to 2*200 kHz PTO axis with "Revision 0 CPU"

When the axis is configured as 200 kHz , output channel "Output 4" (Value = PTOx HS Pulse) and "Output 5" (Value = PTOx HS Dir) are configured as pulse and direction output for the first axis and "Output 6" and "Output 7" is configured as pulse and direction output for the second axis.

Below table shows the user an overview of the hardware channels configured based on the number of axes configured, this can be used as a reference for the pulse and direction wiring to PTI drive.


|  |  |  | Axis frequency in $\mathbf{k H z}$ |  |
| :---: | :---: | :---: | :---: | :---: |
|  | DO5 | PWM/PTO1 LS Pulse <br> PTOO HS Dir/Cw <br> PTOO HS Single 1 | PTO0 HS Dir/Cw | PTO0 HS Dir/Cw |
|  | DO6 | PWM/PTO2 LS <br> Pulse <br> PTO1 HS <br> Pulse/Cw <br> PTOO HS Single <br> 2 |  | PTO1 HS <br> Pulse/Cw |
|  | DO7 | PWM/PTO3 LS <br> Pulse <br> PTO1 HS Dir/Cw <br> PTOO HS Single 3 |  | PTO1 HS Dir/Cw |

PTO axes 200 When two PTO axes are configured as 200 kHz , the "Motion Solution Wizard" set the onboard kHz output configuration. For the first axis, Pulse as "Output 4" and direction as "Output 5" and for the second axis pulse as "Output 6" and direction as "Output 7".

| Parameter | Value |
| :---: | :---: |
| - Run on config fault | No |
| +- Digital inputs 24 VDC |  |
| $=-$ Digital outputs 24 VDC / 0,5A transistor |  |
| - Output 0, channel configuration | Output / PTOO Dir |
| - Output 1, channel configuration | Output / PTO1 Dir |
| - Output 2, channel configuration | Output / PTO2 Dir |
| - Output 3, channel configuration | Output / PTO3 Dir |
| - Output 4, channel configuration | PTOO HS Pulse/Cw |
| - Output 5, channel configuration | PTOO HS Dir/Ccw |
| - Output 6, channel configuration | PTO1 HS Pulse/Cw |
| - Output 7, channel configuration | PTO1 HS Dir/Cow |
| Variable Channel <br> +- Digital inputs 24 VDC  |  |
|  |  |
| - Digital outputs 24 VDC / 0,5A transistor |  |
| $\pm{ }^{*}$ | Fast outputs DOO-D... |
| $\cdots$ | Fast outputs D04-D... |
| *) PTO_Axis_1_Pulse | Digital output DO4 |
| *) PTO_Axis_1_Direction | Digital output DO5 |
| *) PTO_Axis_2_Pulse | Pigital output D06 |
| \% PTO_Axis_2_Direction | Digital output D07 |

Up to 2* 100 kHz and 1 * 200 kHz PTO axis with "Revision 0 CPU"

When some axes are configured as 100 kHz and some are configured as 200 kHz frequency, user must take care following points:

- Make sure the 200 kHz axis is configured as first or last axis and not in between 100 kHz axis.
- When the first axis is 200 kHz , Automation Builder will configure the "Output 4" and "Output 5 " channels and for 100 kHz "Output 6" and "Output 7" is configured as pulse and "Output 2" and "Output 3" are configured as direction.

Below table shows the user an overview of the hardware channels configured based on the number of axes configured.

|  |  | Axis frequency in kHz |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | Axis1 | 100 | 200 | 100 | 200 | 100 |
|  |  | Axis2 | 200 | 100 | 100 | 100 | 200 |
|  |  | Axis3 |  |  | 200 | 100 | 100 |
|  |  | Axis4 |  |  |  |  |  |
| HW channel selection | DO0 | Output/ PTO0 Dir | $\begin{aligned} & \text { Output// } \\ & \text { PTO0 Dir } \end{aligned}$ |  | Output/ PTOO Dir |  |  |
|  | DO1 | Output/ <br> PTO1 Dir |  |  | $\begin{array}{\|l} \text { Output/ } \\ \text { PTO1 Dir } \end{array}$ |  |  |
|  | DO2 | Output/ <br> PTO2 Dir |  | Output/ <br> PTO2 Dir |  | Output/ <br> PTO2 Dir |  |
|  | DO3 | Output/ <br> PTO3 Dir |  |  |  | Output/ <br> PTO3 Dir |  |
|  | DO4 | PWM/ PTOO LS <br> Pulse <br> PTOO HS <br> Pulse/Cw <br> PTOO HS <br> Single 0 | PWM/ <br> PTO0 LS <br> Pulse | PTOO HS Pulse/Cw | PWM/ <br> PTO0 LS <br> Pulse | PTOO HS Pulse/Cw |  |
|  | DO5 | PWM/ PTO1 LS <br> Pulse <br> PTO1 HS Pulse/Cw <br> PTOO HS <br> Single 1 |  | PTOO HS Pulse/Cw | PWM/ PTO1 LS Pulse | PTOO HS Pulse/Cw | Configuration not allowed. |
|  | DO6 | PWM/ PTO2 LS <br> Pulse <br> PTO2 HS <br> Pulse/Cw <br> PTO2 HS <br> Single 2 | PTO1 HS Pulse/Cw | PWM/ <br> PTO2 LS <br> Pulse | PTO1 HS Pulse/Cw | PWM/ <br> PTO2 LS <br> Pulse | kHz axis as first or last axis. |
|  | DO7 | PWM/ PTO3 LS <br> Pulse <br> PTO3 HS <br> Pulse/Cw <br> PTOO HS <br> Single 3 | PTO1 HS Pulse/Cw |  | PTO1 HS Pulse/Cw | PWM/ <br> PTO3 LS Pulse |  |

## EtherCAT operation limits

Due to differences in the performance of CPU types, there are different limits for EtherCAT which are detailed below.

## Configuration minimum EtherCAT cycle time in each PLC type

| PLC Type | PM5072 | PM5082 | PM5630 | PM5650 | PM5670/ <br> PM5675 |
| :--- | :--- | :--- | :--- | :--- | :--- |
| Min. <br> EtherCAT <br> master cycle <br> time configu- <br> rable (ms) | 2 ms | 1 ms | 2 ms | 1 ms | 0.5 ms |

Maximum number of synchronized axes in each PLC type

There are limit on the minimum cycle time for each PLC type and limit on the number of servo drives can be connected for each PLC type based on the cycle time configured. Refer the help file for more details ("PLC Automation with V3 CPUs $\rightarrow$ Libraries and solutions $\rightarrow$ Motion control library $\rightarrow$ Preconditions for the use of the libraries") and adapt the cycle time or PLC Type accordingly to avoid error messages during program download.

| PLC Type | PM5072 | PM5082 | PM5630 | PM5650 | PM5670/ <br> PM5675 |
| :--- | :--- | :--- | :--- | :--- | :--- |
| Number of <br> synchr. axis in <br> 1 ms | - | 4 | - | 8 | 16 |
| Number of <br> synchr. axis in <br> 2 ms | 4 | 8 | 4 | 16 | 32 |
| Number of <br> synchr. axis in <br> 4 ms | 8 | 16 | 8 | 32 | 64 |

These limits are based on the EtherCAT master cycle time configured in the EtherCAT master.
The "Number of axes" is counted in Automation Builder are based on the axis configured, therefore PTO axis also counted in case if configured in the AC500-eCo Pro PLC.

Mor axis can be used by:

- Increasing the EtherCAT cycle time to accommodate a higher "Number of axes" in the same PLC type.
- Choosing a higher PLC type.

The "Motion Solution" window can be used to see how many axes are supported and how many are already used for the particular PLC type for the EtherCAT master cycle time configured by checking the slider at the bottom of the "Motion Solution Wizard" overview page:


### 6.5.10.1.4 Creation a new "Motion Solution Project"

Once the installation process is completed and the elements that will be needed in the configuration are known then the "Motion Solution Project" can be used to allow the user to create a matching configuration.

## Creation a "New Project"

1. Start Automation Builder and select "New Project".
2. Select the "Motion Solution Project" icon.

3. Click the [OK] button and a new project will be created in the specified location with the specified name.

## Addition of a PLC

1. Automation Builder will now pop up a "Select PLC" window and from here user can select one of the V3 PLC's.

$\Rightarrow$
The PLC type can be changed at any time using the steps outlined earlier.
2. Click the [Add PLC] button to get it added to Automation Builder hardware tree.
3. After selecting the PLC type, "Motion Solution Wizard" will guide the user to the "Motion Solution Wizard" page.
4. After creating the hardware tree, Automation Builder will launch the "Motion Solution Wizard" overview page. From here user can add the axis to the application.
Depending on the PLC type selected some of the axis types are not possible for the user to add to the project.


## Addition of the PTO axes

All AC500-eCo PLC's are supported with a maximum of 4 PTO axis via onboard I/O channels.
After creating the hardware tree, Automation Builder will launch the "Motion Solution Wizard" overview page. From here user can add the PTO axis to the application by clicking on the [Add PTO axis] button from the "Motion Solution Wizard" overview page.


Nested underneath the "OnBoard_IO" object is the PTO axis object. From here users can configure each axis they have added separately as per the application requirement by opening the motion axis object which is added under the "OnBoard_IO".


To do this double click on the object to open "Settings" tab.


Here user can update settings as per the application requirement. Later when downloading the application, the wizard will use these settings to define the operation and scaling of this axis object.

## PTO axis settings

All settings related to the application and axis specific will be done here and need to be carefully updated for each axis. Based on the inputs provided here, wizard will compile and generate the code.
"Axis type" The user can select the type of axis to be configured based on the application requirement.

| Axis type <br> - Modulo (rotary)$\quad$ Finite (rotary) | Linear (rotary screw) | O Linear (linear motor) | $\square$ Axis simulation mode |
| :--- | :--- | :--- | :--- | :--- |

Below is the list of settings which user can configure from the "Motion Control Wizard" along with their meanings:

| Axis type | Detail | Max Limit (Counts) |
| :--- | :--- | :--- |
| Modulo (rotary) | Default setting in the wizard. <br> By selecting the "Modulo" <br> (rotary) your axis will be con- <br> figured as a roll-over axis and <br> the desired modulo range can <br> be configured later. | 16\#7FFFFFFF[RB1] |
| Finite (rotary) | Your axis will be configured <br> as a roll-over axis where <br> in modulo range is non <br> editable by the user and <br> calculated based on the <br> "Unit" selection, Inc_Per_R, <br> U_Per_Rev_Nominator and <br> U_Per_Rev_Denominator set- <br> ting. | 16\#40000000 |
| Linear | This needs to be configured <br> when the user having a rotary <br> motor with linear movements <br> (linear axis). | 16\#7FFFFFFFF+/- |
| (rotary screw) | This needs to be configured <br> when the user axis is a linear <br> motor. | 16\#7FFFFFFFF+/- |
| Linear <br> (linear motor) | This option is read only from <br> here and needed when the <br> user configured the axis but <br> not have the real hardware <br> yet. This can be selected from <br> the "Motion Solution Wizard" <br> overview page. <br> Virtual axis configuration. <br> a Chapter 6.5.10.1.4.6 "Addi- <br> tion of a virtual axis" <br> on page 3932 | 16\#7FFFFFFFF |
| Axis simulation mode |  |  |

"PTO " (Hard- With "Revision 1 CPU" or higher, by default 4 * 200 kHz PTO axis is supported, there is no ware settings) special settings needed in the axis configuration page.

| PTO |  |
| :--- | :--- |
| PTO frequency | 200 kHz |

If a "Revision 0 CPU" is used, user can set the PTO axis frequency to 100 kHz or 200 kHz using the drop down as shown below. Based on the PTO axis frequency selection, the onboard I/O configuration will be updated.

| PTO |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| PTO frequency | 200 kHz マ | $\checkmark$ Closed loop | Attention: Non-Revision 1 CPU is selected |  |  |
| Unit | 100 kHz |  |  |  |  |
| pulse | 200 kHz | nm | $\bigcirc$ degree | inch | Orevolution |

"Closed loop" By default, the PTO axis is open loop and however user can also configure a maximum of two PTO axis as closed loop.
To configure the axis as closed loop, user need to configure an encoder axis -> encoder source -> encoder channel -> encoder purpose as the PTO axis which needs to be a closed loop axis.
"Unit" Based on the application requirement user can select the desired unit in the wizard and the wizard will update the subsequent parameters to the selected user unit.
From the below picture user can find the currently supported unit formats.

| Unit | $\bigcirc \mathrm{mm}$ | $\bigcirc \mu \mathrm{m}$ | $\bigcirc \mathrm{nm}$ | $\odot$ degree | $\bigcirc$ inch | $\bigcirc$ revolution |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |

As an example, when the user selects the axis type as "Modulo (rotary)" and unit as degree, the wizard will update the subsequent parameters to the selected user unit and fill with default values, ex: modulo range = 360 degree (default). However make sure the user updates the subsequent parameters as per the actual application requirement.

"Pulses per revolution scaling"

When the axis is open loop, user can update the "Pulses per revolution scaling" with the steps per revolution.


When the PTO axis is open loop, it is important that the user set the steps per revolution and maximum rpm by keeping the PTO frequency limits into consideration.
For example, for a 100 kHz PTO axis if the steps per revolution is 2000, the maximum rpm the axis can support is $3000 \mathrm{rpm}(=2000 * 3000 / 60=100 \mathrm{kHz})$

## "Application gearing"

Based on the actual application requirement, here user can check/uncheck the "Application has gearing" check box. Here the user can also update the required tool travel distance per motor revolution.
When the user unchecks the "Application has gearing", user can update the "Tool travel distance per motor revolution" as per the application requirement as shown in the below picture.
Application gearing
$\square$ Application has gearing
Units per revolution scaling (without gearbox)
Tool travel distance per motor revolution (2)
Reference: Unit conversion formula
Number of pulses (pulse) $=\frac{\text { Encoder increments per revolution (1) }}{\text { Tool travel distance per motor revolution (2) }}$ * Travel distance (in user defined units)

When the user checks the "Application has gearing", user will be prompted to provide the gear box details additionally as shown in the below figure and during the generate application, the wizard will update the same accordingly.

"Modulo range" The user can provide the "Modulo range" here. This is the value at which the axis position will wrap back to zero. This window will be active only when the user selects the axis type as any of the rotary axis.

"Software The user can configure some of the common "Software limits" from the wizard itself. Below is limits" the list of software limits which user can configure from the wizard in the selected application units.

By default, software limits in wizard are not enabled and user need to enable the same by enabling the check box "Enable limits".

| Software limits |  |  |
| :--- | :--- | :--- |
| $\square$ Enable limits |  |  |
| Forward limit (axis stop) | 1000 | degree |
| Reverse limit (axis stop) | 0 | degree |
| Forward limit (warning) | 990 | degree |
| Reverse limit (warning) | 10 | degree |

"Direction
correction"
In some of the application we need to change the relationship between its real direction and that within the PLC program. By default, the check box will be unchecked, and the direction will be normal. By selecting the check box "Invert direction" both actual and reference position will be inverted, and the axis will move in opposite direction.

## Direction correction

| "Position | Here the user can configure the parameters related to position control and supervision. Details <br> control (cyclic <br> of each parameter is explained in our system technology chapter. |
| :--- | :--- |
| sync mode)" | « Chapter 6.5.10.3.5.6.3 "Position control loop" on page 4028 |
|  | " Chapter 6.5.10.3.5.6.2 "Supervision" on page 4027 |


| Position control (cyclic sync mode) |  |
| :--- | :--- |
| Control time | 100 |
| Feed forward percentage $(0-100 \%)$ | 25 |
| Following error percentage $(0-300 \%)$ | 200 |
| Delay time velocity check | 10000 |
| Position lag supervision | Deactivated $\sim$ |

"Dynamic limits"

Users can update the maximum limits here. Some parameters depend on the drive settings and needs to be set correctly to get the desired result.
The user can set the "Maximum application velocity" to a desired value to limit the maximum application speed.

| Dynamic limits |  |
| :--- | :--- |
| Maximum application velocity  <br>   <br> Maximum acceleration degree $/ \mathrm{sec}$ <br> Maximum deceleration 10000 <br> Maximum jerk 10000 <br>  degree $/ \mathrm{sec}^{2}$ | 100000 |

## "Drive based limits"

Maximum reference value is the maximum frequency of the drive in which it reaches the maximum rpm. In some drives, the calculated maximum reference value may need to be modified, and user can select "Modify" box and change the value. Note, we have the hardware limitation as 100 kHz or 200 kHz based on the configuration and user must make sure this limitation is not crossed.

Maximum speed is recommended to keep the same maximum speed at the drive and at the PLC parameter.

| Drive based limits |  |  |
| :--- | :--- | :--- | :--- |
| Maximum reference value | 200000 | $\square$ Modify |
| Maximum speed (user defined) | 6000 | rpm Maximum speed allowed with this configuration is: 6000 |

Currently the applications torque limits in the wizard are not valid for PTO axis and this is ignored.

When the PTO axis is open loop, it is important that the user set the steps per revolution and maximum rpm by keeping the PTO frequency limits into consideration.

For example, for a 100 kHz PTO axis if the steps per revolution is 2000, the maximum rpm the axis can support is $3000 \mathrm{rpm}(=2000 * 3000 / 60=100 \mathrm{kHz})$

When in closed loop, set the maximum rpm for the axis to reach when the maximum PTO axis frequency is provided.

For example, if the axis is configured as 200 kHz and closed loop, set the maximum rpm for the axis to reach when the 200 kHz frequency is achieved.
"Results Based on the inputs provided, wizard will calculate the results and can be viewed immediately at (calculated)" the end of the configuration page.

| Results (calculated) |  |  |
| :--- | :--- | :--- |
| Position resolution | 364.0889 | pulses/ degree |
| Maximum possible velocity | 36000 | degree $/ \mathrm{sec}$ |
| Maximum allowed following error | 3600 | degree |

## Addition of an EtherCAT axes

From Automation Builder 2.7.0, users will have two options to realize an EtherCAT network; Using the AC500 and CM579-ETH solution (as we have supported before) and the new option of use AC500-eCo PM5072 and PM5082 PLC's (EtherCAT master using onboard ETH1 port.)

## Differences of the AC500 and AC500-eCo wizard views

After creating the hardware tree the user has selected the CPU. Automation Builder will launch the "Motion Solution Wizard" overview page. As we can see below the user will see a slightly different view based on their selection.

## AC500 CPU adding the EtherCAT master via CM579-ETHCAT

To start adding devices click the [Add CM579-ETHCAT axis] button.


AC500-eCo Pro CPU's adding the EtherCAT master onboard ETH1 port for AC500-eCo CPU's

1. To start adding devices click "Add onboard EtherCAT axis".

2. Adding EtherCAT devices to the master.

Here user find any of EtherCAT drives or devices needed for the application. The user can also add the axis by right clicking on the "EtherCAT_Master" object and select the "Add Object "option or by using the [Add axis] button from "Motion Solution Wizard" overview page.


If user wants to add multiple servo drives, please uncheck the "Close this dialog after each transaction" check box (in left bottom corner) and add multiple drives without closing. In AC500 CPU user can add the EtherCAT motion axis only on the coupler slot 1.

## Configuration of the EtherCAT master

As mentioned in the beginning of the chapter, user can now configure EtherCAT master in AC500 CPU via CM579-ETHCAT communication module or in AC500-eCo Pro PLC's via onboard ETH1 port.

In general, all EtherCAT settings are similar in both solutions but in CM579-ETHCAT user will have more settings tab than onboard EtherCAT. Based on which EtherCAT solution is used, user needs to also use the relevant libraries for diagnosis/SDO handling etc.
For CM579-ETHCAT use the library "AC500_EtherCAT" and for onboard EtherCAT use the library "ETCStack". Both solutions are not identical, and user need to manually adapt the applications incase if moving from CM579-ETHCAT to onboard EtherCAT or vice versa.

CM579-ETHCAT These are the settings specific to CM579-ETHCAT master and not valid for onboard EtherCAT. The label will normally CM579_ETHCAT this can be changed by the user but is normally left as default. If so, the name will be CM579_ETHCAT (CM579-ETHCAT).


In most cases these settings can be left at default but occasionally the user might need to change these to fit the application requirements.

- "Run on Config Fault " - As default it is set to "Yes" meaning the CM579-ETHCAT will not go into error and the PLC program will not go into stop if a slave is lost, this is beneficial as the PLC is then still available to handle other follow up actions. This can be changed if the user wants a configuration where a missing slave will prevent the PLC running.
- "Distributed clocks" - As default it is set to "Active" which is a must if you want to use syncronised motion over EtherCAT.
- "Optimize I/O update" - As default it is set to "On" meaning that, consecutive I/O's are merged in one block to optimize the performance.

ETHCAT_Master This second part of the EtherCAT master defines the specific settings that define the behavior (EtherCĀTMaster) of the EtherCAT operation and from here settings are similar in both CM579-ETHCAT and in onboard EtherCAT.


Onboard EtherCAT is only possible to configure it on PM5072 and PM5082 ETH1 port. It is recommended not to configure any other protocols on the EtherCAT configured ETH port.

The label will normally be in the format of '[name](ETHCAT-Master)' this can be changed by the user but is normally left as default. If so, the name will be ETHCAT_Master (ETHCAT-Master). The most important settings for this are mentioned below:

- "Auto-config master/slaves" - As Default this is activated, this means the parameters are set automatically here in accordance with the default settings. This setting is recommended unless the user is very familiar with the setup of EtherCAT networks.
- "Cycle time" - The default EtherCAT cycle time is $4000 \mu$ s but based on the application requirement, user can adapt the EtherCAT cycle time as as shown below.


EtherCAT cycle time will directly influence the PLC load. If your PLC load is higher than desired, please increase the cycle time or upgrade the PLC type.

## Configuration of the EtherCAT slave and axis

This section explains how to configure the behavior of the Comms slave module. From here we can define how the hardware will behave. It's important to understand the EtherCAT slave axis once added is split into two parts in the project tree. These two parts are described below.

## EtherCAT slave drive object settings

The label will normally be in the format of [Drive name](Drive Type) this can be changed by the user but is normally left as default. If so, the name could be "MicroFlex_e190" (MicroFlex-e190).

In most cases these settings can be left at default but occasionally the user might need to change these to fit the application requirements. Useful settings may be to enable "Expert Settings" to add additional PDO mappings, to check and set mappings or to check the status of the device once online.

## PDO mappings

## PDO and startup parameters (SDO)

Based on the selection under "Motion axis $\rightarrow$ Mapping", PDO and startup parameters are generated automatically as shown below.

Based on the control type and PDO mapping selected in the wizard, it will update the "Process Data" tab from the slave axis object and assign the autogenerated name to each object which is added.

(17) Microflex_e190 x

| General | $\vdots り$ ¢ $\times$ Clear mappings ${ }^{\text {¢ }} \mathbf{Y}$ |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | Object Name | Variable | Channel | Address | Type | Description |
| Process Data | MicroFlex_e 190 | MicroFlex_e190_ControlWord | AX0_ControlWord_U16 | \%QW2 | UINT | AX0_ControlWord_U16 |
| Startup Parameters | MicroFlex_e190 | Microflex_e190_TargetPosition | AX0_TargetPosition_I32 | \%QD2 | DINT | AX0_TargetPosition_I32 |
| Starup Paraters | Microflex_e190 | MicroFlex_e190_StatusWord | AX0_StatusWord_U16 | \%IW2 | UINT | AX0_StatusWord_U16 |
| EoE Settings | MicroFlex_e190 | MicroFlex_e190_ActualPosition | AXO_ActualPosition_132 | \%ID2 | DINT | AXO_ActualPosition_132 |
| Log |  |  |  |  |  |  |
| EtherCAT I/O Mapping |  |  |  |  |  |  |
| EtherCAT IEC Objects |  |  |  |  |  |  |
| I/O mapping list |  |  |  |  |  |  |

Like PDO mapping, SDO startup will be updated based on the control type and the mapping selected in the wizard.


Based on the application requirement, user can add more PDO/startup parameters manually.

## User defintion of PDO mappings

If the user wants to add additional PDO mapped objects first enable the "Expert settings", then the user will get an additional tab called "Expert Process Data" below the general tab and here user can add/edit/delete the mapping.


## EtherCAT axis settings

Nested underneath the EtherCAT slave drive object is the axis object, from here users can configure each axis as per the application requirement.


To open this double-click on the object to open the configuration page which then has two separate tabs, one for "Settings" and the other one for "Mapping".

The user can update both the pages as per the application requirement.


Later during the "Generate application" process, the wizard will use these settings to define the operation and scaling of this axis object.

EtherCAT axis object settings

All settings related to the application and axis specific will be done here and need to be carefully updated for each axis. Based on the inputs provided here, wizard will compile and generate the code.
"Axis type" The user can select the type of axis to be configured based on the application requirement.

| Axis type | Linear (rotary screw) $\quad$ Linear (linear motor) (rotary) $\quad \square$ Axis simulation mode |
| :---: | :---: | :---: |

Below is the list of settings which user can configure from the "Motion Control Wizard" along with their meanings:

| Axis type | Detail | Max Limit (Counts) |
| :---: | :---: | :---: |
| Modulo (rotary) | Default setting in the wizard. <br> By selecting the "Modulo" (rotary) your axis will be configured as a roll-over axis and the desired modulo range can be configured later. | 16\#7FFFFFFF[RB1] |
| Finite (rotary) | Your axis will be configured as a roll-over axis where in modulo range is non editable by the user and calculated based on the "Unit" selection, Inc_Per_R, U_Per_Rev_Nominator and U_Per_Rev_Denominator setting. | 16\#40000000 |
| Linear <br> (rotary screw) | This needs to be configured when the user having a rotary motor with linear movements (linear axis). | 16\#7FFFFFFF+/- |
| Linear <br> (linear motor) | This needs to be configured when the user axis is a linear motor. | 16\#7FFFFFFF+/- |
| Axis simulation mode | This option is read only from here and needed when the user configured the axis but not have the real hardware yet. This can be selected from the "Motion Solution Wizard" overview page. <br> Virtual axis configuration. <br> Chapter 6.5.10.1.4.6 "Addition of a virtual axis" on page 3932 | 16\#7FFFFFFF |

"Unit" Based on the application requirement user can select the desired unit in the wizard and the wizard will update the subsequent parameters to the selected user unit.
From the below picture user can find the currently supported unit formats.
$\square$
As an example, when the user selects the axis type as "Modulo (rotary)" and unit as degree, the wizard will update the subsequent parameters to the selected user unit and fill with default values, ex: modulo range = 360 degree (default). However make sure the user updates the subsequent parameters as per the actual application requirement.

For rotary axis the units $m m, \mu m, n m$, and inch might lead to inaccuracy due to rounding errors.
"Pulses per revolution scaling"

User can update the "Pulses per revolution scaling" with the actual encoder increments per motor revolution.

## Pulses per revolution scaling

Encoder increments per motor revolution (1) 131072 pulses/revolution

## "Application gearing/units per revolution"

Here the user can set the tool travel distance per motor revolution and define the gearing relationship with the mechanical system. As default the "Application has gearing" box is unchecked, this means the application has no gearing and the user can update the "Tool travel distance per motor revolution" as per the application requirement:
Application gearing
$\square$ Application has gearing
Units per revolution scaling (without gearbox)

| Tool travel distance per motor revolution (2) |
| :--- |
| Reference: Unit conversion formula |
| Number of pulses (pulse) $=\frac{\text { Encoder increments per revolution (1) }}{\text { Tool travel distance per motor revolution (2) }} *$ Travel distance (in user defined units) | 360

If the application has gearing the user can check the "Application has gearing" box and they will be prompted to provide the gearing details. During the generate application, the wizard will update the same accordingly.

"Modulo range" The user can provide the "Modulo range" here. This is the value at which the axis position will wrap back to zero. This window will be active only when the user selects the axis type as any of the rotary axis.


## "Software limits"

The user can configure some of the common "Software limits" from the wizard itself. Below is the list of software limits which user can configure from the wizard in the selected application units.

By default, software limits in wizard are not enabled and user need to enable the same by enabling the check box "Enable limits".

| Software limits |  |
| :--- | :--- |
| $\square$  <br> Enable limits 1000 <br> Forward limit (axis stop) 0 <br> Reverse limit (axis stop) 990 <br> Forward limit (warning) 10 <br> Reverse limit (warning) degree |  |

"Direction and Homing type"
" Invert direction" - This needs to be set for the application which needs to change the axis relationship between its real direction and that within the PLC program. By default, the check box will be unchecked, and the direction will be normal. By selecting the check box "Invert direction" both actual and reference position will be inverted, and the axis will move in opposite direction.

"Homing using DRIVE IO Touch Probe" - Selecting this will allow the axis to home to a drive based touch probe using EtherCAT.

Selecting this also by default make the PDO mapping preselected for the user, but this can be later changed based on the application need.

```
Additional PDO mapping
    Touch probe 1
    |ising Edge (Pos)
    Falling Edge (Neg)
    Touch probe 2
    \square \text { Rising Edge (Pos)}
    \square \text { Falling Edge (Neg)}
```

Here the user can configure the parameters related to position control and supervision. Details of each parameter is explained in our system technology chapter.
② Chapter 6.5.10.3.5.6.3 "Position control loop" on page 4028
« Chapter 6.5.10.3.5.6.2 "Supervision" on page 4027

| Position control (cyclic sync mode) |  |
| :--- | :--- |
| Control time | 100 |
| Feed forward percentage $(0-100 \%)$ | 25 |
| Following error percentage $(0-300 \%)$ | 200 |
| Delay time velocity check | 10000 |
| Position lag supervision | Deactivated $\sim$ |

[^51]| Dynamic limits |  |
| :--- | :--- |
| Maximum application velocity | 36000 |
| Maximum acceleration | 10000 |
| Maximum deceleration | 10000 |
| Maximum jerk | degree $/ \mathrm{sec}$ |

Servo drive Example, if user is using an ABB servo drive with, 131072 encoder increments per revolution and a maximum speed is 6000 rpm .
Maximum application velocity = Max application velocity in RPM * Tool travel distance per revolution $/ 60$ * Gearbox nominator / gearbox denominator $=6000$ * $360 / 60$ * $1 / 1$
$=36000$ degree $/$ sec
For easy calculation of parameters user can use the excel "AC500_V3_MotionControl_Startup guide for MC parameterization.xIsx" from example program folder PS5611-Motion.

> Use cases and how to handle the function blocks of a certain library is described in sample projects. After the respective library has been installed the corresponding sample projects are available in the default path AutomationBuilder Examples. If the default path is inaccessible, click "Help $\rightarrow$ Project examples" in the Automation Builder menu.
"Drive based limits"

Here the user can define the limits that will dictate the expected behavior of the drive.

| Drive based limits |  |  |
| :--- | :--- | :--- |
| Maximum speed (user defined) | 6000 | rpm |
| Maximum positive torque | 300 | $\%$ |
| Maximum negative torque | -300 |  |

It is recommended to keep the same "Maximum speed" at the drive and at the PLC parameter.
Currently users can define the applications torque limits in the wizard and they will be written to the SDO startup parameter only if the user selects "Torque limits" in the "Mapping" page. These parameters are not used in the program by default.

```
SDO startup parameter mapping
    Give EtherCAT control
    V Operating mode
    V Torque limits
```

"Results Based on the inputs provided, wizard will calculate the results and can be viewed immediately at (calculated)" the end of the configuration page.

| Results (calculated) |  |  |
| :--- | :--- | :--- |
| Position resolution | 364.0889 | pulses/ degree |
| Maximum possible velocity | 36000 | degree $/ \mathrm{sec}$ |
| Maximum allowed following error | 3600 | degree |

## Mapping (Axis control type and object mapping)

The "Mapping" and "Control Type" tab can be selected if the user wants to set a "Control type" other than CSP (default) and mappings other than default (Control Word, Set Position, Status Word, Actual Position). It can be found under the axis object here:


Control type By default, wizard is selected for "Cyclic synchronous position mode (CSP)". User can change the same based on the application requirement. The supported control modes are:
Cyclic synchronous position mode (CSP).
Cyclic synchronous velocity mode (CSV).
Cyclic synchronous velocity mode for load control (CSVL).

CSVL is an ABB specific mode to achieve load control/profiling. By using this mode, the user can use the "Motion Control Load library" which is implemented based on the "PLCopen Motion Part 6 - Fluid Power Extensions". For more details on load/torque control please refer to the library integrated documentation, system technology in online help file and the example program / description from example program folder.

Additional PDO
mapping

If the application needs "Additional PDO mapping" the wizard helps the user to add most used PDO mapping just by selecting them here.

Based on the control type selected, a few of the mandatory PDO mapping are generated automatically and from additional mapping area in wizard user can find the most common PDO mapping and user can add the same based on the application requirement.

The user can add additional PDO mapping which are not listed here manually by enabling the expert settings from the slave device general configuration page (as described earlier).

SDO start-up By default, two of the SDO startup parameters are selected and it is recommended not to parameter map- change these unless the user has expert level knowledge of DS 402 control modes or intends to ping

```
SDO startup parameter mapping
    \ Give EtherCAT control
    \ Operating mode
    \square \text { Torque limits}
```

The user can select the "Torque limits" and the torque values set from the "Settings" page will be written to the respective slave drives startup parameters list.

| Drive based limits |  |  |
| :--- | :--- | :--- |
| Maximum speed (user defined) | 6000 | rpm |
| Maximum positive torque | 300 | $\%$ |
| Maximum negative torque | -300 |  |

Once these settings are made and the generated code is executed you can see that these settings have change the drives EtherCAT slave configuration as shown in the picture below:

| General <br> Process Data |  | \% Add IEdt $X$ Delete \& Move Up \& Move Down |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | Line | Index:Subindex | Name | Value |
|  |  | 1 | 16a5002:16a00 | Give EtherCAT Control | 1 |
| Sartup Parameters | POU_2 (PRG) | $-2$ | $16=6060: 16=00$ | ModesOfOperation = 8 (Cydic Synchronous Position Mode) | 8 |
|  |  | $-3$ | 16a5023:16=00 | Maximum Torque Limit | 3000 |
| EoE Settings |  | 4 | $16=5022: 16=00$ | Mrinum Torque Limit | -3000 |

## Addition of an encoder axis

For any application which needs an encoder, an encoder axis can be added and configured. This can be added either by opening the "Motion Solution Wizard" object and clicking on the [Add encoder axis] button on the bottom of the page as shown below or by right clicking on the "Motion Solution Wizard" and selecting "Add object".


Fig. 310: SC_2023_AB_Motion_wiring_appl_progr
After adding an encoder axis, the user can find it under the "Motion Solution Wizard" in the Automation Builder. Users can double click on the added encoder axis object to get the settings page and configure it as per the requirement. Settings here is like any other motion axis expect here user need to configure encoder source.
When an encoder axis is added, user must configure the encoder source channel and encoder source.

## Selection of the encoder source

For a user to use an encoder they must know where the data source that supports it comes from.


Below are the options for the encoder source:

- "Drive encoder" (EtherCAT) - When there is an EtherCAT axis, the user will have a chance to use the EtherCAT servo drives, master encoder (16\#400C:03) as a source for the encoder axis (which will be scanned at EtherCAT cyclic rate). This can only be used as a 'master encoder".
- PLC encoder (OBIO) - When an AC500-eCo PLC is used there is an option to use the onboard IO to take in a two channel Automation Builder encoder signal using the 24 V digital inputs. This can be used as a 'master encoder'* or to close the position loop to a PTO drive in a PTO axis is configured.
- "Data source" - here the user can select any data source from which the value of the encoder can be driven. This requires a data area or variable configured within the program.
* This can now be used to latch positions, or as a master for another axis to follow.


## Using an EtherCAT drive based encoder source

1. Go to the desired EtherCAT drives axis object.
2. Open the "Mapping" tab.
3. Tick the box for "Master encoder".

$\Rightarrow$ Now the configured "Master encoder" is ready to be selected. This selection will make the encoder object 16\#400C:03 available as a source for the encoder axis (which will be scanned at EtherCAT cyclic rate).
4. Add or select the desired encoder axis.
5. Select "Encoder source channel" as "Drive encoder" and the "Encoder source" via the name of the EtherCAT drive's name which the encoder will be selected.


## Using a PLC (OBIO) based encoder source (AC500-eCo only)

When there is a PTO axis, user will have an option to use the onboard encoder channels from AC500-eCo PLC as a source for the encoder axis.

User can add or select the desired encoder axis and select "Encoder source channel" as "PLC encoder OBIO" and the "Encoder source" via the channel (0 or 1). In addition, the purpose should be defined, either "Master encoder" (This can now be used to latch positions, or as a master for another axis to follow) or "Close pos loop" to close the position loop for a PTO axis:


Filtering is also available should the encoder data require it.

## Data source

Here the user can use a "Data source" to define a variable which will then create the variable in the mentioned name as data type "DINT".


Alternatively they can connect to an existing variable from the project to encoder axis. The variable must be of data type "DINT".

| Encoder |  |
| :--- | :--- |
| Encoder source channel |  |
| Encoder source | Data source $\quad$. |

Other encoder axis parameters are same as EtherCAT or PTO axis and for details please refer to the previous chapters.

## Addition of a virtual axis

If the applications require a virtual axis (one which has no associated hardware) then the user can add this.

Eeither by and .

1. Opening the "Motion Solution Wizard" object (by doubling clicking on it).
2. Clicking on the [Add virtual axis] button on the bottom of the page.

3. Open the"Motion Solution Wizard" object by doubling clicking on it.
4. Click on the [Add virtual axis] button on the bottom of the page.

5. Alternatively, right click on the "Motion Solution Wizard "object and by selecting "Add object" option to add the axis.
6. After adding a virtual axis, the user can find it object under the "Motion Solution Wizard ". Users can double click on the added virtual axis object to access the settings page and configure it.

$\Rightarrow$ Settings here are the same as the motion axis and details for how to configure the parameters, refer to the previous chapters.

## Important general PLC settings

Compared with a standard AC500 project, when the user is using motion solution wizard, some default settings are changed during code Generation. These settings are always overwritten as long as there is a change in axis configuration or added a new axis. Below are the settings which are updated by the "Motion Solution Wizard":

|  | Project type |  |
| :--- | :--- | :--- |
| Setting | EtherCAT | PTO Only |
| IO bus - Run on congif fault | Yes | Yes |
| PLC > PLC Settings > Always <br> update variables | "Disabled" | "Disabled" |
| PLC > PLC Settings > Bus <br> Cycle Options | e.g. "Task" - any real slow task must be selected and it <br> shouldn't be set as "unspecified" as this can lead to a negative <br> on cyclic operation. |  |
| CPU-Parameters > Check <br> battery | Off | NA |
| CM579-ETHERCAT > Run on <br> congif fault | Yes | NA |
| CM579-ETHERCAT > Bus <br> Behaviour | Sync Mode 1 | NA |

It is recommended that user change the above settings manually based on the actual application requirement. To update these setting manually, user need to change the setting after all the axis changes are done and then generate the code.

## Generation the motion configuration

Once all the configuration is completed, the user can generated the application (like compile) which will then update; its settings, the PDO and SDO mapping, and the motion task configuration automatically based on the settings and parameters provided in the wizard. To do this, go to "Build" menu and click on "Generate Code"

Automation Builder will generate the axis configuration. Once completed successfully, Automation Builder will show the message "Motion Solution Generation successful" in the message window. This can take some time based on the number of axes configured in the project and PDO mapping selection.

## Task configuration

## Tasks within the Motion Wizard

The wizard will automatically create a fast task and a slow task to help guide the user to correctly distribute their code. The kernel function block and the transfer of axis IO data with EtherCAT or other hardware should be processed in a faster task. This task should be as short and real time (if EtherCAT) or as fast possible (if PTO) to achieve the best motion control performance.
In case the position reference is transferred to the drive the task of the axis implementation should be synchronized to the fieldbus cycle. The following figures show an example for EtherCAT.


The Cyclic task is show as below:


## MotionSolution_Task

The "Motion Solution Wizard" will automatically create a fast task called "MotionSolution_Task". This will call all the axis operation code, and will be configured dependently on the configuration.

## MotionSolution_Task in EtherCAT applications

In the case of EtherCAT applications synchronized the setting of "Cycle time" is defined in the EtherCAT master.


Once the solution is generated this will then define a task which has a type of external and a cycle time which matches that configured in the EtherCAT master and the Priority is set as " 0 ".

```
Confguraton
Prioity ( 0.16 ): 0
Type
#xtemal Extemal vent CoupleEvivent1
\squareEnable
```


## MotionSolution_Task in PTO only applications

In the case of PTO only applications, set the cycle time in the "Motion Solution Wizard".
Once the solution is generated this will then define a task which has a "Type" of "Cyclic" and a cycle time which matches that configured in the solution wizard and the priority is set as "10".


## Function blocks must be called in the Motion Solution Task

As mentioned previously the "MotionSolution_Task" is the fastest and highest priority task in the CPU and so of course the most draining on resources. It is possible for the user to call parts of the motion application in this "MotionSolution_Task" - however excessive amounts of code or that with a poor structure in this task can lead to overloading of the CPU.

Despite these limitations some function blocks must be called in the same task as the kernel function block ("MotionSolution_Task").

- MC_CombineAxes
- MCA_MoveByExternalReference
- MCA_MoveByExtRefRelative
- MCA_DigitalCamSwitch
- MCA_GetTappetValue
- ECAT_CiA402 Touchprobe_App
- ECAT_HomingOnTouchprobe
- ECAT_CiA402ParameterHoming


For onboard EtherCAT motion axis projects, it is recommended to call all the EtherCAT function blocks in one task, for coupler EtherCAT this is not relevant.

## Cyclic tasks and created task "Task"

As default the wizard will also create a 10 ms cyclic task called "Task" which the user can use for such code calling for noncritical motion function blocks:


## Guidance for code distribution within tasks

It is recommended for most non-real time motion PLCopen function blocks in the program to use a different cyclic task with a cycle time at least double of that used in the "MotionSolution_Task" and a lower priority (such as 'Task'). This will reserve more capacity for better performance on the EtherCAT by the PLC.
To save PLC processing time the most PLCopen function blocks as well as the application logic can also be processed in a task which runs on a lower priority than the real time task with the axis implementation as shown in the figure below.


## Motion solution libraries

Based on the control type selected, mandatory libraries are added to the Library Manager. The libraries added depend on the control type selected. See the table below for the libraries added based on the control type selected.
$\left.\begin{array}{|l|l|l|l|l|}\hline \text { Control Type } & \text { Library } & \begin{array}{l}\text { AC500_Ecat_CiA } \\ 402\end{array} & \begin{array}{l}\text { AC500_Motion- } \\ \text { Control }\end{array} & \begin{array}{l}\text { AC500_Motion- } \\ \text { ControILoad }\end{array}\end{array} \begin{array}{l}\text { AC500_Motion- } \\ \text { ControIEco }\end{array}\right]$.

For example, if the user selects control type as CSVL we can see the libraries indicated in the Library Manager.



## Axis program generated (Hidden by default)

The code which is generated for each axis is hidden by default (using the tick box in the "Motion Solution Wizard') but users can choose to make it visible if needed.

To view the generated axis program, the user must access the "Motion Solution Wizard" window.


Then navigate to the bottom and uncheck the [Hide generated code] button from the "Motion Solution Wizard"overview page.


Once the user has unchecked [Hide generated code], they can find the folder "MotionSolution_Generated", which has sub folders for each of the axis configured using the wizard. Each axes subfolder has a GVL and a PRG respectively. Name of the folder GVL/PRG/ Function Blocks/variable are generated based on the axis object name.

Below is the example from EtherCAT axis, for other axis it follows the similar structure.


It is not recommended to change the program manually without exert knowledge as any changes which will be made later the axis object will overwrite all the changes done by the user if the "Generate" check box is checked.

## Axis parameters generated (GVL)

The GVL generated by the wizard maps the parameters set by the wizard to the library variables in the Axis_Ref structure, and the Axis_IO declaration is initiated here.

```
VAR GLOBAL
    MicroFlex_e190_Kernel : ONC_Basic_Kernel; ('Axis reference for MicroFlex_e190_Ker
    MicroFlex_e190_Ecat : ECAT_Cla402_Control_App; ('Axis reference for MicroFlex_el90
    MicroFlex_e190_axis : AC500_MotionControl.Axis_Ref := (
        expert:=1
        ECAT := MicroFlex_e190_Ecat,
            OperatingMode := B,
            Kernel i= MicroFlex_e190_Kernel
            ).
            paramecer:=(
            paraReverseDirection i= 0, //0 = nomal direction, 1 = reverse imput f
            paraEnablePosLagMonitoring := TRUE, //Position Lag super vision
            paraMaxVelocityAppl I= 36000, //Naximum application velocity
            paraMaxAcceleracionSystem :m 10000, //Maximum acceleration
            paraMaxDeceleracionSystem i= 10000, //Naximum deceleration
            paraMaxJerk := 2000 //Maximu= Jerk
            |): ('Axis class for MicroFlex_e190_axis')
    MicroFlex_e190_axis_ReadPos : AC500_MotionControl.MC_ReadactualPosition;
    MicroFlex_e190_axis_ReadVel : MC500_MocionControl.MC_ReadActualVelocity;
END_VAR
```

It is recommended not to add/edit the program here manually since this will be lost next time when the user click on the [Generate Code].

## Generation a program (PRG)

Program which is generated by the wizard initiates the CMC_Axis_Control_Parameter, CMC_Basic_Kernel/CMC_Load_Motion_Kernel and ECAT_CiA402_Control_App func-


For CSP and CSV control type, CMC_Basic_Kernel function block will be called and for CSVL control type CMC_Load_Motion_Kernel function block will be called.


It is recommended not to add/edit the program here manually since this will be lost if there was any modification done on the particular axis reference.

### 6.5.10.1.5 Wiring application program

It is recommended to the user to write the application by creating new POU's and adding the POU's to a new task and setting a cycle time and task priority lower than "MotionSolution_Task". Dependent on the code writing guidelines mentioned earlier.

An exception to the above general guideline is for the function blocks which needs to be called in the same task where the axis is running and, in such cases, user need to call those function blocks in a separate POU add call the POU in "MotionSolution_Task". If a function block needs to be called inside the real time task ("MotionSolution_Task"), this is mentioned in the function block help file.


An example of a function blocks which needs to be called in the "MotionSolution_Task" is MCA_MoveByExternalReference. To use this in a program, a new POU with this (and any other real time function blocks) has been created called "MyMotion".


This POU can then be called from the "MotionSolution_Task" by adding it as shown below.

| Projects <br> PLC_AC500_V3 (PM5650-2ETH - TB5620-2ETH) <br> Moton Solution Wizard <br> - 3. 3 PLCLogic <br> - Application <br> $+\square$ Diagnosis <br> * MotionSolution_Generated <br> Lbrary Manager <br> 0) MyMotion (PRG) <br> - PLC_PRG (PRG) <br> = Task Configuration <br> - \& MotonSolution_Task <br> (1) PRG_Mcroflex_e 190_Axis [1] MyMoVion <br> Task <br> \#1) PLC_PRG |
| :---: |

### 6.5.10.2 Cams and Tappet tables

### 6.5.10.2.1 Definition of a "Cam table"

A "Cam table" is an object type containing a tool which allows the user to create and edit Cams and tappets either graphically or by means of tables. After the code is generated for the corresponding application, there elements or tables are transposed into global data structures or "Cam Data" which the IEC program can access.

## Definition of a Cam

An electronic Cam describes the functional dependency of the position one axis (slave) to another axis (master). The relationship is described by a continuous function (or curve) that maps a defined range of master actual position values to slave target position values.

A software Cam is created by breaking the motion into several segments. Each segment defines how far the slave axis will travel for a given move distance on the master axis. Once these segments have been defined, a path between them is produced, the interpolation method being set by the user. The user controls the size and accuracy of the segments. Using many small segments will produce smoother motion.

Motion across each segment and across segment boundaries will go through all specified points. The segment information for a Cam profile is defined in an array called a table. A table of positions must be defined together with an optional table of master increments.

## Definition of a Tappet

An electronic Tappet describes a function which allows the user to derive a digital state from an axis or cam position which can be user to trigger an event or indicate a state at which an action should occur. You may add tappets (binary switches) to the cam at any position. In this way, you can create cam tables which contain tappets.
6.5.10.2.2 Cam table

Cam table types
With Automation Builder 2.7.0 user has two different Cam objects available in add object option as shown below. From the context menu: "Cam table ABB" and "Cam table ABB basic".


## Differences in the objects

The difference between these objects is the code that is generated by them and the libraries they are compatible with. By hovering over them the user can see the guidance for when to use them:
"Cam table ABB"

- This allows the user to get additional data via Cam structure.
- In the project the user need to use "MCA_CamStructureSelect" (and not "MC_CamTableSelect").
- Compatible with Automation Builder 2.6.1 or higher.

"Cam table ABB basic"
- This gives the user only basic information from the Cam.
- In the project the user need to use "MC_CamTableSelect" as in the past.
- Compatible with Automation Builder 2.6.0 or lower.



## Differences in using the two "Cam tables" types in application code

To use the "Cam table" the user needs to call at least the "MCA_CamStructureSelect" (new) or "MC_CamTableSelect" (old/basic) depending on which version of the "Cam table" objects that are used. Also, in either case you will also need "CamIn" or "CamInDirect" function blocks from the AC500_MotionControl library.

If you use the wrong function block for the wrong type of Cam you can expect your "MC_CamIn/MC_CamInDirect" to never go into 'InSync' = TRUE.

### 6.5.10.2.3 Cam table object structure

The "Cam table" is split into individual editors, which allow the users to edit and add elements to the selected views.

The editor consists of the following tabs:

- [Cam]: Here, you can display and modify the Graphs showing slave position, slave velocity, slave accel-eration, and slave jerk. This view can show the user when you program a movement with high acceleration.
- [Cam table]: Here, you can display and modify the slave position, slave velocity, slave acceleration, and slave jerk in a tabular view.
- [Tappets]: Is an editor for programming tappets (switch points) in a diagram. This display provides a very good overview of the sequential order of the tappets.
- [Tappet table]: Is an editor which lists switch points in a table. Here, you can specify the switch points.


### 6.5.10.2.4 Cam tab

In the Cam graphical editor, the Cam graphs are shown and can be interacted with.
The editor window displays the curves of four graphs:

- Slave position (black)
- Slave velocity (blue)
- Slave acceleration (green)
- Slave jerk (yellow)


The horizontal axis of all four coordinate systems shows the range of the master values ([Minimum - Maximum]). The vertical axis in the position diagram shows the value range that is defined in the Cam properties. The vertical axis of velocity, acceleration, and jerk is scaled automatically.
It is possible to modify all curves, except the jerk curve. As velocity, acceleration, and jerk are derived curves, changes to one graph causes changes to the other graphs.

Switch between the graphical editor and the alternative tabular editor at any time ("Cam table" tab).

### 6.5.10.2.5 Cam table tab

The "Cam table" is an alternative to the graphical editor for defining the Cam graphs ("Cam" tab). The first line of the table always contains the start position of the master (and the related slave values) and the last line is always the end position. The lines in-between alternately define segments and points.
Below is a list of key elements and their descriptions:

| $\oplus$ | Inserts a new line. |
| :--- | :--- |
| 畐 | Deletes the selected segment |
| X | X-position of the slave axis |
| Y | Y-position of the slave axis |
| V | Velocity of the slave axis |
| A | Acceleration of the slave axis |
| J | Jerk of the slave axis |
| Segment type | $\bullet$ Poly5: 5th degree polynomial <br> $\bullet$ |

These elements are always shown in every "Cam" configuration:

The following values result from the values of the respective segment. They cannot be modified.

| $\min$ (Position) | Minimum value of the slave position |
| :--- | :--- |
| $\max$ (Position) | Maximum value of the slave position |
| $\max$ (Velocity) | Maximum value of the velocity of the slave, based on the master axis |
| $\max$ (Acceleration) | Maximum value of the acceleration of the slave, based on the master <br> axis |

Table 745: View "ToolBox"

| $\star$ Select | Select a line in the table by using this tool. <br> Selected points are deleted by pressing the [Del] key. |
| :--- | :--- |
| $\oplus$ Add point | Add new points with this tool. Click the insertion point in the diagram. <br> Te graph is then adapted automatically so that its curve runs through <br> the new inserted point. |

Switch between the table editor and the graphical editor at any time ("Cam" tab).

### 6.5.10.2.6 Tappets tab

The tappet behaviors are shown and can be defined in this graphical editor. A tappet path defines one or more tappets depending on the master position. At the upper edge of the editor window, a horizontal axis approaches the range of the master positions. The individual tappet paths are defined below.
You can switch between the graphical editor and the alternative tabular editor at any time ("Tappet table" tab).

| 1 형 | Track ID of the tappet path． <br> All tappets of a tappet path refer to the same tappet switch（a variable <br> of type BOOL）． |
| :--- | :--- |

Table 746：View＂ToolBox＂

| A Select | Select the tappets by means of this tool．You can drag the selected <br> tappets to another position． <br> You can modify the switch on／off attribute of a tappet by clicking the <br> relevant end of the crossed line $(\Sigma)$. <br> Delete the selected tappet by pressing the［Del］key． |
| :--- | :--- |
| $\mathbf{x}$ | Add new tappets with this tool．Click the insertion point in the path． |

Table 747：View＂Properties＂
The tappet is assigned to a result，if it is passed from the position of the master axis in the positive（increasing master values）or negative direction．

| X | Position of the tappet |
| :---: | :---: |
| Positive pass | Switch on／off attribute <br> －No action <br> －Switch to ON <br> －Switch to OFF <br> －Invert |
| Negative pass | Switch on／off attribute <br> －No action <br> －Switch to ON <br> －Switch to OFF <br> －Invert |

Table 748：Table of the possible combinations of tappet attributes

| Tappet symbol | Positive pass | Negative pass |
| :---: | :---: | :---: |
| Z | No action | No action |
| Z | Switch to ON | No action |
| $\triangle$ | Switch to OFF | No action |
| － | No action | Switch to ON |
| Z | No action | Switch to OFF |
| 区 | Switch to ON | Switch to OFF |
| Z | Switch to ON | Switch to OFF |
| $\triangle$ | Switch to OFF | Switch to ON |
| 又 | Switch to OFF | Switch to OFF |
| X | Invert | No action |
| 又 | No action | Invert |
| 区 | Switch to ON | Invert |
| 区 | Invert | Switch to ON |
| X | Invert | Switch to OFF |


| Tappet symbol | Positive pass | Negative pass |
| :--- | :--- | :--- |
| $Z$ | Switch to OFF | Invert |
| $区$ | Invert | Invert |

### 6.5.10.2.7 Tappet table tab

This tabular editor is an alternative to the graphical editor for configuring the tappet paths ("Tappets" tab). A tappet path defines one or more tappets depending on the master position. In the table, the lines with the definitions of the associated tappets follow below each line that defines a tappet path.

Switch between the graphical editor and the alternative tabular editor at any time ("Tappet table" tab).

| - | Inserts a new tappet. |
| :---: | :---: |
| - | Deletes the tappet. |
| Track ID | ID of the tappet path. <br> All tappets of a tappet path refer to the same tappet switch (a variable of type BOOL). |
| X | Position of the tappet. |
| Positive pass | Switch on/off attribute <br> - No action <br> - Switch to ON <br> - Switch to OFF <br> - Invert |
| Negative pass | Switch on/off attribute <br> - No action <br> - Switch to ON <br> - Switch to OFF <br> - Invert |

Table 749: View "ToolBox"

| $\star$ Select | Select the tappets by means of this tool. You can drag the selected <br> tappets to another position. |
| :--- | :--- | tappets to another position.

Table 750: View "Properties"
The tappet is assigned to a result, if it is passed from the position of the master axis in the positive (increasing master values) or negative direction.
X $\quad$ Position of the tappet

| Positive pass | Switch on/off attribute <br> - No action <br> - Switch to ON <br> - Switch to OFF <br> - Invert |
| :---: | :---: |
| Negative pass | Switch on/off attribute <br> - No action <br> - Switch to ON <br> - Switch to OFF <br> - Invert |

### 6.5.10.2.8 Cam additional settings

Once the user has the "Cam" view open (double-click on the "Cam" object in the device tree) the user will have access to several additional "Cam" settings and options:


## Cam ToolBox

This will allow the user to add and remove elements to the "Cam" and tappet editors. Typical options are:

Table 751: View "ToolBox"

| * Select | Select a line in the table by using this tool. <br> Selected points are deleted by pressing the [Del] key. |
| :--- | :--- |
| Add point | Add new points with this tool. Click the insertion point in the diagram. <br> Te graph is then adapted automatically so that its curve runs through <br> the new inserted point. |

This can be accessed by either "View $\rightarrow$ Properties" or by right clicking on the "Cam" object and selecting properties. It will allow the user to define the behavior and dimensions of the "Cam".


Here the user can change:

- Dimensions: The master and slave "Cam table" start and end positions.
- Slave period: Depending on these parameters, the slave start point is adjusted automatically when the end point is changed, as well as the other way around. This adjustment optimizes the period transition to be as smooth and jerk-free as possible.
- Continuity requirements: Activation of these options prompt a continuity check, which reports any violations to the message view (Cam). Clear these options, if the application requires it.
- Compile format: In the Automation Builder is only one option: "polynomial (XYVA)" - Polynomial description of the points, for master position, slave position, slave velocity, and slave acceleration.


### 6.5.10.2.9 Display generated Code

To view the code that is generated by the "Cam" object:

1. Close all Cam views.
2. Open the Cam which's code needs to be viewed.
3. Go to "Cam" tab from the main Automation Builder tab.
4. Select "the display generated code" option.


The code will be shown where user can find the Cam points that are created in an array. "Cam table" function block from motion library is already initialized with few inputs.

Below is an example of the code generated by the "Cam table $A B B$ " object (Automation Builder 2.6.1 and newer).


Here you can see more in depth information about the Cam is encoded in the structure.
Below is an example of the code generated by the "Cam table ABB basic" object (Automation Builder 2.6.0 and older).

```
S cenerded coce >
(fattribuce 'IInkalvays')
vaz olobal
Cam_21 גNNAY[0..3] or MC_PDrofile i= !
    master_position i= 0, interpolation_poiat i= 0, velocity_ratio i= 0, aoceleration_ratio i- 0)
    (master position i= 220, interpolation, poist i= 120, velootty_ratio i= 1, acceleration_ratio i= 0)
    (master position i= 360, interpolacion point i= 360, veloeve\ ratio i= 0, acceleration ratio i= 0) )
```



```
[0%_vas
```



Here you can see only basic Cam data is encoded in the structure.
Currently it can only show the code generated from one Cam which is open.

### 6.5.10.2.10 <br> Creating Cams

The steps to create a Cam can be explained using an example application. In this application, we can say that there is a rotary table with eight slots ( $360^{\circ} / 8=45^{\circ}$ division). Inside there is a component to be joined by ultrasonic welding. The welding tool is fed by a linear drive after the rotary table has rotated. After welding, the linear axis returns and the rotary table continues to rotate.

Work steps:

- Rotary table turns $45^{\circ}$ (duration: 400 ms ).
- The welding head is moved down by a vertical axis of 250 mm (duration: 200 ms ).
- Start welding (duration: 1200 ms ).
- The welding head is moved up by a vertical axis of 250 mm (duration: 200 ms ).

A cycle time of 2000 ms results from total times.
The application is implemented by means of a virtual master axis that runs continuously (modulo). The end value of the axis is projected according to the cycle time of 2000 ms . The rotary table is achieved as a Cam (modulo; end value: $45^{\circ}$ ). The vertical axis is also achieved as a Cam (restricted; end value: 300 mm ). The welding process is controlled by a tappet.

### 6.5.10.2.11 Setting up project

1. Select "Motion Solution" project.
2. Select the required AC500 controller.
3. Add two real drives and configure as below:
$\Rightarrow$ Name "RotaryAxis", Type: Modulo, units: Degrees, scale: Gearing 8/1, 1 Tool Rev = $360^{\circ}$, Modulo 0-45 ${ }^{\circ}$
$\Rightarrow$ Name "VerticalAxis" Type: Linear (Rotary Screw), units: mm, scale: 1 Motor Rev = 10 mm
4. Add a virtual axis (and don't rename it)
$\Rightarrow$ Name "MasterAxis" Type: Modulo, units: Pulse, Modulo range $=2000$


Adding a Cam to the device tree

1. Select the "Application" object in the device tree.
2. Click "Project $\rightarrow$ Add object $\rightarrow$ Cam table".
$\Rightarrow$ "Cam table $A B B$ " - This allows the user to get additional data via Cam structure and user need to use "MCA_CamStructureSelect" instead to "MC_CamTableSelect".
$\Rightarrow$ "Cam table ABB basic" - This allows the user to use the Cam object in the traditional ABB way together with "MC_CamTableSelect".
3. Specify the name "RotaryAxisCam" for the Cam and click [OK].
$\Rightarrow$ The object is inserted into the device tree.
4. Insert another Cam.
5. Rename it "VerticalAxisCam".

## Setting the properties of the Cam

1. Select the "Rotary table" Cam in the device tree.
2. Click "Properties" in the "View" menu or in the context menu.
3. Select the "Cam" tab.
4. Specify the following values:

- Master start position: 0
- Master end position: 2000
- Slave start position: 0
- Slave end position: 45
- Slave Period: 45 (NA)
- Smooth transition: (deactivated)

5. Click $[O K]$ to close the dialog. Confirm the dialog for changing the Cam object.
6. Change the values for the "Vertical axis" Cam in the same way:

- Master start position: 0
- Master end position: 2000
- Slave start position: 0
- Slave end position: 300
- Slave Period: 300
- Smooth transition: (activated)

7. Click $[O K]$ to close the dialog. Confirm the dialog for changing the ca, m object.

### 6.5.10.2.12 Changing the Cam path

These instructions use the example from the section \& Chapter 6.5.10.2.10 "Creating Cams" on page 3949 to demonstrate how to change a Cam.

## Changing the path with the graphical editor

1. Open the "Rotary table" Cam in the editor.
$\Rightarrow$ The "Cam" tab is visible.
2. Select the point at 120 and 240 and delete them by pressing the key [Del].
3. Select the "Add point" tool from the "ToolBox" view.
$\Rightarrow$ The mouse pointer turns into crosshairs when you move it into the editor.
4. Click near "Master position" 400 and "Slave position" 45 in the upper graphs (slave position).
$\Rightarrow$ The curve of the slave position is changed. The curves of velocity, acceleration, and jerk also change.
5. Select the new inserted point by clicking it.
6. Drag the point to another position.
$\Rightarrow$ The curve of the slave position is adjusted accordingly.
7. Select this point and open the properties view and change the " $X$ " value to 400 and the " $Y$ " value to 45 .
8. 9. Using the same method, change the end point to an " $X$ " value of 2000 and " $Y$ " value to 45.
1. Select the "Select" tool from the "ToolBox" view.
2. Select the second curve element (between 400 and 2000).
3. Change the "Segment type" property to "Line".
4. Check the curve in the graphical editor.


## Changing the Cam path with a "Cam table" editor

1. Open the "Vertical axis" Cam in the editor.
$\Rightarrow$ The "Cam" tab is visible.
2. Select the "Cam table" tab.
3. Delete the points at 120 and 240 by clicking the symbol.
4. Click the symbol.
$\Rightarrow$ A new point and a new segment are inserted at (1000/150).
5. Add two more points.
6. Change the values $X / Y$ of the following points:

- Point 1: $0 / 0$
- Point 2: 400 / 0
- Point 3: $600 / 250$
- Point 4: 1800 / 250
- Point 5: 2000 / 0
$\Rightarrow$ The curve of the slave position is changed. The curves of velocity, acceleration, and jerk also change.

7. In the "Cam table", change the "Segment type" of the first and third segments to "Line".
8. Check the curve in the graphical editor.


By clicking "Display generated Code," you can display the automatically created global variables.

## Application program using a generated Cam and Tappet table

Once the Cam and Tappet tables are created by the Cam editor, user can use the Cam function blocks to use the same in user application. These instructions use the same example as the previous example (from the section ${ }^{4} \Rightarrow$ Chapter 6.5.10.2.10 "Creating Cams" on page 3949) to demonstrate how to define the application program for Cam and tappet points.

Using Cam tables in application code

In the below example we have used an instance of "MC_CamStructureSelect" to connect to the Cam directly (no ADR is needed). Also this function has inputs to allow the user to directly define "Reverse" or "Periodic" settings so the use of "MCA_Cam_Extra" is no longer needed.


Note that in this instance the Cam; "Cam_1" is of the type "ABB Cam Table".

Using Tappet To use the Cam Tappets the user has two options: "MCA_CamDigitalSwitch" or tables in application code
"MCA_GetTappetValue".
"MCA_GetTappetValue" uses the output from the "MCA_CamStructureSelect" and will reflect the value based upon the "CAM SLAVE" position and "W$/ L L$ " take into account offsets or changes to scaling.

In the example, as the second Cam editor object name is "VerticalAxisCam". If the user has opted to use the new "ABB Cam Table" object and "MCA_CamStructureSelect" then they can use this function to set an output based on the Cam. An excerpt from the generated code can be seen below:

"MCA_CamDigitalSwitch" uses part of the "CAM STRUCTURE" and will reflect the value based upon the "AXIS" position and will NOT take into account offsets or changes to scaling. If using the this the "MCA_CAM" tappet array mut be accessed. This is generated automatically based on the configuration in tappet. The array name follows the format [Cam object name] $+[T]$.

In the example, as the second Cam editor object name is "VerticalAxisCam" then the "MCA_CAMTappet" array instance name is "VerticalAxisCam_T". An except from the generated code can be seen below:

VerticalAxisCam_T: ARRAY[0..1] OF MCA_CAMTappet := [
This can be directly passed to input pin name "Tappets" of function block
"MCA_DigitalCamSwitch".


More details on how to use the Cam curve can be found in Automation Builder online help and from the "AC500_MotionControl library" integrated documentation.

Importing a Cam from 3rd party CODESYS controller

## Exporting the Cam for the the 3rd party PLC

Using a Cam application that has been previously written using another 3rd party CODESYS based PLC it's possible to reuse these same data points in your new Automation Builder project to save the user time and effort. The steps to do this are below, in the example application we are using a inovance AM600 PLC program.

1. Open the 3rd party CODESYS environment and the project in question.
$\Rightarrow$ As when using ABB's solution, the Cam object must be selected from the project tree to allow the user to access the "Cam" item in the system menu at the top of the page which will be needed for some of the steps that follow.


The first thing you should consider is the master and slave settings such as start and end position, and slave period.
2. Check this, with a right click on the Cam object in the hardware tree and select $P$ "'roperties.
$\Rightarrow$ From here you will get a pop up where you can double check the settings that will define the Cam operation. Make a note of these.

3. Look at the "Cam table" just to get an idea about the general layout, here we can see this Cam has 6 points (beginning, end and 4 user selected items).

| Com | Cam table | Tappets Tappet table |  |  |  | Segm... |  |  |  | $\max (1)$ |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | X | Y | V | A | J |  | $\min \left(P^{-}\right.$ | $\max (\ldots$ | $\max (1)$ |  |
|  | 0 | 0 | 0 | 0 | 0 |  |  |  |  |  |
| $\bigcirc$ |  |  |  |  |  | Line | 0 | 0 | 0 | 0 |
| - | 10 | 0 | 0 | 0 | 0 |  |  |  |  |  |
| - |  |  |  |  |  | Poly5 | -4.69... | 54.76... | 2.168... | 0.056... |
| 曾 | 103.6... | 54.76... | 2.168... | 0 | 0 |  |  |  |  |  |
| $\bigcirc$ |  |  |  |  |  | Line | 54.76... | 227.9... | 2.168... | 0 |
| - | 183.5... | 227.9... | 2.168... | 0 | 0 |  |  |  |  |  |
| - |  |  |  |  |  | Poly5 | -6.25... | 273.8... | 3.587... | 0.098... |
| - | 350 | 0 | 0 | 0 |  |  |  |  |  |  |
| $\bigcirc$ |  |  |  |  |  | Line | 0 | 0 | 0 | 0 |
|  | 360 | 0 | 0 | 0 | 0 |  |  |  |  |  |

4. Consider the method for getting the Cam data exported from the application.
5. Select the "Cam" object to access the Cam menu.
6. Select the option for "Write Cam Data into ASCII Table".
7. Export or "Write" the data to an ASCII file.

8. Select somewhere to save it.
$\Rightarrow$ Then you will see a pop up which will ask you how many Cam data points you want to export into the file, we can treat this like resolution. In the example below we select 128 data points.


If we select many of points when the Cam is exported, we can expect the larger number of data points will give the same profile even using the 3rd party Cam builder's interpolation types to ensure the Cam profile is immediately matched perfectly to the previous application.
If we select a lower number of points, one that matches the interpolation points (the smallest number possible) for example then you are relying on Automation builder to plot the interceding Cam points so the interpolation types selected within Automation Builder are more important and must be checked.
Now we are finished with the 3rd party environment.

Importing the
Cam data into
Automation
Builder

1. Switch to Automation Builder in which we need to have a project which has a Cam in it ready for us to write the data into.
2. Check the properties of the Cam to check that it has the same maximum and minimum values and other characteristics.
3. Select the Cam object to access the menu in the "Tool bar" from which we can select "Read Cam Data from ASCII table".

4. Select the ASCII file we created before.

You will then be asked how many data points you want to import into the project. Typically, this should always match the value that was exported.


In the figure below you can see the result with all 128 data points imported.


Edit the segments (interpolation types) used in each segment in this newly imported Cam via the "Cam table" view.

| , Cam | [1] Diag_10_Bus |  | $\Theta^{9}$ Trace | $\sqrt{7}$ | Microflex_e ${ }^{\text {1 }}$ | 90_axis | Microflex_e190_1_axis |  |  | © | Motion ! |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| cam | cam table |  |  |  |  |  |  |  |  |  |  |
|  | X | Y | V | A | J | Segme... | $\min \left(\mathrm{P}_{0} \ldots\right.$ | $\max (\mathrm{P} .$. | $\max \left(\mathrm{l} \mathrm{V}_{\text {... }}\right.$ | $\max (1)$ |  |
|  | 0 | 0 | 0 | 0 | 0 |  |  |  |  |  |  |
| © |  |  |  |  |  | Polys ${ }^{5}$ | 0 | 0 | 0 | 0 | 0 |
| \% | 2.8346... | 0 | 0 | 0 | 0 |  |  |  |  |  |  |
| ¢ |  |  |  |  |  | Poly5 | -3.2302... | 0 | 4.4703... | 2.3264... |  |
| \% | 5.6692... | 0 | 0 | -2.3264... | 0 |  |  |  |  |  |  |
| © |  |  |  |  |  | Poly5 | -2.6924... | 1.6136... | 0.0001... | 0.0006... |  |
| \% | 8.5039... | 0 | -0.0001... | -0.0006... | 0 |  |  |  |  |  |  |

Once all necessary adjustments have been made, now the Cam is ready to be used in the program.

### 6.5.10.3 Motion control library

```
\square
Check the latest version of "Motion Library System Technology" in chapter 10.1 of the "AC500 V3 Motion Control Guide 3ADR 3ADR011239.pdf.
```


## ABB Library

```
Or via pdf document at: "C:|Users \(\mid\) Public \(\mid\) Documents 1 Automation Builder 2.x ExamplesIPS5611-Motion\Documentation
If the default path is inaccessible, click "Help \(\rightarrow\) Project examples" in the Automation Builder menu.
```


### 6.5.10.3.1 Safety instructions

- All pertinent state, regional, and local safety regulations must be observed when installing and using this product. When functions or devices are used for applications with technical safety requirements, the relevant instructions must be followed.
- Read the complete safety instructions of the user's manuals for the drives you are using, before installation and commissioning.
- Read all safety instructions and important user information of the AC500 PLC.

²) Chapter 1.4 "Regulations" on page 9

### 6.5.10.3.2 Preconditions for the use of the libraries

The library package has been released for the software and firmware versions listed in the readme file of Automation Builder only ("Help $\rightarrow$ Automation Builder Release Notes") . In no event will $A B B$ or its representatives be liable for loss of data, profits, revenue or consequential, incidental or other damage that may result from the use of other versions of product, software or firmware versions. The error-free operation of the HA library with other devices, software or firmware versions should be possible but cannot be guaranteed and may need adaptations e.g. of example programs.

The first version of Motion Control Library Package PS5611-Motion has been released with Automation Builder 2.4.0. There after the package is updated with several changes. For details on all changes please refer PS5611-Motion release note area from Automation Builder release notes.

The motion control package contains follows libraries:

| Library | Automation Builder | PLC firmware |
| :---: | :---: | :---: |
| ABB_MotionControl_AC500 | AB 2.4.0 or higher | AC500 V3 firmware version 3.3.1 or higher <br> AC500-eCo V3 firmware version 3.4.0 or higher |
| ABB_Ecat_CiA402_AC500 |  |  |
| ABB_MathFunctions_AC500 |  |  |
| ABB_MotionControlEco_AC500 (kernel blocks for eCo V3 PLCs) |  |  |
| ABB_MotionControlLoad_AC500 | AB 2.5.0 or higher | AC500 V3 firmware version 3.5.0 or higher |

The PS5611-Motion libraries have been tested with the following product/firmware/software versions:

- AC500 V3 PLC firmware 3.3.1 or higher
- AC500-eCo V3 (PTO \& PWM) firmware 3.4.0 or higher
- CM579-ETHCAT EtherCAT communication module firmware 4.4.3.21 or higher
- ABB e190 Drive
- CD522 module

In no event will $A B B$ or its representatives be liable for loss of data, profits, revenue or consequential, incidental or other damage that may result from the use of other versions of product/software/firmware versions. The error-free operation of the PS5611-Motion with other devices/software/firmware versions should be possible but can not be guaranteed and may need adaptations e. g. of example programs.

## CAUTION!

Generally, the user in all applications is fully and alone responsible for checking all functions carefully, especially for safe and reliable operation.

The function blocks contained in the library can only be executed in RUN mode of the PLC, but not in simulation mode.

Limits on number of synchronized axis

There are limits on the minimum EtherCAT cycle time, user can configure in each PLC type.

Table 752: Details on the limits on the minimum EtherCAT cycle time

| PLC type | PM5630 | PM5650 | PM5670 |
| :--- | :--- | :--- | :--- |
| Min. EtherCAT master cycle <br> time | 2 ms | 1 ms | $0,5 \mathrm{~ms}$ |

Other than the above limits, there is also limits on configuring the number of synchronized axis in each PLC type. This limits is based on the EtherCAT master cycle time configured under EtherCAT master.

Table 753: Details on the limits for each PLC type

| PLC type | PM5630 | PM5650 | PM5670 |
| :--- | :--- | :--- | :--- |
| Number of synchronized <br> axis in 1 ms | - | 8 | 16 |
| Number of synchronized <br> axis in 2 ms | 4 | 16 | 32 |
| Number of synchronized <br> axis in 4 ms | 8 | 32 | 64 |

"Number of axis" is counted in Automation Builder is based on the number of Kernel function block instance declared in the IEC application. In this way, it is made sure all real and virtual axis are counted.

User can increase the EtherCAT cycle time to accommodate more "Number of axis" in the same PLC type.

User can use the [Statistics] tab from Automation Builder to see how many axis are supported for the particular PLC type and for the EtherCAT master cycle time configured. Once the axis is configured user need to update the [Statistics] tab by "Generate Code" to get the updated information.

Automation Builder allows an additional axis than what is mentioned in the above table to support one virtual axis additionally.

Please remove any Kernel function block instance which is declared but not used in the application to get the correct number of axis calculated by Automation Builder under the [Statistics] tab.

### 6.5.10.3.3 Overview

## Introduction

The PS5611-Motion is a motion control library for AC500 V3 CPUs, to create motion control applications based on function blocks according to the standard of PLCopen motion control. These function blocks can be used for PLC-based motion control and cover a wide range of possible motion control functionalities. Starting from single axis movements to master-follower axes to perform electronic gearing and CAM functions.

The detailed functionality of each function block is defined in the integrated documentation of the library.

## PLC-based motion control

With PS5611-Motion the application program and the profile generator are realized in the PLC. The implementation of the profile generator is based on a set of function blocks which are named central motion control (CMC).
The profile generator of many possible axes is centrally placed inside the AC500 PLC. Therefore multiaxis motion functionalities become easily available and can be accessed by PLCopen function blocks. As a result, motion control functionalities are almost drive independent.

The detailed functionality of each function block is defined in the integrated documentation of the library.

Available motion control functionalities:

- Simple axis Movements
- Electronic Gearin
- Electronic CAMs
- Position Profiles
- Velocity Profiles
- Acceleration Profiles
- Load control (Torque profilling)

Then the output is a position reference signal which the drive will follow. A new position reference value will be calculated with every cycle of the PLC and has to be transferred to the drive, which demands real time capabilities to the PLC and to the communication channel. A real time fieldbus like EtherCAT is needed. The feedback of the actual position can be used for supervision purposes during operation and is needed to adjust the value of the position reference before the drive will be enabled.

## AC500 as Motion Controller (Central Motion Control)



Servo Drive (Non Intolligent)


Fig. 311: System structure of PLC-based motion control with AC500 PLC and PS5611-Motion
With PLC-based motion control it is also possible to include the position control loop to the AC500 PLC. In this case a speed reference signal will be transferred to the drive, which makes it possible to perform the full range of motion functionalities with standard drives. To close the position control loop, feedback of the actual position is mandatory.


Fig. 312: PLC-based motion control with AC500 PLC and PS5611-Motion, closed position control loop

With PLC-based motion control it is also possible to include the load control loop to the AC500 PLC. In this case a speed reference signal will be transferred to the drive, which makes it possible to perform the full range of motion functionalities with standard drives. To close the position control loop, feedback of the actual position is mandatory and to close the load control loop, feedback of the actual load / torque is mandatory.

AC500 as Motion Controller (Central Motion Control)


Fig. 313: PLC-based motion control with AC500 PLC and PS5611-Motion, closed load control loop


Fig. 314: Central motion control with AC500 PLC and PS5611-Motion, different axis implementations at the same time

## Overview of PLCopen function blocks

The following tables give an overview of the defined function blocks, divided into administrative (not driving motion) and motion related sets. They give an overview which function block could be used for the different possible configurations.

These function blocks are part of the ABB_MotionControl_AC500 and ABB_MotionControlLoad_AC500 library.

If there are restrictions concerning a certain drive ("XXX") which lead to a different or limited behavior compared to the standard the respective chapter is supplemented with an additional paragraph "Notes for XXX".

The "KERNEL" function blocks are available in different variants.
The "CMC_Basic_Kernel" and "CMC_Load_Motion_Kernel" function block is designed to be used in standard $\bar{V} 3$ PLCs, and can either work with drives connected to a fieldbus or IOs.
The "OBIO_PTOMotionKernel" or "OBIO_PWMMotionKernel" function blocks (part of AC500_MotionControlEco) are solely to be used in AC500-eCo V3 CPUs and to make use of the integrated stepper-IO along with PLCopen function blocks. It connects automatically to the internal IOs.

For details of the limitations of PTO and PWM outputs in eCo V3 PLCs, refer to Automation Builder help file.

Table 754: Motion control administrative function blocks

| Function block | PLC-based motion control |  |  |
| :--- | :--- | :--- | :--- |
|  | CMC_Basic_ <br> Kernel | CMC_Load_Motion <br> _Kernel | OBIO_PTOMotion- <br> Kernel/ <br> OBIO_PWMMotion- <br> Kernel |
| MC_Power | X | X | X |
| MC_ReadStatus | X | X | X |
| MC_ReadAxisError | X | X | X |
| MC_ReadParameter | X | X | X |
| MC_ReadBoolParameter | X | X | X |
| MC_WriteParameter | X | X | X |
| MC_WriteBoolParameter | X | X | X |
| MC_Reset | X | X | X |
| MC_ReadActualPosition | X | X | X |
| MC_ReadActualVelocity | X | X | X |
| MC_SetOverride | X | X | X |
| MC_SetPosition | X | X | X |
| MC_CamTableSelect | X | X | X |

Table 755: Motion control single-axis function blocks

| Function block | PLC-based motion control |  |  |
| :--- | :--- | :--- | :--- |
|  | CMC_Basic_K <br> ernel | CMC_Load_Motion <br> _Kernel | OBIO_PTOMotion- <br> Kernel/ <br> OBIO_PWMMo- <br> tionKernel |
|  | X | X | X |
| MC_MoveRelative | X | X | X |
| MC_MoveAdditive | X | X | X |
| MC_MoveSuperimposed | X | X | X |
| MC_HaltSuperimposed | X | X | X |
| MC_MoveVelocity | X | X | X |
| MC_MoveContinuousAbsolute | X | X | X |
| MC_MoveContinuousRelative | X | X | X |
| MC_Stop | X | X | X |
| MC_PositionProfile | X | X |  |
| MC_VelocityProfile | X | X | X |
| MC_AccelerationProfile | X | X | X |
| MC_Halt |  |  |  |

Table 756: Motion control multi-axis function blocks

| Function block | PLC-based motion control |  |  |
| :---: | :---: | :---: | :---: |
|  | CMC_Basic_K ernel | CMC_Load_Motion _Kernel | OBIO_PTOMotionKernel/ OBIO_PWMMotionKernel |
| MC_Camln | X | X | X |
| MC_CamOut | X | X | X |
| MC_Gearln | X | X | X |
| MC_GearlnPos | X | X | X |
| MC_GearOut | X | X | X |
| MC_PhasingAbsolute | X | X | X |
| MC_PhasingRelative | X | X | X |
| MC_CombineAxes | X | X | X |
| MC_HaltPhasing | X | X | X |

Table 757: Motion control homing function blocks

| Function block | PLC-based motion control |  |  |
| :--- | :--- | :--- | :--- |
|  | CMC_Basic__ <br> Kernel | CMC_Load_Motion__ <br> Kernel | OBIO_PTOMotion- <br> Kernel/ <br> OBIO_PWMMo- <br> tionKernel |
| MC_StepAbsSwitch | X | X | X |
| MC_StepLimitSwitch | X | X | X |


| Function block | PLC-based motion control |  |  |
| :--- | :--- | :--- | :--- |
|  | CMC_Basic_ <br> Kernel | CMC_Load_Motion__ <br> Kernel | OBIO_PTOMotion- <br> Kernel/ <br> OBIO_PWMMo- <br> tionKernel |
|  | X | X | X |
| MC_StepDirect | X | X | X |

Table 758: Motion control ABB specific function blocks

| Function block | PLC-based motion control |  |  |
| :---: | :---: | :---: | :---: |
|  | CMC_Basic_K ernel | CMC_Load_Motion _Kernel | OBIO_PTOMotionKernel/ OBIO_PWMMotionKernel |
| MCA_Cam_Extra | X | X | X |
| MCA_Indexing | X | X | X |
| MCA_JogAxis | X | X | X |
| MCA_MoveByExternalReference | X | X | X |
| MCA_MoveVelocityContinuous | X | X | X |
| MCA_Parameter | X | X | X |
| MCA_ReadParameterList | X | X | X |
| MCA_WriteParameterList | X | X | X |
| MCA_SetPositionContinuous | X | X | X |
| MCA_GearInDirect | X | X | X |
| MCA_CamInDirect | X | X | X |
| MCA_SetOperatingMode | X | X | X |
| MCA_CamInfo | X | X | X |
| MCA_DriveBasedHome | X | X | X |
| MCA_MoveRelativeOpto | X | X | X |
| MCA_PhasingByMaster | X | X | X |

Table 759: Motion control fluid power function blocks

| Function block | PLC-based motion control |  |  |
| :--- | :--- | :--- | :--- |
|  | CMC_Basic__ <br> Kernel | CMC_Load_Motion_ <br> Kernel | OBIO_PTOMotion- <br> Kernel// <br> OBIO_PWMMo- <br> tionKernel |
|  | - | X | - |
| MC_LimitMotion | - | X | - |
| MC_LoadControl | - | X | - |
| MC_LoadProfile | - | X | - |
| MC_LoadSuperimposed | - | X | - |
| MC_TorqueControl | - | X | - |

## Overview of libraries

$\triangleright \quad$ Add the following libraries for the listed applications.
$\Rightarrow$ In some cases by adding a library, there will be other libraries added automatically.

| Application | Library to be added manually |
| :--- | :--- |
| PLC-based motion control | ABB_MotionControl_AC500.-compiled-library |
|  | ABB_MathFunctions_AC500.-compiled-library |
| PLC-based motion control, optional for <br> EtherCAT | ABB_Ecat_CiA402_AC500.-library |
| Motion control with eCo V3 (OBIO_PTOMo- <br> tionKernel \& OBIO_PWMMotionKernel) | ABB_MotionControlEco_AC500.compiled <br> ibrary |
| PLC-based motion control - Fluid Power <br> Extensions | ABB_MotionControlLoad_AC500.compiled <br> ibrary |

The features of the function blocks provided with PS5611-Motion can be used from the PLC program according to PLCopen standard. Different drives and different motion control realizations could be used and can be combined with each other as well as different fieldbuses. ABB_Ecat_CiA402_AC500.library is editable and can be adapted based on the drive configuration and drive type.

## Overview of data types

The following data types are used for the motion control library. The data types are defined in the library file ABB_MotionControl_AC500 compiled-library. The corresponding elements can be used for the function blocks inputs.

Table 760: Structures

| Data type | Elements | Element data type |
| :--- | :--- | :--- |
| CMC_AXIS_IO | limitSwitchPos | BOOL |
|  | limitSwitchNeg | BOOL |
|  | absRefSwitch | BOOL |
|  | master_position | LREAL |
|  | interpolation_point | LREAL |
| MC_TPROFILE | velocity_ratio | LREAL |
| M Chapter 6.5.10.3.5.7.3 <br> "PositionTimeProfile" <br> on page 4036 | interpolation_point | first_derivative |
|  | second_derivative | LREAL |
|  | delta_time | LREAL |

Table 761: Enum

| Data type | Possible values |
| :--- | :--- |
| MC_ABB_iTYPES_ENUM | MCA_SPLINE_COMPLETE |
| « <br> "Interpolation types for pro- <br> files" on page 4036 | MCA_SPLINE_NATURAL |
|  |  |


| Data type | Possible values |
| :---: | :---: |
|  | MCA_POLY3 |
|  | MCA_LINEAR |
| MC_BUFFERMODE | mcABORTING |
|  | mcBuFFERED |
|  | mcBLENDINGIow |
|  | mcBLENDINGprevious |
|  | mcBLENDINGnext |
|  | mcBLENDINGhigh |
| MC_DIRECTION | DEFAULT |
|  | POSITIVE |
|  | SHORTEST |
|  | NEGATIVE |
|  | CURRENT |
|  | POSITIVE_STOP |
|  | NEGATIVE_STOP |
|  | CURRENT_STOP |
| MC_HOMING_DIRECTION | MC_SwitchNegative |
|  | MC_SwitchPositive |
|  | MC_Positive |
|  | MC_Negative |
| MC_HOMING_EDGE | MC_EdgeOn |
|  | MC_EdgeOff |
|  | MC_On |
|  | MC_Off |
| MC_HOMING_MODE | MC_REFPULSE |
|  | MC_DIRECT |
| MC_SOURCE | mcActualValue |
|  | mcSetValue |
| ERROR_ID | MC_Ok |
|  | Wrong_State |
|  | Drive_Problem |
|  | Parameter_Exceeds_Limit |
|  | No_Field_Access |
|  | Bus_Problem |
|  | Abs_Switch_Error |
|  | Timeout |
|  | NAK |
|  | MC_TimeLimitExceeded |
|  | MC_DistanceLimitExceeded |
|  | MC_TorqueLimitExceeded |


| Data type | Possible values |
| :---: | :---: |
|  | Not_Implemented |
|  | ErrorID_POSITION_FOLLOW |
|  | ErrorID_POSSW |
|  | ErrorID_NEGSW |
|  | ErrorID_VELOCITY_FAULT |
|  | ErrorID_INTERPOLATION_FAULT |
|  | ErrorID_WARNING_VELOCITYLIMIT |
|  | ErrorID_WARNING_POSITIONLIMITPOS |
|  | ErrorID_WARNING_POSITIONLIMITNEG |
|  | ErrorID_WARNING_POSITIONOVERRUN |
|  | ErrorID_WARNING_ABORT |
|  | ErrorID_WARNING_MOVEMENT_DIRECTION |

## Naming of function blocks and data structures

PLCopen All function blocks and data types named MC_xxx are implemented according to PLCopen definition and follow the PLCopen documentation. They may have additional inputs but according to PLCopen rules.
All function blocks and data types named MCA_xxx are implemented corresponding to PLCopen rules with adaptations specific to AC500. They are AC500 specific extensions to the PLCopen library.

## PLC-based motion control

All function blocks named CMC_xxx belong to the implementation of PLC-based motion control. All data types named CMC_xxx belong to the implementation of PLC-based motion control.
All data types named AXIS_xxx exist according to PLCopen definition. The content is ABB specific and not documented.
All function blocks named zCMC_xxx belong to the implementation of PLC-based motion control. These are not documented and not intended for customer use.
All function blocks and data types named MC_xxx are implemented according PLCopen definition and follow the PLCopen documentation.
All function blocks and data types named OBIO_xxx in the ABB_MotionControlEco_AC500 library are intended for use with AC500-eCo V3-PLCs only.
All function blocks named xxx_APP are not write protected and may be modified for adaptations. Editable library is available in the example folder.

Editable library is available in the example folder.

### 6.5.10.3.4 PLCopen

Introduction
Based on application requirements and project specifications engineers are required to use or select a wide range of motion control hardware. In the past this required unique software to be created for each application even though the functions are the same. PLCopen motion standard provide a way to have standard application libraries that are reusable for multiple hardware platforms. This lowers development, maintenance and support costs while eliminating confusion. In addition, engineering becomes easier, training costs decrease, and the software
is reusable across platforms. Effectively, this standardization is done by defining libraries of reusable components. In this way the programming is less hardware dependent, the reusability of the application software increased, the cost involved in training and support reduced, and the application becomes scalable across different control solutions. Due to the data hiding and encapsulation, it is usable on different architectures, for instance ranging from centralized to distributed or integrated to networked control. It is not specifically designed for one application, but will serve as a basic layer for ongoing definitions in different areas. As such it is open to existing and future technologies.

ABB is a member of the PLCopen organization. More Information about PLCopen can be read on the PLCopen website.


Fig. 315: PLCopen motion control logo

Function blocks according to PLCopen are designed for controlling axes via the language elements consistent with those defined in the IEC 61131-3 standard. It was decided by the task force that it would not be practical to encapsulate all the aspects of one axis into only one function block. The retained solution is to provide a set of command-oriented function blocks that have a reference to the axis, e.g. the abstract data type Axis, which offers flexibility, ease of use and reusability.

Implementations based on IEC 61131-3 (for instance via function blocks and SFC) will be focused towards the interface (look-and-feel/proxy) of the function blocks. This specification does not define the internal operation of the function blocks.

PLCopen motion control function blocks can be used in any IEC 61131-3 programming language. The following figure shows an example of a function block used in Function Block Diagram (FBD) language.


Fig. 316: Command for absolute positioning according to PLCopen standard
Application programs which use the manufacturer independent function blocks according to PLCopen will lead to the following advantages:

- Reusable software structure for different platforms.
- Programming based on function blocks.
- Function blocks can be used in any IEC 61131-3 language.

All function blocks which are defined by PLCopen will have the following qualities independently to the manufacturer of the motion control system:

- Same inputs/outputs
- Same functional behavior
- Same name

The following parts of the PLCopen motion control definition are completely or partly included in this product:

- Part 1: Function blocks for motion control
- Part 2: Extensions
- Part 3: User Guidelines
- Part 4: Homing Procedures
- Part 6: Function blocks for motion control - Fluid Power Extensions


## Programming guidelines

This chapter explains some rules on the usage of the libraries and the structure Axis_Ref.

- In general, the kernel function block and the transfer of axis IO data should be processed in a cyclic task. This task should be as short and real-time as possible to achieve the best motion control performance. Always make sure Kernel function block is called at the highest priority task and other applications must be at a lower priority task.
- If Axis_Ref is used as input on a user defined function block or program or function, always use it as VAR_IN_OUT and never use it as VAR_INPUT or VAR_OUTPUT. The reason is that this would
- Break the consistency and destroy data.
- Consume a lot of computing power by copying data.
- Any instance of a function block should be called only once per cycle and in only one specific task.
If the instance is used in several tasks, it has to be checked that is not called several times.
Because this could corrupt the handshake from function block to Axis_Ref to
CMC_Basic_Kernel and vice versa.
- Some PLCopen function blocks are only allowed to be called within the same task as the CMC_Basic_Kernel function block. This is mentioned in the function block descriptions.
- If PLCopen function blocks are called from a different task, the cycle time should be at least 2 times the cycle time for CMC_Basic_Kernel function block.


## Axis data type Axis_Ref

The Axis_Ref is a structure that contains information on the corresponding axis. It is used as a VAR_IN_OUT in all motion control function blocks defined in this document. The content of this structure is implementation dependent and can ultimately be empty. If there are elements in this structure, the supplier shall support the access to them, but this is outside of the scope of this document. The refresh rate of this structure is also implementation dependent. According to IEC 61131-3 it is allowed to switch the Axis_Ref for an active function block, for instance from Axis1 to Axis2. However, the behavior of this can vary across different platforms, and is not encouraged to do.

Axis_Ref data type declaration:
TYPE Axis_Ref : STRUCT
(Content is implementation dependent)
END_STRUCT

```
Example: TYPE Axis_Ref : STRUCT
    AxisNo: UINT; AxisName: STRING (255);
    END_STRUCT
```


## The single axis state diagram

The following diagram normatively defines the behavior of the axis at a high level when multiple motion control function blocks are simultaneously activated. This combination of motion profiles is useful in building a more complicated profile or to handle exceptions within a program. (In real implementations there may be additional states at a lower level defined). The basic rule is that motion commands are always taken sequentially, even if the PLC had the capability of real parallel processing. These commands act on the axis' state diagram.
The axis is always in one of the defined states (see diagram below). Any motion command that causes a transition changes the state of the axis and, as a consequence, modifies the way the current motion is computed. The single axis state diagram is an abstraction layer of what the real state of the axis is, comparable to the image of the I/O points within a cyclic (PLC) program. A change of state is reflected immediately when issuing the corresponding motion command.


The diagram is focused on a single axis. The multiple axis function blocks, MC_Camln, MC_Gearln and MC_Phasing, can be looked at, from a single axis state diagram point of view, as multiple single-axes all in specific states. For instance, the CAM-master can be in the state Continuous Motion. The corresponding slave is in the state Synchronized Motion. Connecting a slave axis to a master axis has no influence on the master axis.
The state Disabled describes the initial state of the axis. In this state the movement of the axis is not influenced by the function blocks. The axis feedback is operational. If the MC_Power function block is called with Enable=TRUE while being in state Disabled, this either leads to Standstill if there is no error inside the axis, or to ErrorStop if an error exists.

Calling MC_Power with Enable=FALSE in any state, the axis goes to the state Disabled, either directly or via any other state. If a motion generating function block controls an axis, while the MC_Power function block with Enable=FALSE is called, the motion generating function block is aborted (CommandAborted).
The intention of the state ErrorStop is that the axis goes to a stop, if possible. There are no further inputs from function blocks accepted until a reset has been done from the ErrorStop state.
The transition Error refers to errors from the axis and axis control, and not from the function block instances. These axis errors may also be reflected in the output of the function blocks instances errors.
Issuing MC_Home in any other state than StandStill will go to ErrorStop, even if MC_Home is issued from the state Homing itself.
Function blocks which are not listed in the single axis state diagram do not affect the state of the axis, meaning that whenever they are called the state does not change. They are:

MC_ReadStatus; MC_ReadAxisError; MC_ReadParameter; MC_ReadBoolParameter; MC_WriteParameter; MC_WriteBoolParameter; MC_ReadActualPosition and MC_CamTableSelect.

Calling the function block MC_Stop in state StandStill changes the state to Stopping and back to Standstill when Execute = FALSE. The state Stopping is kept as long as the input Execute is TRUE. The output Done is set when the stop ramp is finished.


Fig. 317: Function block state behavior

$$
\begin{aligned}
& \text { 1. In this state ErrorStop or Stopping, all function blocks can be called, although } \\
& \text { they will not be executed, except MC_Reset and Error - they will generate the } \\
& \text { transition to StandStill or ErrorStop respectively. }
\end{aligned}
$$

2. Power.Enable=TRUE and there is an error in the Axis.
3. Power.Enable=TRUE and there is no error in the Axis.
4. MC_Stop.Done AND NOT MC_Stop.Execute.

## Visualizations

For usage with the PLCopen Library, a set of visualization objects is defined. These visualizations use the placeholder concept, which means that they could be used in an actual visualization several times and be instantiated by replacing the "placeholder" with an effective data-structure.

Two types of visualizations exist:

- As placeholder, an instance of Axis_Ref should be used. These are named:

MC_VISU_Axis_name. Here the name could be state machine or its actual.

- As placeholder, an instance of the respective PLCopen function block should be used. These visualizations are named MC_VISU_FB_name where "name" could be MoveAbsolute or MoveVelocity, so the complete element is named MC_VISU_FB_MoveAbsolute or MC_VISU_FB_MoveVelocity.
The background colour and the colour for the title of each element could be changed. The colours are defined in some global predefined variables in MC_VISU_COLOR_INFORMATION. By changing these values, different colours will be used.


## MC_VISU_COLOR_INFORMATION (GVL)

InOut:

| Name | Type | Initial | Comment |
| :--- | :--- | :--- | :--- |
| MC_VISU_BACKGROUND_COLOR | DWORD | $16 \# F F 888888$ | 16\#FFRRGGBB (* Color combination for the <br> Grey color $\left.{ }^{*}\right)$ |
| MC_VISU_TITLE_COLOR | DWORD | $16 \#$ FFFFFF00 | $16 \# F F R R G G B B ~(* ~ C o l o r ~ c o m b i n a t i o n ~ f o r ~$ <br> yellow*) |

Below, some existing visualizations are shown.

MC_VISU_Axis_ This shows the state machine of the axis according to PLCopen definition. The active state is StateMachine shown green except the ErrorStop which is shown red. Usually, it starts with Disabled. When no remote connection to the drive is available, it will switch to ErrorStop immediately.

The placeholder of this visualization has to be connected to an instance of the data type Axis_Ref.


MC_VISU_Axis_ This object shows some actual values.
The Placeholder of this visualization has to be connected to an instance of the data type Axis_Ref.

| Axis actual values |  |
| :--- | :--- |
| AxisRefX |  |
| Position | -6021340.889 |
| Velocity | 0.00 |
| Axis error | 0 |
| Drive error | 0 |

MC_VISU_Axis_ This object shows the error information connected to the PLCopen function blocks. This is NOT FB_error a drive error. If no error occurs in the execution of a function block, just the name is shown. If an error occurred, it shows the name of the function block as well as the error number and a short description. In the example below, the MC_Power function block recognized that no fieldbus connection to the drive was available.

The Placeholder of this visualization has to be connected to an instance of the data type Axis_Ref.


## Error codes

Besides the diagnosis information of the drive which is described in the respective drive documentation, there are a number of error codes directly related to the function blocks. These error codes are displayed at the output "ErrorID" of the function block.

| Error Code | Mnemonic | Explanation |
| :--- | :--- | :--- |
| 0 | MC_Ok | No Error |
| 1 | WRONG_STATE | A function block was activated not according <br> to the state machine, e.g. tried to start a <br> movement while in state Disabled. |
| 2 | DRIVE_PROBLEM | The drive indicates an error, e.g. tripped. |
| 3 | PARAM- <br> ETER_EXCEEDS_LIMIT | A parameter at the function block is outside <br> the possible range. This does not refer to the <br> parameter range which is allowed for the drive <br> but just to the 32-Bit Integer which is used for <br> internal calculation. |
| 4 | NO_FIELD_ACCESS | No fieldbus connection to the drive. |
| 5 | BUS_PROBLEM | Not used |
| 6 | ABS_SWITCH_ERROR | During Homing, (when done by function <br> blocks) limit switch not according to moving <br> direction e.g. the positive switch occurred <br> when moving in negative direction. |
| 7 | TIMEOUT | Timeout in block execution. |


| Error Code | Mnemonic | Explanation |
| :---: | :---: | :---: |
| 8 | NAK | Parameter access not applicable |
| 9 | MC_TimeLimitExceeded | Used by function blocks with TimeLimit. |
| 10 | MC_DistanceLimitExceeded | Used by function blocks with DistanceLimit. |
| 11 | MC_TorqueLimitExceeded | Used by function blocks with TorqueLimit. |
| 12 | NOT_IMPLEMENTED | Functionality not implemented for certain axis type. |
| 101 | ErrorID POSITION_FOLLOW | Following error, caused by > position error => ERRORSTOP. (parameter POS_LAG_PERCENTAGE) |
| 102 | ErrorID_POSSW | Positive software limit switch => ERRORSTOP. The actual position did exceed the positive Software limit switch position. This supervision has to be activated with MC_WriteParameter. |
| 103 | ErrorID_NEGSW | Negative software limit switch => ERRORSTOP. The actual position did exceed the negative Software limit switch position. This supervision has to be activated with MC_WriteParameter. |
| 104 | ErrorID_VELOCITY_FAUL T | The measured velocity and commanded velocity are > 50\% (related to maximum velocity) apart, for a certain time =>ERRORSTOP (parameter V_CHECKTIME) |
| 105 | ErrorID_INTERPOLATION_FAULT | following error, caused by interpolation problem =>ERRORSTOP. Position following error occurred, but reason most likely a interpo- lation problem, not drive problem (e.g. CAM Table, position step). |
| 110 | ErrorID_WARNING_VELO CITYLIMIT | Velocity or acceleration/deceleration are in limitation, set by parameter EnableLimitVelocity, MaxVelocityAppl, MaxDecelerationAppl |
| 111 | ErrorID_WARNING_POSITIONLIMITPOS | Position is in limitation towards position limit (SWLimit2DecPos), axis decelerates near positive software limit switch |
| 112 | ErrorID_WARNING_POSITIONLIMITNEG | Position is in limitation towards position limit (SWLimit2DecNeg)., axis decelerates near negative software limit switch |
| 113 | ErrorID_WARNING_POSITIONOVERRUN | A linear axis created a 32bit position overrun ( $>2147483647$ u=>inc) =>configure modulo |
| 114 | ```ErrorID_WARNING_ABOR T``` | Axis aborted due to too large position gap due to velocity limitation |
| 115 | ErrorID_WARNING_MOVE MENT_DIRECTION | Either positive or negative direction blocked by MC_Power |

## Error handling

All access to the drive/motion control is via function blocks. Internally these function blocks provide basic error checking on the input data. Exactly, how this is done is implementation dependent. For instance, if MaxVelocity is set to 6000, and the Velocity input to a function block is set to 10,000, a basic error report is generated. In the case where an intelligent drive is coupled via a network to the system, the MaxVelocity parameter is probably stored on the drive. The function block must take care of the errors generated by the drive internally. With another implementation, the MaxVelocity value could be stored locally. In this case the function block will generate the error locally.
Both centralized and decentralized error handling methods are possible when using the motion control function blocks.
Centralized error handling is used to simplify programming of the function block. Error reaction is the same independent of the instance in which the error has occurred.


Fig. 318: Function blocks with centralized error handling

Decentralized error handling gives the possibility of different reactions depending on the function block in which an error occurred.


Fig. 319: function blocks with decentralized error handling

## PLCopen parameter

Additional parameters are available by ReadParameter and WriteParameter function blocks.


Following function blocks can be used for the read and write operation. Functionality of these blocks and its variables are explained in the integrated documentation

- MC_ReadParameter
- MC_WriteParameter
- MC_ReadBoolParameter
- MC_WriteBoolParameter

| Parameter numbe r (PN) | $\begin{aligned} & \text { Nam } \\ & \text { e } \end{aligned}$ | Datatype | Min. | Max. | Default | R/W | Comments |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 1 | Com-man-dedPositi on | DINT |  |  |  | R | Commanded position. |
| 2 | $\begin{aligned} & \text { SWLi } \\ & \text { mitPo } \\ & \mathrm{s} \end{aligned}$ | DINT | $\begin{aligned} & \hline-214748364 \\ & 7 \end{aligned}$ | 2147483647 | $\begin{aligned} & 214748 \\ & 3647 \end{aligned}$ | R/W | Positive Software limit switch position. |
| 3 | $\begin{aligned} & \text { SWLi } \\ & \text { mitN } \\ & \text { eg } \end{aligned}$ | DINT | $\begin{array}{\|l\|} \hline-214748364 \\ 7 \end{array}$ | 2147483647 | $\begin{aligned} & -21474 \\ & 83647 \end{aligned}$ | R/W | Negative Software limit switch position. |
| 4 | $\begin{aligned} & \text { Ena- } \\ & \text { bleLi- } \\ & \text { mitPo } \\ & \text { s } \end{aligned}$ | BOOL | FALSE | TRUE | FALSE | R/W | Enable positive software limit switch. |
| 5 | Ena-bleLimitN eg | BOOL | FALSE | TRUE | FALSE | R/W | Enable negative software limit switch. |
| 6 | $\begin{aligned} & \text { Ena- } \\ & \text { ble- } \\ & \text { Pos- } \\ & \text { LagM } \\ & \text { onitor } \\ & \text {-ing } \end{aligned}$ | BOOL | FALSE | TRUE | TRUE | R/W | Enable monitoring of position lag (following error). |
| 7 | Max-PositionL ag | DINT | 1 | ${\underset{*}{* * *}}_{2144483647}$ |  | R | Maximal position lag. |
| 8 | MaxVeloc itySyste m | DINT |  |  | 32767 | R | Maximal allowed velocity of the axis in the motion system. |
| 9 | MaxVeloc ityAp pl | DINT | 0** | 32767 | 32767 | R/W | Maximal allowed velocity of the axis in the application. |
| 10 | Actua IVelocity | DINT | -32767 | 32767 |  | R | Actual velocity. |
| 11 | Com- <br> man- <br> ded- <br> Veloc <br> ity | DINT | -32767 | 32767 |  | R | Commanded velocity. |
| 12 | Max- <br> Accel era-tionSyste m | DINT |  |  | 32767 | R | Maximal allowed acceleration of the axis in the motion system. |


| Param- <br> eter <br> numbe <br> r (PN) | Nam <br> e | Datatype | Min. | Max. | Default | R/W | Comments |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| 13 | Max- <br> Accel <br> era- <br> tio- <br> nAppl | DINT | 10 | 32767 | 32767 | R/W | Maximal allowed <br> acceleration of <br> the axis in the <br> application. |
| 14 | Max- <br> Decel <br> era- <br> tion- <br> Syste <br> m | DINT |  |  |  |  |  |
| 15 | Max- <br> Decel <br> era- <br> tio- <br> nAppl | DINT | 10 | 32767 | 32767 | R/W | Maximal allowed <br> deceleration of <br> the axis. |
| 16 | Max- <br> Jerk | DINT | $0 *$ | 2147483647 | 214748 | R/W | Maximal allowed <br> jerk of the axis. |
| 2001 | MOD <br> ULO <br> NO | DINT | 1 | 2147483647 | 1 | R/W | ABB specific <br> parameter. Used <br> for PLC-based |
| MIN- <br> ATO <br> R |  | Maximal allowed <br> deceleration of <br> the axis. |  |  |  |  |  |
| implementration: |  |  |  |  |  |  |  |
| Gearbox modifier |  |  |  |  |  |  |  |
| to |  |  |  |  |  |  |  |
| MODULO_RANG |  |  |  |  |  |  |  |
| E |  |  |  |  |  |  |  |$|$


| Param- eter numbe r (PN) | Nam <br> e | Datatype | Min. | Max. | Default | R/W | Comments |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 2004 | Ena-bleLi-mitAbort | BOOL | FALSE | TRUE | FALSE | R/W | Enable that software limit <br> switches will abort ongoing movement FALSE = Limits position and velocity, decelerates and shows a warning until the position limit is reached, then ERROR STOP TRUE = Switches off any ongoing motion and decelerates to the position limit, then ERROR STOP |
| 2005 | Ena-ble-LimtVeloc ity | BOOL | FALSE | TRUE | FALSE | R/W | If the velocity is limited the unmoved position will be covered whenever possible |
| 2006 | $\begin{aligned} & \text { SWLi } \\ & \text { mit2 } \\ & \text { DecP } \\ & \text { os } \end{aligned}$ | LREAL | $\begin{array}{\|l} \hline-214748364 \\ 7 \end{array}$ | 2147483647 | $\begin{aligned} & 214748 \\ & 3647 \end{aligned}$ | R/W | Used as end position for Ena-bleLimit2Decelerate |
| 2007 | SWLi <br> mit2 <br> DecN <br> eg | LREAL | $\begin{aligned} & \hline-214748364 \\ & 7 \end{aligned}$ | 2147483647 | $\begin{aligned} & 214748 \\ & 3647 \end{aligned}$ | R/W | Used as end position for Ena-bleLimit2Decelerate |
| 2008 | Max- <br> Posi- <br> tion- <br> GapL <br> L | LREAL | 0 | $\begin{aligned} & 2147483647 \\ & 00 \end{aligned}$ | 0 | R/W | Used to stop the ongoing movement if position is behind |

0* means: no limitation of jerk is performed.
**Axis will stay in stop.
***is modified by CMC_Axis_Control_Parameter, the max. Value is calculated in increments, the value which is delivered by ReadParameter will be given in [u].

In addition to the above parameters certain other operation can be done using the below parameters from the data type "Axis_Parameter"

| Name | Type | Initial | Comment |
| :---: | :---: | :---: | :---: |
| paraFilterVariant | INT |  | Filter for actual velocity $\begin{aligned} & 0=\text { PT1 } \\ & 1 \text { = LinearRegression } \end{aligned}$ |
| paraFilterTime | INT | 10 | Time in PLC cycles, used with paraFilterVariant |
| paraFilterForecast | INT | 0 | Time in PLC cycles, used with paraFilterVariant = 1 |
| paraReverseDirection | INT | 0 | Changes the direction for actual and reference positions based on the mode selected. <br> $0=$ normal direction <br> 1 = reverse input position <br> 2 = reverse output position and speed reference <br> 3 = reverse both |
| paraEarlyClosedLoop | BOOL | FALSE | TRUE: hold the position when Drive_Release is set (not wait for Drive_InOperation = TRUE) |
| paraLateOpenLoop | BOOL | FALSE | TRUE: hold the position until Drive_InOperation $=$ FALSE |

Limits
Table 762: Limitations for the inputs of PLCopen function blocks when used with CMC_Basic_Kernel

| Parameter | Min. | Max. |
| :--- | :--- | :--- |
| Velocity | 0 | $x$ |
| Acceleration, Deceleration | 0 | $x$ |
| Position | -2147483647 | 2147483647 |

## General restrictions

Restrictions for the available function blocks

- As buffered mode, MC_Aborting is realized as a default. This does NOT mean that the axis stops when another movement is started while an ongoing movement is still active. It means instead that the new movement will take control immediately and change the velocity to its own velocity by using its own acceleration or deceleration.
- The buffered mode MC_Buffered could be reached with using the axis state StandStill as enable signal for the Execute of the next block.
- From the Extended Inputs and Outputs at the function blocks, the following are not realized:
- BufferedMode: The realization just supports the MC_Aborting mode.
- The following Outputs at ReadStatus are not supported: ConstantVelocity, Accelerating and Decelerating.
- TorqueLimit for Homing function blocks.



MC_Aborting The diagram shows the behavior with BufferMode MC_Aborting, which is the only available Mode BufferMode. When the second Block is activated, it will take control and will continue on its own velocity. The velocity is changed by using the acceleration value from the second function block. The movement will not be stopped in between. The first function block shows CommandAborted when the second function block is activated.

MC_Buffered A behavior according to BufferMode MC_Buffered could be reached by using the Done output from the first function block to enable the Execute of the second function block.

## Behavior of the function block inputs and outputs

## General rules

Table 763: General rules

| Output exclusivity | The outputs Busy, Done, Error, and CommandAborted are mutually <br> exclusive: <br> Only one of them can be TRUE on one function block. If Execute <br> is TRUE, one of these outputs has to be TRUE. Only one of the out- <br> puts Active, Error, Done and CommandAborted is set at the same <br> time. |
| :--- | :--- |
| Output status | The outputs Done, InGear, InSync, InVelocity, Error, ErrorID and <br> CommandAborted are reset with the falling edge of Execute. How- <br> ever, the falling edge of Execute does not stop or even influence <br> the execution of the actual function block. It must be guaranteed <br> that the corresponding outputs are set for at least one cycle if the <br> situation occurs, even if execute was reset before the function block <br> completed. If an instance of a function block receives a new exe- <br> cute before it has finished (as a series of commands on the same <br> instance), the function block will not return any feedback, like Done <br> or CommandAborted, for the previous action. |
| Input parameters | The parameters are used with the rising edge of the execute input. <br> To modify any parameter, it is necessary to change the input param- <br> eter(s) and to trigger the motion again. |
| Missing input parame- <br> ters | According to IEC 61131-3, if any parameter of a function block input <br> is missing (open) then the value from the previous invocation of <br> this instance will be used. In the first invocation the initial value is <br> applied. |
| Position versus distance | Position is a value defined within a coordinate system. Distance is a <br> relative measure related to technical units. Distance is the difference <br> between two positions. |
| Sign rules | Velocity, Acceleration, Deceleration and Jerk are always positive <br> values. Position and Distance can be both positive and negative. |
| Error Handling Behavior | All function blocks have two outputs, which deal with errors that can <br> occur while executing that function block. These outputs are defined <br> as follow: |
| Error Rising edge of Error informs that an error occurred during the <br> execution of the function block. <br> ErrorID: Error number |  |
|  | Bera |


|  | The outputs Done, InVelocity, InGear, and InSync mean successful completion so these signals are logically exclusive to Error. <br> Types of errors: <br> - Function blocks (e.g. parameters out of range, state machine violation attempted), <br> - Communication, <br> - Drive Instance errors do not always result in an axis error (bringing the axis to StandStill). The error outputs of the relevant function block are reset with falling edge of Execute. |
| :---: | :---: |
| Function block naming | In case of multiple libraries within one system (to support multiple drive/ motion control systems), the function block naming may be changed to MC_FunctionBlockName_SupplierID. |
| Behavior of Done output | The outputs Done, InGear, InSync... are set when the commanded action has been completed successfully. With multiple function blocks working on the same axis in a sequence, the following applies: <br> When one movement on an axis is interrupted with another movement on the same axis without having reached the final goal, Done of the first function block will not be set. |
| Behavior of CommandAborted output | CommandAborted is set, when a commanded motion is interrupted by another motion command. The reset-behavior of CommandAborted is like that of Done. When CommandAborted occurs, the other output-signals such as InVelocity are reset. |
| Inputs exceeding application limits | If a function block is commanded with parameters which result in a violation of application limits, the instance of the function block generates an error. The consequences of this error for the axis are application specific and thus should be handled by the application program. |
| Behavior of Busy output | Every function block can have an output Busy, reflecting that the function block is not finished. Busy is SET at the rising edge of Execute and RESET when one of the outputs Done, Aborted, or Error is set. It is recommended that this function block should be kept in the active loop of the application program for at least as long as Busy is true, because the outputs may still change. For one axis, several function blocks might be busy, but only one can be active at a time. <br> Exceptions are MC_SuperImposed and MC_Phasing, where more than one function block related to one axis can be active. |
| Output Active | The output Active is required on buffered function blocks. This output is set at the moment the function block takes control of the motion of the according axis. For un-buffered mode the outputs Active and Busy can have the same value. |
| Enable and Valid/Status | The input Enable is coupled to output Valid. Enable is level sensitive, and Valid shows that a valid set of outputs is available at the function block. The output Valid is TRUE as long as an output value of Valid is available and the input Enable is TRUE. The relevant output value can be refreshed as long as the input Enable is TRUE. If there is a function block error, the output is not Valid (Valid set to FALSE). When the error condition disappears, the values will reappear and output Valid will be set again. |



Fig. 320: Behavior of the Execute/Done style function blocks.

Why is the command input edge sensitive?
The input Execute for the different function blocks described in this document always triggers the function with its rising edge. The reason for this is that with edge triggered Execute new input values may be commanded during execution of a previous command. The advantage of this method is a precise management of the instant a motion command is performed. Combining different function blocks is then easier in both centralized and decentralized models of axis management. The output Done can be used to trigger the next part of the movement. The example given below is intended to explain the behavior of the function block execution.
The following figure illustrates the sequence of three function blocks First, Second and Third controlling the same axis. These three function blocks could be for instance various absolute or relative move commands. When First is completed the motion its rising output First.Done triggers Second.Execute. The output Second.Done AND $\ln 13$ triggers the Third.Execute.


Fig. 321: Function blocks to perform a complex movement

## The input ContinuousUpdate

Like described in the previous chapter, the input Execute triggers a new movement. With a rising edge of this input the values of the other function block inputs are defining the movement. Until version 1.1 of PLCopen there was the general rule that a later change in these input parameters does not affect the ongoing motion.
Nevertheless, there are numerous application examples, where a continuous change of the parameters is needed. The user could retrigger the input Execute of the function block, but this complicated the application.
Therefore, the input ContinuousUpdate has been introduced. It is an extended input to all applicable function blocks. If it is TRUE, when the function block is triggered (rising Execute), it will - as long as it stays TRUE - make the function block use the current values of the input variables and apply it to the ongoing movement. This does not influence the general behavior of the function block nor does it impact the single axis state diagram. In other words it only influences the ongoing movement and its impact ends as soon as the function block is no longer Busy or the input ContinuousUpdate is set to FALSE.


If ContinuousUpdate is FALSE with the rising edge of the input Execute, a change in the input parameters is ignored during the whole movement and the original behavior of previous versions is applicable. The ContinuousUpdate is not a retriggering of the input Execute of the function block. A retriggering of a function block which was previously aborted, stopped, or completed, would regain control on the axis and modify its single axis state diagram. Opposite to this, the ContinuousUpdate only effects an ongoing movement. Also, a ContinuousUpdate of relative inputs (e.g. Distance in MC_MoveRelative) always refers to the initial condition (at rising edge of Execute).

## Example

- MC_MoveContinuousRelative is started at Position 0 with Distance 100, Velocity 10 and ContinuousUpdate set TRUE. Execute is Set and so the movement is started to position 100.
- While the movement is executed (let the drive be at position 50), the input Distance is changed to 130, Velocity 20.
- The axis will accelerate (to the new Velocity 20) and stop at Position 130 and set the output Done and does not accept any new values.


## Unit of length

The only specification for physical quantities is made on the unit of length (noted as [u]) that is to be coherent with its derivatives i.e. (velocity [u/s]; acceleration [u/s $\left.{ }^{2}\right]$; jerk $\left[\mathrm{u} / \mathrm{s}^{3}\right]$ ). Nevertheless, the unit [ $u$ ] is not specified (manufacturer dependent). Only its relations with others are specified.

## Aborting versus buffered modes

Some of the function blocks have an input called BufferMode. With this input, the function block can either work in a Non-buffered mode (default behavior) or in a Buffered mode. The difference between those modes is when they should start their action:

- A command in a non-buffered mode acts immediately, even if this interrupts another motion,
- A command in a buffered mode waits till the current function block sets its output Done (or InPosition, InVelocity...).
- The library just supports the mode "aborting" (MCAborting)

The following examples describe the different behavior of these modes:

## Example 1: <br> Standard <br> behavior of two <br> following abso- <br> lute move- <br> ments



Fig. 322: Basic example with two MC_MoveAbsolute on same axis


Fig. 323: Timing diagram for example above without interference between function block 1 and function block 2

## Example 2: <br> Aborting <br> motion




Fig. 324: Timing diagram for example above with function block 2 interrupting function block 1 (McAborting Mode)

If an on-going motion is aborted by another movement, it can occur that the braking distance is not sufficient due to deceleration limits.

In rotary axis, a modulo can be added. A modulo axis could go to the earliest repetition of the absolute position specified, in cases where the axis should not change direction and reverse to attain the target position.
In linear systems, the resulting overshoot can be resolved by reversing, as each position is unique and therefore there is no need to add a modulo to reach the correct position..

## PLCopen examples

Example: A The following figure shows an example where the function block (MC_MoveVelocity) is used
function block instance controls different motions of an axis
to control AxisX with three different values of Velocity. In a Sequential Function Chart (SFC) the velocity 10,20 , and 0 is assigned to V . To trigger the input Execute with a rising edge the variable E is stepwise set and reset.


Fig. 325: Single function block with SFC
The following timing diagram explains how it works:


Fig. 326: Timing diagram for a usage of single function block

The second InVelocity is set for only one cycle because the Execute has gone low before the ActualVelocity equals CommandedVelocity.

Example: Dif- Different instances related to the same axis can control the motions on an axis. Each instance ferent function will then be responsible for one part of the global profile.
blocks
instances control the
motions of an
axis


Fig. 327: Cascaded function blocks
The timing diagram:


Fig. 328: Cascaded function blocks timing diagram
A corresponding solution written in LD looks like:


Fig. 329: Cascaded function blocks with LD

### 6.5.10.3.5 PLC-based motion control

## PLC-based motion control architecture

With PS5611-Motion different motion control system structures are possible. Independent of the system structure a typical motion control application consists of the following system elements:

- An application program which contains PLCopen function blocks that defines the general application behavior and logics.
- A profile generator which generates a position profile based on the dynamic specifications of the application program to guide the axis to the desired positions.
- A position control loop which outputs a speed reference signal to minimize the following error.

To achieve the best system structure for an application these components can be separated into different devices. Each type of structure has its own kind of interface and type of signals which need to be transferred between the interacting devices.

> All shown motion control system structures (Central motion control with or without position control loop) can be combined together in the same application program for a motion control project.

With the function blocks of motion library a motion control profiler can be used inside the PLC. As shown in the following figure it is needed to provide the actual position of the drive. The output can be either a position or a velocity reference signal. The used output signal will then be used to move the axis in the desired way.


There are 2 possibilities to send a reference value to the drive:

- When the position control loop is closed by the PLC by a CMC_Basic_Kernel function block, the output Speed_Reference should be connected to the drive. The value of Speed_Reference can be scaled with the axis parameters Max_Rpm and Ref_Max.
- When the position control loop is closed by the drive, the output Position_Reference should be connected to the drive. The unit for the output Position_Reference is incremented as well as the input Drive_ActualPosition.


Fig. 330: Architecture for centralized motion control

In general the programming of a machine consists of two layers as shown in the figure above.
In the application layer function blocks according to PLCopen motion control are used to program the application sequences with all necessary types of movements and administrational commands. Due to the standard PLCopen motion control this can be reused in any other machine programs that used PLCopen function blocks.

The axis implementation layer is responsible for the execution of the commands from the application layer and can be programmed for each axis in a different way depending on the used hardware components.

Table 764: Needed function blocks for an application with PLC-based motion control

| Library | Content |
| :--- | :--- |
| ABB_MotionControl_AC500.library | Kernel function block, Parameters function <br> block, Axis Simulation function block |
|  | Data types for AC500 motion control |
|  | Motion control function blocks according to <br> PLCopen |



The library design is independent from any bus architecture or any specific drive features.

| Example for a possible system architecture | System | Velocity reference | Position feedback |
| :---: | :---: | :---: | :---: |
|  | System A | Output via analog output channel as voltage or current | From incremental encoder connected to CD522 I/O module |
|  | System B | Output via EtherCAT network | Input via EtherCAT network |
|  | System C | Output as frequency signal of CD522 I/O module | From incremental encoder connected to CD522 I/O module |
|  | System D | Output via PROFINET IO network | Input via PROFINET IO network |
|  | System E | Output via PTO \& PWM channel in eCo V3 | Input via either encoder (included in onboard IO), or the PTO or PWM pulse count. |

In case the velocity reference value is used from the kernel function block the position control loop is closed inside the drive. In this case, it is necessary to adjust the related parameters from the parameters function block. When the position reference will be used the position control loop is closed inside the drive. In this case, the internal control loop is just used to monitor the position and velocity.


The drive's status should be managed by a specialized function block that supports the used type of drive as shown in the figure above. The kernel function block is the main function block which is needed to operate an axis with PLC-based motion control. It must be used with the parameter function block which is the interface to input parameters which are used to setup the axis.


The drive has to be accessed outside the CMC_Basic_Kernel function block. Actual values and reference values might be transferred by a synchronized fieldbus or by I/Os. The function block CMC_Basic_Kernel has to be called every cycle and at least once before any function block MC or MCA is activated.
The following figure shows an example with an axis simulation. The main data signals are drawn in bold lines. Here, the drive will receive a speed reference signal which means that the position control loop is closed inside the PLC by the central motion function blocks. The time behavior of the simulated drive can be set by the parameter T1 at the axis simulation function block. If the time constant is to slow and the axis parameter Control_Time is too short the simulation axis will run into instability - like a real drive. Sample values ${ }^{\sharp}$ Chapter 6.5.10.3.5.2.3 "How to use the axis simulation" on page 4005


A different option to create a virtual or simulated axis is to engage the Enable_Virtual input at CMC_Basic_Kernel. This virtual axis will follow the speed reference without additional delay, whereas the CMC_Axis_Simu creates a first order delay.


The following figure shows an example with a CiA402 drive on an EtherCAT network. The main data signals are drawn in bold lines. Here, the drive will receive a position reference signal which means that the position control loop is closed inside the drive.


In the example the main signals are to be transferred via EtherCAT network. The drive control function block for the Microflex e190 can be found in the ABB_Ecat_CiA402_AC500.library.

If using the eCo V3 PLCs, use the OBIO_PTOMotionKernel function block (separate library ABB_MotionControlEco_AC500.library) instead of CMC_Basic_Kernel for the PTO functionality.

| OBIO_PTOMotionKernel |  |
| :---: | :---: |
| Enable BOOL | BOOL Error |
| Drive_Ref_Ok BOOL | ERROR_ID ErrorId |
| Drive_InOperation BOOL | BOOL Drive_Reset_Fault |
| Drive_ActualPosition DINTT | BOOL Drive_Release |
| Axis Axis_Ref | BOOL Drive_Set_Ref |
| Control_Parameter CMC_Axis_Control_Parameter | DINT Drive_Set_Position |
| IO CMC_Axis_IO | DINT Speed_Reference |
| Enable_Virtual BOOL | DIVT Position_Reference |
| PTO_Motion_Parameter OBIO PTO Motion_Parameter | AC500 OnboardiO.Error ID PTO_ErrorID |

In the eCo V3 PLC, if PWM is used in the configuration, use the kernel function block OBIO_PWMMotionKernel function block instead of CMC_Basic_Kernel function block.

## Basic functionalities

## How to connect a drive

The connection to a drive must be done with the inputs and outputs of the function block CMC_Basic_Kernel. All inputs and outputs of the kernel function block with the prefix "Drive_" are intended to be used with a drive, but in some cases not all of them are needed. In all cases the input Drive_ActualPosition has to be connected with the actual position of the axis. This value can be received by an I/O module of the PLC or via a fieldbus.

Depending on which device closes the position control loop either the output Speed_Reference or Position_Reference output has to be used. The value of Speed_Reference can be connected to an analog output module or be transferred via a fieldbus. The value of Position_Reference should be exclusively sent via a real-time fieldbus like EtherCAT.

## Example 1:

 Analog drive Motor with incremental encoderIn the example the position control loop will be closed by the PLC, therefore the input Drive_ActualPosition and the output Speed_Reference are to be used.
In combination with the I/O module CD522 and the corresponding function block CD522Encoder32Bit the position of the encoder can be used. For the effective resolution of the encoder parameter Inc_Per_R of the parameter function block has to be used.
The output Speed_Reference can be written directly to the global variable of an output channel of an analog module but can also be transferred via a fieldbus. The scaling of this output value can be done with the parameters Ref_Max and Max_Rpm of the function block CMC_Axis_Control_Paramter_Real.

The scaling of the Speed_Reference value can be set with the inputs Ref_Max and Max_Rpm of the parameter function block.


In order to finish a homing sequence which is done by the function block MC_StepRefPulse the outputs Drive_Set_Ref and Drive_Set_Position from the kernel function block have to be connected with the inputs EN_RPI and START_VALUE of the CD552 I/O module function block. Also the output RdyRpi of the CD552 I/O module function block has to be connected with Drive_Ref_Ok from the kernel function block.
To enable and disable the drive Drive_Release could be connected to a binary output to activate the drive. Drive_InOperation could be connected to a binary input to get the information that Drive_Release was successful.


## How to enable and disable a drive

In order to enable a drive the function block MC_Power has to be used within the applicational layer. The kernel function block will then, if possible, output a rising edge on the output Drive_Release which can be connected to the drive-control function block which performs the needed actions on the drives control word to enable the drive. As soon the drive states enabled, this signal can be connected to the input Drive_In_Operation of the kernel function block. The axis state according to the single axis state diagram of PLCopen will then switch from Disabled to Standstill.


Fig. 331: Enabling sequence of a drive

As long as the drive is in state Disabled or ErrorStop the input Drive_Actual_Position will be copied to the output Position_Reference of the kernel function block. The output Speed_Reference will be zero.

When the axis is in operation, which means it is not in state Disabled or ErrorStop, then the output Position_Reference will be calculated by the kernel function block and the position control loop will be closed, which outputs non zero value for the output Speed_Reference in case of a following error. The input Actual_Position should then follow the position reference. The difference of both values is the following error and will be supervised by the kernel function block.

In case of drive problem, Drive_InOperation should be reset. The function block will open the position control loop and Speed_Reference will be set to zero.
For the most drives the status is control by the drives control word whereas the drives status word represents its actual status. In order to enable the drive it might be necessary to pass through several drives states according a defined scheme which depends on the used drive. Therefore the library ABB_Ecat_CiA402_AC500.library is added to PS5611-Motion package which contains function blocks to operate with different drives on an EtherCAT network. There is also the PS5605-DRIVES library package which can be used to control the state of other ABB drives and other protocols.

## How to use the axis simulation

It is possible to use a simulated axis instead of a real drive.
The axis simulation can be used in the following use cases:

- When the real drive is not available the simulation can be used to test all available motion functionalities to verify the application program.
- The simulation can be used to create a virtual master axis and synchronize other axes to it.

The simulation is realized by the function block CMC_Axis_Simu or input Enable_Virtual = TRUE can be used at the KERNEL-block.

Homing will be possible if the limit-switches (data type CMC_Axis_IO) are simulated also. This is not done by CMC_Axis_Simu but could be realized in the PLC program.


Fig. 332: Example for Simulation
The drive velocity is simulated by PT1-Characteristic. The input T1 gives the time constant for this PT1 as multiple of the cycle time. All other properties are simulated according to the CMC_Axis_Control_Parameter.


#### Abstract

The value of the time behavior from the axis simulation function block set by the input $T 1$ has to be at least four times smaller than the value of the axis parameter Control_Time from the parameter function block. If Enable_Virtual = TRUE is used, no delay will be applied to the simulated drive speed, $\overline{\text { and }}$ it will not be possible to test the position-control loop, but it will be fine to be used as virtual axis.


## How to perform a homing

The homing of an axis is a procedure which consists of up to two phases. For each phase there are different function blocks available. The available function blocks are according to PLCopen and belong to the application layer.

Table 765: Overview of the available homing function blocks

|  | Phase 1 | Phase 2/Finish Homing |
| :--- | :--- | :--- |
| MC_StepAbsSwitch | X |  |
| MC_StepDirect |  | X |
| MC_StepLimitSwitch | X |  |
| MC_StepRefPulse |  | X |

In order to create a complete homing sequence one function block of each phase can be used.

First phase The used function blocks will change the axis state to Homing and will move the axis to approach installed limit switches or a dedicated absolute switch in the desired directions. No manipulation of a position value will be done in this phase. The use of function blocks of this phase is optional for a homing.

The signals of the installed limit switches have to be written to a variable of the data type CMC_Axis_IO.

Second phase Function blocks from this phase will also change the axis state to Homing if this has not already happen and will finish the homing. Therefore a new position will be set to the axis. The axis state will then switch back to Standstill.

The use of a function block of the second phase is mandatory for a homing.
In general with AC500 PLC-based motion control there are two position values: One position value will represent the encoder counts of a drive or the CD522 module which is connected to the input Drive_ActualPosition of the kernel function block. The other position is a user defined scaled unit which is used for PLCopen function blocks.
There are different ways to finish the homing by manipulate and adjust a position value. Which value should be manipulated depends on the used drive or module and its capabilities. See the following types $A, B$ and $C$.

Type A The user defined position unit will be changed only. The function block MC_StepDirect must be used here. This type of homing is less complex than the other types but also less precise.


Fig. 333: Homing Type A

Type B The Drive or the CD522 module will change its own position value, the encoder counts.


Application Layer
Axis Implementation Layer


Fig. 334: Homing Type $B$
The process will be started by the execution of the function block MC_StepRefPulse.
The axis will start to move.
The output Drive_Set_Ref of the kernel function block will then set the drive to sense for a digital signal. At the same time the kernel function block outputs a preset value which will replace the actual encoder count value at the moment the digital signal occurs.
This signal can be a Z-pulse of an incremental encoder but also any other signal from a sensor. This functionality may require a configuration of the drive or the CD522 module in order to be used.

In the same cycle when the new position value is set there also has to be a boolean signal stating a new position value at the input Drive_Ref_Ok of the kernel function block. The user defined position value will then be shifted accordingly.

Type C The encoder count position value will not be changed but involves registration capabilities of a drive or the CD522 module.


Fig. 335: Homing Type C
The process will be started by the execution of the function block ECAT_HomingOnTouchProbe_APP (ABB_Ecat_CiA402_AC500.library).

The axis will start to move.
The output Drive_Set_Ref of the kernel function block will then command the drive or the CD522 module to activate the Touch Probe functionality. This will configure the drive to latch a position at the moment a digital signal occurs. The digital signal can be a Z-pulse of an incremental encoder but also any other signal from a sensor. This functionality may require a configuration of the drive or the CD522 module in order to be used.
In combination with the latched position value there is a boolean signal which states that a new latch value has been received. In case of the module CD522 this encoder count position value has to be converted from encoder counts to equivalent user scaled units by the use of the function "CMC_Get_Units_From_Inc" (ABB_MotionControl_AC500.library) before it can be connected to the fünction block ECAT̄_HomingŌnTouchProbe_APP.
To manage the Touch Probe objects of a drive within the CiA402 profile (e.g. Microflex e190) the function block ECAT_HomingOnTouchProbe_APP (ABB_Ecat_CiA402_AC500.library) can be used. This will also cover the conversion from encoder counts to user scaled units.

At the end of the process the function block ECAT_HomingOnTouchProbe_APP will manipulate the user scaled position value according to the latched position from the drive and the users settings.

For further information see:
AN00220-001 - AC500 and MicroFlex e190 - EtherCAT Homing Methods

## How to Use a CAM curve

The CAM functionality is only available in combination with the kernel function block CMC_Basic_Kernel.

From Automation Builder 2.5 .0 onwards user can make use of inbuild Cam Configurator to generate Cam Table. For more details on how to use Cam Configurator please refer Automation Builder Help file.

It is recommended to use the CAM Editor from Automation Builder for those who are new to Cam table or to get the structure of the Cam Table. User can create the complete CAM Table using Cam Editor or can make a copy of CAM Table (IEC Code) and adapt it directly in the IEC code if needed.

Details on the CAM Table structure and different parameters to be considered while creating the CAM is described below.

General usage The usage of a CAM function is based on the following elements:

- CAM table defined with the data type MC_PProfile.
- An instance of the function block MC_CamTableSelect
- An instance of the function block MCA_Cam_Extra (optional)
- An instance of function block MC_CamIn
- An instance of function block MC_CamOut

The following steps are necessary to use a CAM table

1. Declare a CAM table as an array of the data type "MC_PProfile" in the program.
2. Write data to this array.
3. Use the address of the CAM table at the input "CamTable" of the function block "MC_CamTableSelect".
4. Execute the function block "MC_CamTableSelect" to process the data of the CAM table with the function block's input parameters
5. Additionally you can execute the function block "MCA_Cam_Extra" for optional parameters after the processing of the function block "MC_CāTTableSelect".
6. Execute the function block "MC_CamIn" to start the slave axis movement according to the CAM table data and parameters.
$\Rightarrow$ The axis will operate in the axis state "Synchronized Motion".
7. To leave the axis state you can execute the function block "MC_CamOut".
$\Rightarrow$ The axis state will switch to state "Continuous Motion" and maintains its last velocity as long as there is no other command.
8. You can also use any other motion command interrupt the "Synchronized Motion".

Preconditions to use
"MC_CamIn"
and
"MCA CamInDir
ect" function
blocks

## 1. Use no sync option

| Behavior | Precondition | Howto | Possible problems |
| :--- | :--- | :--- | :--- |
| Start directly with <br> cam table positions, <br> previous movement <br> will be aborted if <br> the master is out- <br> side the cam table <br> area, the slave posi- <br> tion is not changed <br> $=>$ velocity=0. | Move master and <br> slave to matching <br> positions (according <br> to cam table) before <br> MC_CamIn OR start <br> the cam with posi- <br> tions=0 and use <br> relative=TRUE OR <br> compensate the posi- <br> tion offset with Mas- <br> terOffset and Slave- <br> Offset (MC_CamIn). | Use MC_CamIn Mas- <br> terStartDistance =0 | A position jump, if pre- <br> condition is not met. |

2. Synchronize to a dedicated position while the master moves from (MasterSyncPosi-tion-MasterStartDistance) to (MasterSyncPosition)

| Behavior | Precondition | Howto | Possible problems |
| :--- | :--- | :--- | :--- |
| Sync is started <br> when the master <br> passes (MasterSync- <br> Position -MasterStart- <br> Distance). | Master position < <br> (MasterSyncPosition | Use MC_CamIn <br> -MasterStartDistance) <br> (for positive move- | MasterStartDis- <br> ments), position can <br> tance>0 <br> be outside the cam <br> table position range. <br> Slave has to be in <br> standstill. |

3. Synchronize while the master moves a certain distance (MasterStartDistance)
$\left.\begin{array}{|l|l|l|l|}\hline \text { Behavior } & \text { Precondition } & \text { Howto } & \text { Possible problems } \\ \hline \begin{array}{l}\text { Sync is started } \\ \text { from the actual posi- } \\ \text { tion and velocity }\end{array} & \text {-slave axis in standstill } & \text { OR } & \begin{array}{l}\text { Use MC_Cam_ID. } \\ \text { and "inSync" when } \\ \text { the master has } \\ \text { moved "MasterStart- } \\ \text { Distance". }\end{array}\end{array} \begin{array}{l}\text {-master already } \\ \text { moving in the right } \\ \text { direction. }\end{array} \quad \begin{array}{l}\text { Acceleration and } \\ \text { sition = TRUE (molocity } \\ \text { data structure, output } \\ \text { from MC_CamTable- } \\ \text { process not limited. }\end{array}\right]$.
4. Synchronize as fast as possible, master can move or standstill

| Behavior | Precondition | Howto | Possible problems |
| :--- | :--- | :--- | :--- |
| Use a given acceler- <br> ation and velocity <br> to move the slave <br> to the position which <br> matches the actual <br> master position. | The master position <br> should be within <br> the cam table posi- <br> tion range, actual <br> velocity=0 or actual <br> velocity <>0 is pos- <br> sible. | MCA_CamInDirect | The allowed acceler- <br> ation and velocity <br> must be higher than <br> required by the cam <br> movement, so the <br> slave axis can catch <br> up. |

It is important to give higher velocity and acceleration values for "MCA_CamInDirect" than to be expected during the cam movement!!

CAM table CAM data is done with one table (two dimensional - describing master and slave positions together).
The data of the elements (array of data type MC_PProfile) can either be assigned within the declaration or can be assigned during run time before the execution of the function block MC_CamTableSelect.

It can be filled with data in the following ways:

- To use a predefined variable list.
- To calculate the values within the program (before using the MC_CamTableSelect).
- To send values by any communication access to the PLC.

In order to use the new data it is necessary to execute the function block MC CamTableSelect again. In case the CAM table is executed the function block MC_CamTableSelect may not be executed.

Elements of the data type MC_PProfile $\stackrel{y}{4}$ Chapter 6.5.10.3.3.5 "Overview of data types" on page 3966

The inputs MasterSyncPosition and MasterSyncDistance of the function block MC_CamIn can be used to define a distance to synchronize the slave axis onto the CAM table during the start. In case master axis moves with negative velocity the parameter MasterSyncDistance can be negative. The MasterSyncPosition should always be within the range of the CAM table master position.
MasterSyncDistance $=0$ will deactivate the synchronization. In this case the slave axis should be moved on the CAM curve before MC_CamIn is executed, otherwise a following error can occure.


Fig. 336: CAM profile figure
The master position in the CAM table must be strictly monotonic rising.
The length of a CAM table is just restricted by the memory size of the PLC. When long tables are used, it is recommended to call CamTableSelect in a task with lower priority as it will need a considerable computing time.
It is possible to hold several CamTables as a pool and to switch from one to another. This has to be done at matching positions as no means for synchronization are available.

The offset and scaling values (except the time-scale) are transferred continuously. This will allow to follow a "Moving Target" by adjusting these values.

The parameters at MC_CamTableSelect, MC_CamIn and function and MCA_Cam_Extra also modify the behavior:

| Parameter MC_Cam- <br> TableSelect | Type | Default <br> value | Comment |
| :--- | :--- | :--- | :--- |
| MasterAbsolute | BOOL | FALSE | TRUE=Master_position from MC_PProfile <br> equals the master axis absolute position. <br> FALSE=CAM is executed relative to the <br> master axis actual position at start. |
| SlaveAbsolute | BOOL | FALSE | TRUE=interpolation_point from MC_PProfile <br> equals the slave axis absolute position. <br> FALSE=CAM is started from actual slave posi- <br> tion. The values "interpolation_point" are rela- <br> tive to the slave axis position at start. |
| iType | MC_ABB_- <br> iTypes_E <br> NUM |  | Interpolationtype. |
| Number_of_pairs | INT |  | Number of points used in TimePosition Array. |

$\left.\left.\begin{array}{|l|l|l|l|}\hline \begin{array}{l}\text { Parameter } \\ \text { MC_CamIn }\end{array} & \text { Type } & \begin{array}{l}\text { Default } \\ \text { value }\end{array} & \text { Comment } \\ \hline \text { MasterOffset } & \text { LREAL } & 0 & \begin{array}{l}\text { Just used with MasterAbsolute=TRUE, } \\ \text { ignored otherwise. } \\ \text { Used position for cam-table is: Master axis } \\ \text { position-Masteroffset. }\end{array} \\ \hline \text { SlaveOffset } & \text { LREAL } & 0 & \begin{array}{l}\text { Just used with SlaveAbsolute=TRUE, ignored } \\ \text { otherwise. Used position is slave axis posi- } \\ \text { tion=interpolation_point+Slaveoffset. }\end{array} \\ \hline \text { MasterScaling } & \text { LREAL } & 1 & \begin{array}{l}\text { The position used for interpolation is multiplied } \\ \text { by MasterScaling, e.g MasterScaling=2, the } \\ \text { scaled master will pass the position range with } \\ \text { double velocity and within the half distance } \\ \text { compared to its real velocity and position. }\end{array} \\ \hline \text { SlaveScaling } & \text { LREAL } & 1 & \begin{array}{l}\text { Interpolation result is multiplied by Slave- } \\ \text { Scaling, e.g SlaveScaling=2: Slave axis will } \\ \text { run twice the distance. }\end{array} \\ \hline \text { MasterSyncPosition } & \text { LREAL } & 0 & \begin{array}{l}\text { Start synchronization at master } \\ \text { axis position=MasterSyncPosition-Master- } \\ \text { StartDistance+MasterOffset, meet the CamT- } \\ \text { able at master axis position=MasterSyncPosi- } \\ \text { tion. } \\ \text { In case of MasterAbsolute=FALSE: start }\end{array} \\ \hline \text { MasterStartDistance } & \text { LREAL } & 0 & \begin{array}{l}\text { actualPosition+MasterSyncPosition-Mas- } \\ \text { terStartDistance", meet the CamTable at }\end{array} \\ \text { "actualPosition+MasterSyncPosition"!!! It is } \\ \text { just possible to use the "sync" mechanism } \\ \text { when the axis is in StandStill on start. }\end{array} \right\rvert\, \begin{array}{l}\text { A negative value will create a reverse syn- } \\ \text { chronization mode, which means the master } \\ \text { should move in negative direction to syn- } \\ \text { chronize. It is independent from the Reverse- } \\ \text { Bit which indicates how to end the movement. }\end{array}\right\}$

These 2 parameters are "extras" to be written with the MCA_Cam_Extra function. When the parameters are used, the MCA_Cam_Extra has to be called after the MC_CamTableSelect.

| Parameter <br> MC_CamIn | Type | Default <br> value | Comment |
| :--- | :--- | :--- | :--- |
| Periodic | BOOL | TRUE for <br> master <br> "Modulo", <br> FALSE for <br> master <br> linear axis | CamTable will not reach "EndOfProfile" but <br> will be repeated periodically. When the master <br> is a linear axis, it has to move forward and <br> backward within the CamTable position range, <br> but even when it leaves this position range, <br> the CamTable will stay active. |
| Reverse | BOOL | FALSE | Just necessary when a CamTable is <br> NOT "periodic" and will run in reverse <br> direction (master with negative velocity) <br> Reverse=FALSE, the CamTable is ready when <br> the master leaves the position range in posi- <br> tive direction, e.g. when it moves from 359 <br> to 0 on a rollover axes Reverse=TRUE, the <br> CamTable is ready when the master leaves <br> the position range in negative direction. |

## Example for

 CAM curve

In the example, the slave will run from 0 to 2000 while the master runs from 0 to 1000 . The slave will start and end with velocity $=0$, no matter which velocity the master has during start. The slave will reach the maximum velocity when it is at position 1000 and the master is at position 500.


## How to use an external axis

To use multiaxis PLCopen function blocks with an externally sensed axis as master axis the following structure can be used for the axis implementation:


Fig. 337: Structure synchronization to an external axis
The use of a feed forward filter function block is needed if the slave axis has to follow the position of the external axis. In this case there will be a time delay between sensing the position of the external axis and moving the follower axis along the sensed position. The filter function block will then add a certain distance to the external axis' position depending of its speed.

The filter function block MATH_LINEAR_REGRESSION from the library ABB_MathFunctions_AC500.library can be used here.

| MATH_LINEAR_REGRESSION |  |
| :---: | :---: |
| ENABLE BOOL | LREAL NEXT_VALUE |
| ACTUAL LREAL | DWORD NEXT_BINARY |
| HISTORY DIVT |  |
| FORECAST LREAL |  |
| POSITIVE_LIMIT DWORD |  |
| NEGATIVE_LIMIT DIVT |  |

Fig. 338: Filter function block to feed forward an externally sensed position
For an axis which is following the external axis, the value "mcActualValue" (from MC_Source enumeration) for the input "MasterValueSource" for multi-axis PLCopen function blocks has to be used.
When the filter function block MATH_LINEAR_REGRESSION is used to process an actual position, 2 different purposes are fulfilled:

- A jitter or noise can be compensated
- It is possible to calculate a forecast-position to compensate for a delay in position measurement

Process the actual position or any other master axis always before the slave axis.

Otherwise, an additional one cycle-delay is introduced.

The MATH_LINEAR_REGRESSION function block calculates the progress for a variable which is captured in equidistant periods of time and is assumed to follow a linear curve. It uses the Gauss "least squares" -algorithm to do so. The line is calculated in a way that the sum of squares for the distances from the measured points to the assumed straight line is minimized.
A noise or jitter influence of the value is compensated and a predictive value for the variable with an adjustable forecast horizon can be calculated.
Linear equation:

$$
\text { Line }[\mathrm{i}]=\text { gradient }{ }^{*} \mathrm{i}+\text { offset }
$$

Sum of squares:

$$
\text { sum }=\sum_{i=1}^{\infty \text { history }}(\mathrm{x}[\mathrm{i}]-\text { line }[\mathrm{i}])^{2}
$$

The gradient and offset for the line are calculated in a way that "sum" is minimized. Then these 2 values are used to calculate the forecast value:

$$
\text { NEXT_VALUE: }=\text { gradient * FORECAST }+ \text { offset }
$$

FORECAST=0 would mean: value right now, no future or past considered.
When the ACTUAL value is a modulo value, for example a single turn encoder or a rollover axis, this has to be considered in the calculation. The 2 input values POSITIVE_LIMIT and NEGATIVE_LIMIT can be used to configure this. They define the upper and lower limit for ACTUAL. Also, the NEXT_BINARY will as a result be limited to these borders.

## Example



Fig. 339: Next Value_Forecast

## How to use an encoder/drive with <> 32-bit position overrun

The incremental position as actual position at the function block CMC_Basic_Kernel is usually assumed as position with a 32-bit position overrun. As well as it is the reference position which is sent to the drive.

Any modulo-axis configuration should be done inside the PLC.
Some drives are requested to correct their positions themselves for a non-linear axis which should constantly run into the same direction.
In this case, the drive has to be configured as a modulo-axis and the function block CMC_Basic_Kernel needs some additional function blocks to create the 32-bit value ${ }^{4}>$ Chapter 6.5.10.3.5.6.5 "Roll-Over axis" on page 4031.


Fig. 340: Kernel
The function block CMC_Modulo2Binary will convert any position with any Modulo_Range to a 32-bit binary position.

The actual_position is assumed to run between 0 to Modulo_Range.
The actual_position should not change > 1/4 Modulo_Range between two scan cycles.
The function block CMC_Modulo2Binary will convert the 32-bit binary position reference from CMC_Basic_Kernel to a position reference which runs from 0 to Modulo_Range.

## How to do position correction "on the fly"

Sometimes it is required to have a position correction "on the fly". For example, it can happen that a position is wrong due to mechanical slip and that a switch which is passed by during the movement is used to capture a position value.

In other cases, it is required to synchronize the position to a print mark, so an actual_position has to be corrected, but not the movement of the printed material.

For both applications, the function block MCA_SetPositionContinuous can be used. It will use ramps and a limited velocity for the correction, so it will be tolerable to execute it during an ongoing movement and while the axis is activated in a multi-axis movement.


Fig. 341: MCA_Set_PositionContinuous_V3
The block can be used in any axis state except ERRORSTOP and HOMING.
Two different operation modes are possible:

1. SuperImp=FALSE

- The actual_position will be modified.
- The block will not cause any movement.
- If a PLCopen block in DISCRETE_MOTION (positioning) is active during the execution, this block will not reach Done as the actual_position is modified.
- If a slave axis is coupled to an axis while MCA_SetPositionContinuous is executed (with SuperImp=FALSE) it will follow.
- This mode is possible while the axis is in state DISABLED.

2. SuperImp=TRUE

- The actual_position will stay constant.
- A mechanical movement is executed (without changing the axis state machine).
- A slave axis will not follow.
- This behavior is similar to a superimposed movement.
- It is not possible when the axis is in state DISABLED.

The block can just be aborted by another MCA_SetPositionContinuous.

## How to limit the movement

It is possible to limit the movement by position (software limit switches) and by velocity. By default, no software limit switches are activated in PS5611-Motion. It is possible to activate them by accessing some PLCopen parameter.
The functionality described below is just available with linear axes.
$\left.\begin{array}{|l|l|l|l|l|l|l|l|}\hline & \text { Parameter } & \text { Data type } & \text { Minimum } & \text { Maximum } & \text { Default } & \text { R/W } & \text { Description } \\ \hline 2 & \begin{array}{l}\text { SWLi- } \\ \text { mitPos }\end{array} & \text { DINT } & \begin{array}{l}21474836 \\ 47\end{array} & \begin{array}{l}21474836 \\ 47\end{array} & \begin{array}{l}214748364 \\ 7\end{array} & \begin{array}{l}\text { R/W }\end{array} & \begin{array}{l}\text { Positive software } \\ \text { limit switch posi- } \\ \text { tion. }\end{array} \\ \hline 3 & \begin{array}{l}\text { SWLLi- } \\ \text { mitNeg }\end{array} & \text { DINT } & \begin{array}{l}21474836 \\ 47\end{array} & \begin{array}{l}21474836 \\ 47\end{array} & \begin{array}{l}214748364 \\ 7\end{array} & \text { R/W } & \begin{array}{l}\text { Negative soft- } \\ \text { ware limit switch } \\ \text { mosition. }\end{array} \\ \hline 4 & \text { BOOL } & \text { FALSE } & \text { TRUE } & \text { FALSE } & \text { R/W } & \begin{array}{l}\text { Enable positive } \\ \text { software limit } \\ \text { switch. }\end{array} \\ \hline \begin{array}{l}\text { EnableLi- } \\ \text { mitNeg }\end{array} & \text { BOOL } & \text { FALSE } & \text { TRUE } & \text { FALSE } & \text { R/W } & \begin{array}{l}\text { Enable negative } \\ \text { software limit } \\ \text { switch. }\end{array} \\ \hline 2003 & \begin{array}{l}\text { Enable- } \\ \text { Limit2Dec } \\ \text { elerate }\end{array} & \text { BOOL } & \text { FALSE } & \text { TRUE } & \text { FALSE } & \text { R/W } & \begin{array}{l}\text { Enable software } \\ \text { limit switches to } \\ \text { decelerate }\end{array} \\ \hline \begin{array}{l}\text { EnableLi- } \\ \text { mitAbort }\end{array} & \text { BOOL } & \text { FALSE } & \text { TRUE } & \text { FALSE } & \text { R/W } & \begin{array}{l}\text { Enable that soft- } \\ \text { ware limit } \\ \text { switches will } \\ \text { abort ongoing } \\ \text { movement } \\ \text { FALSE = Limits } \\ \text { position and } \\ \text { velocity, deceler- } \\ \text { ates and shows } \\ \text { a warning until } \\ \text { the position limit } \\ \text { is reached, then } \\ \text { ERROR STOP }\end{array} \\ \text { TRUE = } \\ \text { Switches off any } \\ \text { ongoing motion } \\ \text { and decelerates } \\ \text { to the position }\end{array}\right\}$

The following different behavior is possible:

- No limitation at all (default)
- Limit position with ERRORSTOP:
- Limit position between SWLimitNeg to SWLimitPos, axis to state ERRORSTOP in case the position range is left.
- Limit velocity and acceleration:
- Limit velocity to paraMaxVelocityAppl and acceleration/deceleration to paraMaxDecelerationAppl, create WARNING_VELOCITY, not state changes for axis, abort movement is optional when MaxPositionGap is reached due to limitation.
- Limit Position with ramp-down:
- In addition, it is possible to limit the position between SWLimit2DecNeg and SWLimit2DecPos. paraMaxDecelerationAppl is used to ramp down.

When activated with EnableLimitPos or EnableLimitNeg, the reaction will be as follows:

- When the control position reaches the respective limit switch, the axis will go to state ERRORSTOP, and Drive_Release will be switched off. The actual_position might be behind, depending on the following error. It is assumed that a drive or application specific braking is performed. The axis will be stopped behind the limit.
- The axis could be switched on again by MC_Power. A movement in the opposite direction will be possible.
- The functionality of EnableLimitPos and EnableLimitNegis unchanged.

You can use the limitation of movement to achieve a soft or adjustable braking in advance before reaching the software limit switch. The limitation is activated by three Boolean parameter and will calculate a position distance to the limit switch, which depends on the actual velocity and given deceleration ramp. "paraMaxDecelerationAppl" is used for deceleration. It will decelerate the axis by the given deceleration ramp when the calculated position is reached and stop at the software limit switch. The original behavior is not modified, so if also these software limit-switches are activated, the axis might be set to state ERRORSTOP.


There are 2 different modes:

- EnableLimitAbort = TRUE

Any ongoing motion will be aborted immediately (when the distance to stop is reached, as shown in the above diagram), a warning is shown
The axis will be decelerated to reach the software limit switch.

- EnableLimitAbort =FALSE, EnableLimitDecelerate=TRUE

A warning is shown and the velocity is reduced, with respect to the given deceleration and position limit.
The ongoing motion is not aborted. If it was just a "tight fit", e.g. in a master slave movement and the direction is turned soon enough, it might be possible to continue the movement.
As the ongoing movement is not interrupted, an activated movement might not be completed, for example a MC_MoveAbsolute will never reach its target position. A warning is shown at function block CMC_Basic_Kernel.
When EnableLimitPos = TRUE or EnableLimitNeg = TRUE, and the values for SWLimitPos or SWLimitNeg are set, the axis will be set to state ERRORSTOP when these position limits are reached.

In addition, the function block will allow to limit the velocity. With EnableLimitVelocity = TRUE, it will monitor the velocity demand from the position reference and limit the position reference, so the given velocity limit will not be exceeded. A warning will be shown. The velocity used for limitation is MaxVelocityAppl.

The velocity limitation can be used to prevent short-term velocity peeks. The limited position will be caught up later, whenever possible. This can result in not-expected behavior. The WARNING issued by CMC_Basic_Kernel can be checked and used to stop a movement. The movement will be aborted automatically when the position is by MaxPositionGap behind.

- For a single axis movement, the commanded velocity is limited at the beginning. No position gap will occur.
- In a multi-axis movement, the slave axis follows a master. This can result in a position gap. A velocity peek from the master axis can be reduced by using the limitation. If the master is too fast because of the value for MaxPositionGap, the movement will be aborted.

When EnableLimit2Decelerate or EnableLimitAbort are used, the velocity is limited to MaxVelocitySystem with EnableLimitVelocity = FALSE. The function modifies the position reference. This modified position reference is used to control the drive. Whenever the limitation interferes the kernel will show a warning or an error. The warning or error message will disappear when the situation is cleared.

| Parameter <br> Number | Parameter Name | Value | Comments |
| :--- | :--- | :--- | :--- |
| 4 | EnableLimitPos | TRUE | ERRORSTOP when positions <br> exceed, no previous warning <br> or deceleration. |
| 5 | EnableLimitNeg | TRUE |  |
| 2003 | EnableLimit2Decelerate | FALSE |  |
| 2004 | EnableLimitAbort | FALSE |  |
| 2005 | EnableLimtVelocity | FALSE |  |


| Parameter <br> Number | Parameter Name | Value | Comments |
| :--- | :--- | :--- | :--- |
| 4 | EnableLimitPos | FALSE/TRUE | Reduce the velocity when <br> reaching a position limit within <br> the deceleration distance cal- <br> culated by using MaxDeceler- <br> ationAppl. Display a warning <br> at CMC_Basic_Kernel. The <br> underlying movement stays <br> active. With EnableLimitPos <br> = TRUE or EnableLimitNeg <br> = TRUE: When the Position <br> limit is reached, the axis is <br> set to mode ERRORSTOP <br> also if EnableLimitPos or Ena- <br> bleLimitNeg are used. Other- <br> wise, just the movement is <br> limited, without affecting the <br> state machine. An activated <br> positioning movement will not <br> reach its target. Velocity is <br> limited to MaxVelocitySystem. |
| 5 | EnableLimitNeg | FALSE/TRU | TRUE |
| 2003 | EnableLimit2Decelerate | FALSE | FALSE |
| 2004 | EnableLimitAbort | EnableLimtVelocity |  |


| Parameter <br> Number | Parameter Name | Value | Comments |
| :--- | :--- | :--- | :--- |
| 4 | EnableLimitPos | FALSE/TRUE | Reduce the velocity when <br> reaching a position limit within <br> the deceleration distance cal- <br> culated by using MaxDeceler- <br> ationAppl. Display a warning <br> at CMC Basic_Kernel. The <br> underlying movement stays <br> active. With EnableLimitPos <br> =TRUE or EnableLimitNeg <br> TRUU: When the Position <br> limit is reached, the axis is <br> set to mode ERRORSTOP <br> also if EnableLimitPos or Ena- <br> bleLimitNeg are used. Other- <br> wise, just the movement is <br> limited, without affecting the <br> state machine. An activated <br> positioning movement will not <br> reach its target. Velocity is <br> limited to MaxVelocitySystem. |
| 5 | EnableLimitNeg | FALSE/TRU | EnableLimit2Decelerate |
| 2003 | EnableLimitAbort | TRUE | The active PLCopen function <br> block is aborted as soon <br> as the warning is issued. <br> With EnableLimitPos = TRUE <br> or EnableLimitNeg = TRUE: <br> When the Position limit is <br> reached, the axis is set to <br> mode ERRORSTOP. |
| 2004 | EnableLimtVelocity | FALSE |  |


| Parameter <br> Number | Parameter Name | Value | Comments |
| :--- | :--- | :--- | :--- |
| 4 | EnableLimitPos | --- | The velocity is checked and <br> also limited to the value Max- <br> VelocityAppl. A warning is <br> shown. The active movement <br> is not aborted. This function- <br> ality works independent from <br> software limit switches. |
| 5 | EnableLimitNeg | --- | --- |
| 2003 | EnableLimitDecelerate | --- | TRUE |

## Programming guidelines

To achieve the best results for motion control the actual position has to be transferred in best possible quality (with minimal jitter) to the PLC. The position feedback is expected to be in increments as the data type is a DINT.

The kernel function block (CMC_Basic_Kernel or OBIO_PTOMotionKernel or OBIO_PWMMotionKernel) has to be called every cycle and its task requires a fixed cycle time.

A variable of type Axis_Ref is used to connect to the PLCopen function blocks and their kernel function block.

The function block CMC_Axis_Control_Parameter has to be used for the axis configuration. ̌ Chapter 6.5.10.3.5.6 "Axis parameters" on page 4026
The signal of the limits switches and the absolute switch should be connected to the elements of the data type CMC_Axis_IO. The signal of the absolute switch must be TRUE in case the axis hits the sensor. The signal of a corresponding limit switch has to be true when the axis leaves the area surrounded by the limit switches. If needed the signal has to be inverted before it is connected to the elements of the data type.

Task configura- The kernel function block and the transfer of axis IO data should be processed in a cyclic tion task. This task should be as short and real-time as possible to achieve the best motion control performance. Always make sure Kernel function block is called at the highest priority task and other applications must be at a lower priority task.
In order to save PLC processing time the most PLCopen function blocks as well as the application logic can also be processed in a task which runs on a lower priority than the real-time task with the axis implementation as shown in the figure below.


## Optional: <br> Unsynchronized Task Execution

## Synchronized Task Execution

All PLCopen function blocks which must be called in the same task than the kernel function block:

- MC_CombineAxes
- MCA_MoveByExternalReference

In case the position reference is transferred to the drive the task of the axis implementation should be synchronized to the fieldbus cycle. The following figures show an example for EtherCAT:

Fig. 342: Task of axis layer


Fig. 343: Task of application implementation

Visualization
The structure of the position control loop is also as visualization element CMC_Visu_FB_Basic_Kernel. included in ABB_MotionControl_AC500.library. As placeholder, an instance of CMC_Basic_Kernel has to be used. The visualization shows all numbers as they are really used inside the block, the adjustment for different resolution or cycle times is already included.


## Kernel function block

Kernel Arithmetic

The "KERNEL" function blocks are available in two variants.
The OBIO_PTOMotionKernel / OBIO_PWMMotionKernel function blocks are solely to be used in eCo V3 CPUs and to make use of the integrated stepper-IO. It connects automatically to the internal IOs. Use the PTO or PWM specific kernel block based on your configuration ${ }^{*} \boldsymbol{y}$ Chapter 5.2.1.1.2.1 "Functionality" on page 196.

The CMC_Basic_Kernel block is designed to be used in any V3 PLCs and can either work with drives connected to a fieldbus or IOs.

| Topic | OBIO_PTOMotion- <br> Kernel/ <br> OBIO_PWMMotion- <br> Kernel | CMC_Basic_Kernel |
| :--- | :--- | :--- |
| Recommended PLC | eCo V3 PLC | All V3 PLC‘s |

How does the parameter for jerk influence the axis move-
ments


Fig. 344: Velocity reference with different jerk values
The diagram shows the result with different jerk values and the same velocity and acceleration. The time needed for acceleration with jerk $=0$ is:

Time1=velocity/acceleration=(20/100)s=0.2s
The additional time with jerk $=500$ will be:
Time2=acceleration/jerk=(500/100)s $=5 \mathrm{~s}$
So the total time is:
Time $=$ Time $1+$ Time2 $=0.2 s+5 s=5.2 s$
In the last example with jerk=100, the velocity and acceleration values are not reached.

Axis parameters

## General

The parameters for axis configuration and adjustment are set by the function blocks CMC_Axis_Control_Parameter.
Depending on the version of the kernel function block the corresponding version of the parameters function block has to be used. The instance will then be connected to the kernel function block by its instance name.

| Example | ControlParameter |
| :---: | :---: |
|  |  |

In the example the control structure is a simple position control loop with just proportional gain. When the application does not require minimized position following error it should be used this way as it is simple to adjust, robust and requires minimal performance. The proportional gain is then adjusted by Control_Time. Just change values at CMC_Axis_Control_Parameter when the position control loo $\bar{p}$ is open (Drive_Release=FALSE, the axis state is Disabled). The values are sending to the control loop whit a positive edge at "Enable". The CMC_Basic_Kernel function block needs to be already enabled.

## Supervision

Pos_Lag_Per- This parameter configures the position window for the supervision of the following error.
The default value is $150[\%]$. A value of $0[\%$ ] will deactivate the supervision function.
The size of the position window depends on the setting of the parameters Control_Time and Max_Rpm «y "Control_Time" on page 4028.
Position Window [Increments] = (Inc_Per_R) * (Max_Rpm/60) * (Control_Time/1000)
Position Window [Units] = (U_Per_Rev_Nominator/ U_Per_Rev_Denominator) * (Max_Rpm/60)

* (Control_Time/1000)

| Example | Position Window [Increments] $=(10000) *(6000 / 60) *(50 / 1000)=50000$ [Increments] |
| :--- | :--- |
| Position Window [Units] $=(1 / 1) *(6000 / 60) *(50 / 1000)=5[$ Units] |  |

A value of $100 \%$ will result in a position window which corresponds to the expected following error with the giving Control_Time at Max_Rpm. Therefore it is recommended to use values higher than 100[\%]. In case the parameter FF_Percentage is used smaller values can be used. If the supervised position window is exceeded the axis state will change to ERRORSTOP.

V_Check_Time After the configured time the drive's actual velocity has to be at least $50 \%$ of the commanded velocity. This function can also be used in case the Position Reference is transferred to the drive.
A value of 0 will deactivate this supervision function.
If the supervised velocity window is exceeded the axis state will change to ERRORSTOP.

## Position control loop



Fig. 345: Basic structure of position control loop

Control_Time The default value is 100 which leads to a proportional gain of 10 .

In case the value of Control Time is too short the position control loop will run into instability.

In case the position control loop is not used this parameter must not be set to 0 .


Fig. 346: Control Time and static following error in case the feed forward of velocity and the integrational part of the position control loop is not used.

The static following error depends on the axis velocity and can be calculated easily: Control Time multiplied by the axis velocity ( $p$ _error = v*CT ).
In general it should be aimed to reach a high position control loop gain with a short Control Time to achieve a small following error. As the reaction times take account in the possible Control Time of the complete system (parameters of the drive control loop, PLC cycle time as well as the communication fieldbus) should be considered.

As a basic rule the Control Time should be at least four times longer than the reaction time between the output of the Speed Reference and the input of actual position.


When the time Ts and Tt is measured, a control_time of 4 * (Ts + Tt) will result in an aperiodic damping of the position control loop. It is important to measure the values from inside the PLC (e.g. Trace) to have the complete reaction times included. Practical values for Control_Time might be from 50-500ms. The PLC cycle time as well as bus cycle times and mechanical reaction will influence the value.

FF_Percentage The default value is 0 .
In case a velocity feedforward has to be configured a value of up to 80 is recommended. For larger values than 80 the parameter Horizon needs to be used as the resulted position will overshoot otherwise.
A value of 100 adds a velocity to the Speed Reference output which corresponds exactly to the ongoing Position Reference value.


Integral_Part The integral part of the position control loop can be used to eliminate a permanent positioning error, e.g. in case of hanging loads.

The time value can be regarded as the time the integrator needs to sum up the input value to reach the same value for its output.

In case the Integral Part Time is too short the position control loop will run into instability.

Horizon A communication delay of the Speed Reference value to the drive system can cause an overshoot during positioning caused by the velocity feedforward gain.
This function will compensate this communication delay to prevent an overshoot by time shifting the signals Velocity Feed Forward and Position Reference relatively to each other.

The value of Horizon can be approximately assumed to be the time delay of the communication delay.

The delay time might be caused by the cycle time of the control loop and by any delay in sending the speed reference, delay in the drive to build up the torque and delay to receive the actual position. To overcome this delay, a Horizon > 0 might be used. The feed forward reference will be created in advance, while the proportional gain is applied to the original motion profile. The delay is then compensated.
This function should not be used if the feed forward parameter FF_Percentage is 0 .
A value of 0 will deactivate this function, which is the default value.
While this function is used, it will increase the needed PLC calculation time for this axis.


Fig. 347: Result with Horizon=0


Fig. 348: Result with Horizon>0

## PLC cycle time

Cycle
This parameter represents the cycle time in which the kernel function block of the axis is called. If the configured cycle time is not correct the resulting acceleration and speed of an axis will be not correct also.
In case the task execution of the axis is synchronized to a fieldbus (e.g. EtherCAT) the cycle time of the fieldbus has to be used.

## Roll-Over axis

If the Position Reference value is used, the drive must able to perform a position over-run after 32 bit. If the drive's position over-run is different, it can be adapted with the function blocks CMC_Binary2Modulo and CMC_Modulo2Binary from the library ABB_MotionControl_AC500.library. Incompatibility can cause an axis to trip after hours of operation.
The possible position following error has to be smaller the $1 / 2$ Modulo_Range. Make sure that the modulo range is large enough.

Position following error = (100-FF_Percentage) * Max_Rpm * Inc_Per_R * Control_Time/6000000. This is the maximum value at constant velocity.

En_Modulo With this parameter the axis can be configured as a roll-over axis.

Modulo_Range The modulo range will be defined in drive position counts (DINT). It will result that the scaled unit position which is used by the PLCopen function blocks will stay within the defined range.

## Example

```
En Modulo = TRUE
Mo\overline{dulo_Range = 20000}
Inc_Pe\overline{r_Rev = 10000}
U_Per_Rev Nominator = 360 (e.g. degree)
U_Per_Rev_Denominator = 1
```

The scaled unit's position will cover the range from 0 to 720 (degrees).
In some cases it is not suitable to set the modulo range of an application with the DINT value of the parameter Modulo_Range only. In such cases the parameters 2001 Modulo_Nominator and 2002 Modulo_Denominator can be used to scale the parameter Modulo_Range to a more precise value.
Parameter
Modulo_Nomi-
nator and
Modulo_Denom-
inator (sup-
ported with
CMC_Basic_Ker
nel)

These parameters can be used to modify the Modulo_Range in a way that fractions of an increment could be used for 1 modulo (=rollover) distance

- Default: Modulo_Nominator=1 and Modulo_Denominator=1: the actual position for an axis is limited between 0 and Modulo_Range increments.
- Limitations: Modulo_Range*Modulo_Nominator < 2147483647. Otherwise: default values will be used.
- When modifying these parameters, the position control loop should be opened.


## Example

```
En Modulo = TRUE
Mo\overline{dulo_Range = 1024}
Modulo_Nominator = 10
Modulo_Denominator = 3
Inc Pe\overline{r R = }1024
U_Pe\overline{r_Re}v_Nominator = 80*5*3
U_Per_Rev_Denominator = 10
```



Result of parameters Modulo_Range, Modulo_Nominator and Modulo_Denominator: The modulo range will cover one revolution of the toothed-belt wheel.

Result of parameters U_Per_Rev_Nominator and U_Per_Rev_Denominator: One scaled unit corresponds to one mm of the tooth belt.

## Example:

Gearbox 10.1

|  | Option1 | Option2 |
| :--- | :--- | :--- |
| En_Modulo | TRUE | TRUE |
| Modulo_Range | 10240 | 10240 |
| Modulo_Nominator | 1 | 1 |
| Modulo_Denominator | 1 | 1 |
| Inc_Per_R | 1024 | 10240 |
| U_Per_Rev_Nominator | 36 | 360 |
| U_Per_Rev_Denominator | 1 | 1 |
| Max_Rpm | 3000 | 300 |

The two options above describe exactly the same configuration. The Modulo_Range is equivalent to 10 motor revolutions and is 10240 increments. For the position, 14 means $1^{\circ}$ and the resolution is $360^{\circ} / 10240 \mathrm{inc}=0,035^{\circ} / \mathrm{Inc}=1^{\circ} / 28,44 \mathrm{Inc}$.

| Example: |  |  |  |
| :--- | :--- | :--- | :--- |
| Gearbox $\mathbf{1 0 . 3}$ |  | Option1 | Option2 |
|  | En_Modulo | TRUE | TRUE |
|  | Modulo_Range | 1024 | 10240 |
|  | Modulo_Nominator | 10 | 1 |
|  | Modulo_Denominator | 3 | 3 |
|  | Inc_Per_R | 1024 | 10240 |
|  | U_Per_Rev_Nominator | 108 | 1080 |
|  | U_Per_Rev_Denominator | 1 | 1 |
|  | Max_Rpm | 3000 | 300 |

The two options above describe exactly the same configuration. The gearbox is $10: 3$, so the Modulo_Range is equivalent to $1024 * 10 / 3=3413+1 / 3$ increments. For the first option, the resulting modulo range is calculated $1024^{*} 10 / 3$, for option2, it is $10240^{*} 1 / 3$. For the position, 1u means $1^{\circ}$ and the resolution is $108^{\circ} / 1024 \mathrm{inc}=0,105^{\circ} / \mathrm{Inc}=1^{\circ} \% .481 \mathrm{Inc}$.

## Scaling of the unit of length

Inc_Per_R With this parameter the number of the drive position counts each revolution of the motor (DINT) have to be entered.

U_Per_Rev_Den ominator \& U_Per_Rev_No minator

With these two parameters the number of units which correspond to one revolution of the motor have to be entered.
The units of length can be scaled to values like: mm, inch, degree, ...
All dynamic parameters of the PLCopen function blocks like velocity, acceleration and jerk are based on seconds. Velocity [units/s], acceleration [units/s²], jerk [units/s ${ }^{3}$ ]

Example 1

```
Inc_Per_Rev
    = 10000
U_Per_Rev_Nominator = 360
U_Per_Rev_Denominator = 1
```

This will scale one unit to one degrees of the motor shaft. Correspondingly a velocity [units/s] of 360 will turn the motor shaft one revolution per second.

Example 2 In the example one unit will be scaled to one millimeter of the conveyor.


Fig. 349: Scaling units
How many units will pass after one revolution of the motor? $(80 * 5 \mathrm{~mm}) / 5=80$

```
Inc Per Rev
    = 1024
U_Pe\overline{r_Rēv Nominator = 80}
U_Per_Rev_Denominator = 1
```

Example 3 In the example one unit will be scaled to one millimeter of the conveyor.


Fig. 350: Scaling units
How many units will pass after one revolution of the motor? $(80 * 5 \mathrm{~mm}) / 32=12,5=125 / 10$

```
Inc_Per_Rev
    = 1024
U_Pēr_Rēv_Nominator = }12
U_Per_Rev_Denominator =}1
```


## Scaling of the speed reference output

These two parameters are used to scale Speed Reference output of the kernel FB in order to reach the intended velocity by the output value and to limit the highest possible output value.

Ref_Max Highest possible output value of the Speed Reference output. The Speed Reference value that corresponds to the parameter Max_Rpm should be used.

Max_Rpm Maximum speed of the motor in revolutions per minute.
Example - Analog Drive: 1000 rpm at 2 Volts, 3200 rpm at 6,4 Volts (max.)

- Analog output module: 10 Volts output at digital value 27648
- Ref_Max = 17695 ( $=27648 / 10$ * 6,4 )
- Max_Rpm = 3200


## Access and modify parameters

All modifications will be effective immediately. There is no extra plausibility check and values are not checked for limitations.

Use this functionality with care.

Some parameters are collected inside a structure in Axis_Ref, and can be accessed and modified immediately. They are the same parameters as used with function blocks MC_WriteParameter and MC_ReadParameter $\stackrel{y}{ }{ }^{\circ}$ Chapter 6.5.10.3.4.8 "PLCopen parameter" on page 3977.
The differences are:

- Only available with CMC_Basic_Kernel
- The parameter values are LREAL instead of DINT and can be used with decimals.
- The parameters will be effective immediately.
- There is no check for consistency or limits.
- The parameters for position control can be checked and modified by accessing the structure Axis_Parameter.CMC_Pos_Control in addition.

| Parameter for position con- <br> trol | Description |
| :--- | :--- |
| KP | Proportional gain in positive direction. Used directly to multiply <br> the following error and create the Reference_Prop. |
| KF | Feed forward in positive direction. Used directly to multiply the <br> speed reference and create the Reference_FF. |
| KP_BACK | Proportional gain in negative direction. Used directly to multiply <br> the following error and create the Reference_Prop. |
| KF_BACK | Feed forward in negative direction. Used directly to multiply the <br> speed reference and create the Reference_FF. |
| TI | Integration time. When parameter is used the position con- <br> trol loop has an additional integral part. In TI cycle, the Ref- <br> erence_ITG will reach the value of Reference_Prop, when <br> KI=100*KP. |
| KI | Proportional gain, used for integral part of position control loop. |
| KF_100 | Value for feed forward gain, if 100\% would be used. |
| Max_Time | Delay time used for supervision of velocity. With Max_Time=0, <br> no supervision is executed. |
| D_XS_Max | Maximum possible velocity in [u/cycle]. <br> The maximum allowed following error is part of the parameter <br> structure, PLCopen parameter paraMaxPositionLag. |
| Ref_Max | Limit for Speed_Reference. |

Element actual of Axis_Ref

The element actual represents actual values from inside the position control loop.

| Value | Description |
| :--- | :--- |
| Position | Actual position in [u] to control the axis. |
| Control_Position | Reference position in [u] which is actually used for control <br> loop. |
| D_XS | Distance in [u] to be moved per cycle. |
| D_XSS | Following error in [u]. |
| Reference_Prop | Proportional part for Speed_Reference. |
| Reference_FF | Feed forward part for Speed_Reference. |
| Reference_ITG | Integral part for Speed_Reference. |

Possible to use See parameter KP/KP_BACK and KF/KF_BACK. different gain for forward/ backward movement, possible improvement for hydraulic axis or vertical movement

Limitation for velocity and

From library version 3.1 on, these values are not limited to the 16-bit range of values (32767). The limit for velocity is calculated by the values given at CMC Axis Control Parameter and the acceleration and acceleration is limited such that this velocity can not be reached faster than $\overline{1}$ cycle.
deceleration

## ABB specific data structures

## General

Not all data structures are defined by PLCopen. Some specific structures are described in the following chapter. In addition to the data in these arrays, the movement is modified by offset and scaling values at the respective function block. These offset and scaling values (except the time-scale) are transferred continuously. This will allow to follow a "Moving Target" by adjusting these values.

## PositionPositionProfile

The data type MC_PProfile is used for CamTable. An array has to be defined and provided at MC_CamTableSelect. Several CamTables could be defined and the axis could change between them on the fly. There is no routine of smooth movement from one table to the next so the user has to take care just to switch on appropriate positions. Details are described in the documentation included with the library.

## Declaration example <br> CAM_table

```
ARRAY[1..3] OF MC_PProfile:=
        (Master_positíon:= 0 ,interpolation_point :=
0 ,Velocíty_ratio:= 0 ,Acceleration_ratio:= 0 ),
    (Master_position:= 50 ,interpolation_point :=
25 ,Velocity_ratio:= 0 ,Acceleration_ratio:= 0 ),
(Master_position:= 100 ,interpolation_point :=
0 ,Velocity_ratio:= 0 ,Acceleration_rätio:= 0 );
```


## PositionTimeProfile

This structure is used for time based profiles, e.g. MC_PositionProfile:

## Interpolation types for profiles

The curves defined by an array of MC_PProfile hold master position points and according slave positions. When the master position is between 2 points, the according position for the slave is interpolated. Different types of interpolation are possible. The type is defined in MC_ABB_iTypes_Enum . The master could be a real axis or some virtual axis which could be created by just writing values for position and velocity to the Axis_Master variable as shown in the example. The same interpolation types could be used on MC_TProfile.


Table 766: Overview of different interpolations

| Interpolation Types | Results in | Requires |
| :--- | :--- | :--- |
| MCA_LINEAR | Linear interpolation <br> with constant velocity <br> between interpolation <br> points. | profile.MC_PProfile_Array[x].master_posi- <br> tion, profile.MC_PProfile_Array[x].interpola- <br> tion_point |
| MCA_SPLINE_NAT- <br> URAL | Cubic spline interpola- <br> tion without jerk. | profile.MC_PProfile_Array[x].master_posi- <br> tion, profile.MC_PProfile_Array[x].interpola- <br> tion_point |
| MCA_SPLINE_COM- <br> PLETE | Cubic spline interpola- <br> tion without jerk, start <br> and end of profile with <br> velocity=0. | profile.MC_PProfile_Array[x].master_posi- <br> tion, profile.MC_PProfile_Array[x].interpola- <br> tion_point |
| MCA_POLY3 | Polynomial interpola-- <br> tion with linear velocity <br> between interpolation <br> points. | profile.MC_PProfile_Array[x].master_posi- <br> tion, profile.MC_PProfile_Array[x].inter- <br> polation_point, profile.MC__PPro- <br> file_Array[x].velocity_ratio |
| MCA_POLY5 | Polynomial interpola- <br> tion with linear accel- <br> eration between inter- <br> polation points. | profile.MC_PProfile_Array[x].master_posi- <br> tion, profile.MC_PProfile_Array[x].inter- <br> polation_point, profile.MC_PPro- <br> file_Array[x].velocity_ratio, <br> profile.MC_PProfile_Array[x].accelera- <br> tion_ratio_ |

The interpolations allow to run on smooth curves without the need to define a large number of points. The following chapter shows the results with different interpolation modes for a sinus-curve with 10 interpolation points. The following table gives the mean deviation.

| Interpolation Type | Mean deviation [ppm] |
| :--- | :--- |
| MCA_LINEAR | $19686=1.9 \%$ |
| MCA_SPLINE_NATURAL | $151=0.0151 \%$ |
| MCA_SPLINE_COMPLETE | $25510=2.5 \%$ |
| MCA_POLY3 | $131=0.0131 \%$ |
| MCA_POLY5 | 0.37 |

The original curve is represented by y_sinus for position and v_sinus for velocity. The diagrams show the result which is achieved by different interpolation types.

## MCA_LINEAR



Fig. 351: Results from linear interpolation
The velocity is constant between the interpolation points.
MCA_POLY3


Fig. 352: Results from polynomial interpolation
The result looks almost identical to the original curve. The mean deviation shows that MCA_POLY3, MCA_POLY5 and MCA_SPLINE_NATURAL produce results which follow the original curve really good and are almost identical. The spline interpolation produces a jerk-free curve without the need of providing velocity values and acceleration values in advance.

## MCA_COMPLETE



Fig. 353: Results from complete spline interpolation
In the beginning and the end, the curve does not follow the original curve. The reason is that it starts with velocity $=0$ and produces a jerk free result.
So the favoured result has to be considered in advance to choose the right interpolation method. With these different methods it is not necessary to provide a large number of interpolation points to get good results and smooth acceleration and deceleration ramps.

## PLC-based motion control - Load control / fluid power extensions

The ABB_MotionControlLoad_AC500 library is an extension to ABB_MotionControl_AC500 library based on PLCopen part 6 called "fluid power" and basically can be used to implement load control as a simple form of torque profiling. It can be used together with all other motion control package libraries. The same structure and general rules are applied and all the above chapters in this document is relevant for ABB_MotionControlLoad_AC500 library as well. It is recommended to read through all the above chapters before start using the function blocks from this library. A difference is that the position control loop has to be closed inside the PLC as it is to be synchronized with the load control loop which is also realized. The implementation of Load function blocks is based on the PLCopen part 6 - Fluid power.
Overview of the defined extended function blocks:

Table 767: Overview of the defined function blocks

| Administrative |  | Motion |  |
| :--- | :--- | :--- | :--- |
| Single axis | Multiple axis | Single axis | Multiple axis |
| MC_LimitLoad | - | MC_LoadControl | - |
| MC_LimitMotion | - | MC_LoadSuperlm- <br> posed | - |
| - | - | MC_LoadProfile | - |
| - | - | MC_TorqueControl | - |

> As per PLCopen MC_TorqueControl is a part 1 function block, however due to its implementation as a wrapper for the load control and limit load blocks this is added to ABB_MotionControlLoad_AC500 library.

The following state diagram is based on the version as defined in 'Part 1 - Function Blocks for Motion Control', Version 2.0.

This specification adds three load function blocks to the state diagram:

- MC_LoadControl
- MC_LoadSuperImposed
- MC_LoadProfile

MC_TorqueControl function block also follows the same state diagram.
Function blocks not listed in the state diagram do not affect the state diagram, meaning that whenever they are called the state does not change.

The state diagram shows synchronized motion because the position-axis follows the load, and the state is related to the position axis.


Note 1: From any state. An error in the axis occurred.
Note 2: From any state. MC_Power.Enable = FALSE and there is no error in the axis
Note 3: MC_Reset and MC_Power.Status = FALSE
Note 4: MC_Reset and MC_Power.Status = TRUE and MC_Power.Enable = TRUE
Note 5: MC_Power.Enable = TRUE and MC_Power.Status = TRUE
Note 6: MC_Stop.Done = TRUE and MC_Stop.Execute = FALSE

## Kernel function block - Fluid power

The basic block is the CMC_Load_Motion_Kernel. It has to be called every cycle and at least once before any MC... block is activated. It is used to combine the position and velocity functionality from CMC_Basic_Kernel with the load control functionality which is utilized by the MC_Load... blocks.



The reference which is used by the CMC_Load_Motion_Kernel is equivalent with the Speed_Reference at CMC_Basic_Kernel, as long as no LOAD-functionality is activated. The documentation from CMC_Basic_Kernel applies to the identical inputs and outputs. Some inputs and outputs are added to serve the load control functionality.

The Load_Ref is used instead of Axis_Ref for the MC_Loadxxx blocks. When the CMC__Load_Motion_Kernel is used, Load_Ref replaces Axis_Ref and user can use āll PLC̄open-BTocks.


The actuator (drive) has to be accessed outside the CMC_Load_Motion_Kernel block. Actual values and reference values might be transferred by a synchronised bus or by l/Os.

- •All inputs and outputs of the function block which are named "DRIVE_xxxx" should be used to connect to the actuator (drive). It does not matter whether this connection is done by fieldbus or by conventional IOs.
The Axis-structure is used to connect to the PLCopen blocks
- The Load_Axis structure is used to connect the fluid-power PLCopen blocks
- The control_parameter-structure is used for configuration of control loop.
- The IO-structure gives a connection to limit- or reference switches.

When the function block will take control (close loop) the output "Drive_Release" is set. The PLC-Program should then start the actuator (actuator (drive)) and set "Drive_InOperation = TRUE" when successful. In case of actuator (actuator (drive)) problem, "Drive_InOperation" should be reset. The function block will then open the position control loop and Speed_Reference will be 0 .


The homing is done with PLCopen-blocks. As the interface to the actual position is outside the CompactMotion, the bit "Drive_Set_Ref" is set when the state is reached to evaluate the zerotrack. When the zero-track was found, Drive_ActualPosition has to be set to "Drive_Set_Position", this has to be indicated by "Drive_Ref_Ok".
The output "Drive Reference" should be send to the actuator (drive). This value is scaled with Max_Rpm and Max_Reference which means: when "Drive_Reference" equals Max_Reference, the motor is expected to run with Max_Rpm.

Load control The function block holds a position control loop and a load control loop. The load control loop is a PIDT1-Block. Both control loops are alternately activated, depending if a MC_Load..block or a MC_Move... block is active. There is a bumbless transition realized between the different control loops.
The PIDT1 controller has a proportional, integral and derivative part. The integral and derivative part can be switched of by using a time value $=0$.

Transfer function


Control algorithm: Simple rectangle rule:

| KP**D | KP | XD |  | T1_TZ |  | 1 | TV | KP |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $Y=$ | + | * | $+\mathrm{YI}(\mathrm{z}-1)$ | + | * (YDTI(z-1) | $+$ | * | - | * (XD - XD(z-1))) |
| 100 | 100 | TN_TZ |  | 1+(T1_TZ) |  | T1_TZ | TZ | 100 |  |

Where:
$\mathrm{Yl}(\mathrm{z}-1)$ : The integral portion from the previous program
(YDTI(z-1) The differential portion from the cycle previous program cycle
$X D(z-1)$ : Control system difference from the previous program cycle

All 3 parts of the control loop are added up. The integral or derivative part could be disabled by setting the respective time constant to 0 , so the following structures are possible:

- $P$
- PDT1
- PI
- PIDT1

The Load_MaxRef and Load_MinRef values will limit the controllers output Y and also apply to the controller's internal integral part. I.e the integral part can only hold values between the high and low limits. If the manipulated variable $Y$ reaches one of the two limits, the controller's integral part is no longer changed. This prevents the integral part from holding meaningless values and, in certain circumstances, not returning to the operating range for a long time. This behavior of a controller is also referred to as a "special anti-reset windup measure".

## Example - Fluid power extensions

MC LimitLoad
In the diagram below, an example is explained. SFC is used here to distinguish between a movement where the MC_LimitLoad functionality has become 'Active' or not. In Step 2 there is a movement like 'MoveA ${ }^{\text {bibsolute', which is limited by the MC_LimitLoad functionality. If the }}$ absolute position is reached without MC_LimitLoad becoming active, the transition via done to step 3 is applicable. However, if the MC_LimitLoad becomes 'Active', the transition to the 'Halt' step is applicable, issuing a MC_Halt.


Fig. 354: MC_LimitLoad used in SFC

MC_LimitMotion e.g. force fitting

The function block is intended to be used in conjunction with a MC_LoadControl or MC_TorqueControl having primary control on the axis. The MC_LimitMotion should be enabled by the 'Active' output of the MC_LoadControl / MC_TorqueControl. If motion values on the axis exceed the given limit, appropriate measures are taken to keep to these limits, implying that the load/torque will not follow the programmed trajectory but depend on the external load conditions. However, the 'Active' output of the MC_LoadControl/MC_TorqueControl will stay TRUE in this case, following the modified PLCopen definition "The 'Active' output indicates, that the FB has control on the set-value generation of the axis". This is despite the fact, that physically only the load-conditions or the movement of an axis can be controlled. With actual motion states below programmed limits, the programmed load/torque trajectory will proceed. Enabling the limiter block with activation of the MC_LoadControl/MC_TorqueControl ensures that limits are only supervised when the MC_LoadControl/MC_TorqueControl takes control on the axis for the first time. Disabling the limiter block with de-activation of the MC_LoadControl/MC_TorqueControl ensures that limits are no more supervised when the MC_LoadControl/MC_TorqueControl loses control on the axis by 'CommandAborted' or 'Error'.

MC_ LoadSuperImposed

Possible Application: Actuator: hydraulic cylinder with fluid pressure sensor actuates the press of plastic injection molding machine in a continuous load operation.
Request: Prior to MC_LoadSuperImposed call, a MC_LoadControl block is 'Active' with a command of $7,500 \mathrm{kPa}$ to press melted plastic into the mold. Once the MC_LoadControl 'InLoad' condition is achieved a superimposed pressure of $5,000 \mathrm{kPa}$ is added several times to cause a hammering effect to relieve stresses in the plastic.

Result: the MC_LoadControl pressure command of $7,500 \mathrm{kPa}$ is superimposed with a discrete pressure command of $5,000 \mathrm{kPa}$. Once the 'LoadSuperlmposed' command is active the system pressure rises to $12,500 \mathrm{kPa}$.

When the superimposed pressure command has been achieved the MC_LoadSuperlmposed block is done and the original command given by the MC_LoadControl resumes the original pressure command. The MC_LoadSuperlmposed block is executed several times without affecting the original pressure command given by the MC_LoadControl block.


MC_TorqueControl

The example (below) opposite signs for 'Direction' \& 'Torque' are used (e. g. Retention or brake control). (In the function block: +Direction -Torque). It is like an unwinding application with torque on the material, and a break in the material. When the material breaks, as shown in the middle of the figure this causes a drop in the real Torque value (in absolute terms): The velocity will decrease, limited by the fastest "deceleration" limit specified by the 'Deceleration' VAR_INPUT down to zero velocity (with no tension there is a risk of having shock breakings, so we have to limit to the fastest). In this case the torque setpoint might not be achieved.


In an unwinding application (derived from this brake control) material tension is the target, not motor torque. The instantaneous diameter of the roll should be taken into account to transform the "User tension setpoint". Also, additional inertia compensation by modification of the torque setpoint for acceleration / deceleration is common from instantaneous weight data (weight is commonly estimated from diameter). Additionally, in unwinding applications, in the case of loose material (same condition as material break), a negative slow velocity reference is usually applied to "rewind" the loose material. In this case, this must be provided by external programming.

## Appendix

List of all PLCopen and ABB specific function blocks in PS552-MC (for V2 PLC) and PS5611Motion (for V3)

| SNo | Funktion block <br> type | Funktion block name | Motion Library <br> V2 (PS552-MC) | Motion Library <br> V3 <br> (PS5611- <br> Motion) |
| :--- | :--- | :--- | :---: | :---: |
| 1 | PLCopen | MC_Power | x | x |
| 2 | PLCopen | MC_Home | x | - |
| 3 | PLCopen | MC_Stop | x | x |
| 4 | PLCopen | MC_Halt | x | x |
| 5 | PLCopen | MC_MoveAbsolute | x | x |
| 6 | PLCopen | MC_MoveRelative | x | x |
| 7 | PLCopen | MC_MoveAdditive | x | x |
| 8 | PLCopen | MC_MoveSuperImposed | x | x |


| SNo | Funktion block type | Funktion block name | Motion Library V2 (PS552-MC) | Motion Library V3 (PS5611Motion) |
| :---: | :---: | :---: | :---: | :---: |
| 9 | PLCopen | MC_HaltSuperlmposed | x | x |
| 10 | PLCopen | MC_MoveVelocity | x | x |
| 11 | PLCopen | MC_MoveContinuousAbsolute | x | x |
| 12 | PLCopen | MC_MoveContinuousRelative | x | x |
| 13 | PLCopen | MC_PositionProfile | x | x |
| 14 | PLCopen | MC_VelocityProfile | X | x |
| 15 | PLCopen | MC_AccelerationProfile | x | x |
| 16 | PLCopen | MC_SetPosition | x | x |
| 17 | PLCopen | MC_SetOverride | x | x |
| 18 | PLCopen | MC_ReadParameter | x | x |
| 19 | PLCopen | MC_ReadBoolParameter | x | x |
| 20 | PLCopen | MC_WriteBoolParameter | x | x |
| 21 | PLCopen | MC_WriteParameter | x | x |
| 22 | PLCopen | MC_ReadActualPosition | x | x |
| 23 | PLCopen | MC_ReadActualVelocity | x | x |
| 24 | PLCopen | MC_ReadStatus | x | x |
| 25 | PLCopen | MC_ReadAxisError | x | x |
| 26 | PLCopen | MC_Reset | x | x |
| 27 | PLCopen | MC_CamTableSelect | x | x |
| 28 | PLCopen | MC_CamIn | x | x |
| 29 | PLCopen | MC_CamOut | x | x |
| 30 | PLCopen | MC_Gearln | x | x |
| 31 | PLCopen | MC_GearOut | x | x |
| 32 | PLCopen | MC_GearlnPos | x | x |
| 33 | PLCopen | MC_PhasingAbsolute | x | x |
| 34 | PLCopen | MC_PhasingRelative | x | x |
| 35 | PLCopen | MC_HaltPhasing | - | x |
| 36 | PLCopen | MC_LoadControl | - | x |
| 37 | PLCopen | MC_LimitLoad | - | x |
| 38 | PLCopen | MC_LimitMotion | - | x |
| 39 | PLCopen | MC_LoadSuperlmposed | - | x |
| 40 | PLCopen | MC_LoadProfile | - | x |
| 41 | PLCopen | MC_TorqueControl | - | x |
| 42 | ABB | MCA_CamInDirect | x | x |
| 43 | ABB | MCA_CamInfo | - | x |
| 44 | ABB | MCA_Cam_Extra | x | x |
| 45 | ABB | MCA_DriveBasedHome | x | x |


| SNo | Funktion block <br> type | Funktion block name | Motion Library <br> V2 (PS552-MC) | Motion Library <br> V3 <br> (PS5611- <br> Motion) |
| :--- | :--- | :--- | :---: | :---: |
| 46 | ABB | MCA_GearInDirect M | x | x |
| 47 | ABB | CA_Indexing | x | x |
| 48 | ABB | MCA_JogAxis | x | x |
| 49 | ABB | MCA_MoveByExternalRe- <br> ference | x | x |
| 50 | ABB | MCA_MoveVelocityContin- <br> uous | x | x |
| 51 | ABB | MCA_MoveRelativeOpti | x | x |
| 52 | ABB | MCA_Parameter | x | x |
| 53 | ABB | MCA_PhasingbyMaster | - | x |
| 54 | ABB | MCA_ReadParameterList | x | x |
| 55 | ABB | MCA_SetOperatingMode | x | x |
| 56 | ABB | MCA_SetPositionContin- <br> uous | x | x |
| 57 | ABB | MCA_WriteParameterList | x | x |
| 58 | ABB | MCA_CamGetInterpola- <br> tionPosition | - | x |
| 59 | ABB | MCA_Home | x | x |
| 60 | ABB | MCA_Power | x | x |
| 61 | ABB | ECAT_402Parameter- <br> Homing_APP | x | x |
| 62 | ABB | ECAT_HomingOnTouchP- <br> robe_APP | x | x |
| 63 | ABB | ECAT_CiA402_TouchP- <br> robe_App | x | x |

PLCopen Part 4-Coordinated Motion is only available for V2 PLC and not yet available for V3 PLC.

## Examples

Prerequisite: Install the related library package with Automation Builder, Installation Manager "Modify"

Use cases and how to handle the function blocks of a certain library is described in sample projects. After the respective library has been installed the corresponding sample projects are available in the default path AutomationBuilder Examples. If the default path is inaccessible, click "Help $\rightarrow$ Project examples" in the Automation Builder menu.

### 6.5.11 MQTT client library

### 6.5.11.1 Structures and enumerations

MQTT_ERROR_I
D (Enum)

| Parameter | Value | Description |
| :--- | :--- | :--- |
| MQTT_ERR_NO_ERROR | 0 | No error. |
| MQTT_ERR_CONN_ <br> SERVICE_UNAVAIL | $16 \# 3001$ | The Network Connection has been made <br> but the MQTT service is unavailable on the <br> specified port. |
| MQTT_ERR_COMMUNI- <br> CATION_TIMEOUT | $16 \# 3013$ | The timeout value for the communication <br> has been exceeded. |
| MQTT_ERR_REC_PACKE <br> T_TOO_LONG | $16 \# 3017$ | Received topic is too long. |
| MQTT_ERR_PING_NO_A <br> NSWER | $16 \# 301$ A | The MQTT broker did not answer the ping. <br> MQTT client has passed the KeepAlive or <br> MQTT broker is unreachable. |
| MQTT_ERR_CONN_CLIE <br> NT_ID_NOT_ALLOWED | $16 \# 301 F$ | The Client identifier is correct UTF-8 but <br> not allowed by the Server. |
| MQTT_ERR_CONN <br> REFUSED_PROTOCOL | $16 \# 3020$ | The Server does not support the level <br> of the MQTT protocol requested by the <br> Client. |
| MQTT_ERR_CONN_REFU <br> SED_CONNECTION | $16 \# 3025$ | Connection refused, maybe the IP address <br> is malformed. |
| MQTT_ERR_UNSPECI- <br> FIED_ERROR | $16 \# 302 \mathrm{~B}$ | Internal library returned an unspecified <br> error. |
| MQTT_ERR_NET- <br> WORK_ERROR | $16 \# 302 D$ | General network error. |
| MQTT_ERR_CONN_AUTH | $16 \# 3217$ |  |
| FAILED |  |  |


| Parameter | Value | Description |
| :---: | :---: | :---: |
| MQTT_ERR_INPUT_02_0 | 16\#4020 | Function block Input 02 error (error case 0 ), specific error depends on used function block: <br> - MqttConnectWithCertBuffer (FB): Parameter Conn of function block was not set. <br> - MqttConnectWithCertFile (FB): Parameter Conn of function block was not set. <br> - MqttGetReceivedPacket (FB): Parameter Conn of function block was not set. <br> - MqttPublish (FB): Parameter Conn of function block was not set. <br> - MqttSubscribe (FB): Parameter Conn of function block was not set. <br> - MqttUnsubscribe (FB): Parameter Conn of function block was not set. MqttPing (FB): Parameter Conn of function block was not set. |
| MQTT_ERR_INPUT_03_0 | 16\#4030 | Function block Input 03 error (error case 0 ), specific error depends on used function block: <br> - MqttGetReceivedPacket (FB): Pointer payload not initialized. <br> - MqttPublish (FB): Publish topic name must not contain wildcard characters (+ or \#). <br> - MqttSubscribe (FB): Topic is missing. <br> - MqttUnsubscribe (FB): Topic is missing. |
| MQTT_ERR_INPUT_03_1 | 16\#4031 | Function block Input 03 error (error case 1), specific error depends on used function block: <br> - MqttPublish (FB): Payload is not set in MQTT_MESSAGE. |
| MQTT_ERR_INPUT_04_0 | 16\#4040 | Function block Input 04 error (error case 0 ), specific error depends on used function block: <br> - MqttConnectWithCertBuffer (FB): Check if Port number has been set correctly ( 0 is not accepted). <br> - MqttConnectWithCertFile (FB): Check if Port number has been set correctly (0 is not accepted). |
| MQTT_ERR_INPUT_06_0 | 16\#4060 | Function block Input 06 error (error case 0 ), specific error depends on used function block: <br> - MqttConnectWithCertFile (FB): Server certificate file was not found. |
| MQTT_ERR_INPUT_07_0 | 16\#4070 | Function block Input 07 error (error case 0 ), specific error depends on used function block: <br> - MqttConnectWithCertFile (FB): Client certificate file was not found. |


| Parameter | Value | Description |
| :--- | :--- | :--- |
| MQTT_ERR_INPUT_08_0 | $16 \# 4080$ | Function block Input 08 error (error case <br> o), specific error depends on used function <br> block: <br> - <br> MqttConnectWithCertFile (FB): Client <br> key file was not found. |
| MQTT_ERR_INPUT_12_0 | $16 \# 4120$ | Function block Input 12 error (error case <br> 0), specific error depends on used function <br> block: <br> - MqttConnectWithCertBuffer (FB): <br> Couldn't initialize Last Will message <br> because the topic is not set. <br> MqttConnectWithCertFile (FB): <br> Couldn't initialize Last Will message <br> because the payload is not set. |
| MQTT_ERR_INPUT_12_1 | 16\#4121 | Function block Input 12 error (error case <br> 1), specific error depends on used function <br> block: <br> - MqttConnectWithCertBuffer (FB): <br> Couldn't initialize Last Will message <br> because the topic is not set. |
|  |  | MqttConnectWithCertFile (FB): <br> Couldn't initialize Last Will message <br> because the payload is not set. |
| MQTT_ERR_FATAL_ERR | 16\#5FFFF | Fatal error state machine. |
| OR |  |  |

MQTT_QOS (Enum)

| Parameter | Value | Description |
| :--- | :--- | :--- |
| QOS_0 | - | Fire and forget (At most once delivered). |
| QOS_1 | - | Simple acknowledgement (At least once delivered). |
| QOS_2 | - | Complex acknowledgement (Exactly once delivered). |

MQTT_MES- This structure is used for messages which can be published or used for LastWill on MqttConSAGE nect(FB).

| Variable name | Data type | Default <br> value | Description |
| :--- | :--- | :--- | :--- |
| sTopic | STRING(MQTT_M <br> AX_TOPIC_LEN $)$ | Empty <br> string | Topic where this message belongs to. |
| pbyPayload | POINTER TO <br> BYTE | 0 | Payload which should be sent. |
| dwLen | DWORD | 0 | Length of the payload. |
| eQos | MQTT_QOS | QOS_0 | Quality of Service level. |
| xRetainFlag | BOOL | FALSE | True $=$ message must be stored by the <br> server, False $=$ server must not store <br> this message. |

MQTT_CON- Internal data required by the library to operate. This structure allocates memory and it is used to NECTION identify the MQTT connection you want to work with

| Parameter | Data type | Range |
| :--- | :--- | :--- |
| abyConn | Array | MQTT_CLIENT_STRUCT_SIZE |
| abyTxBuf | Array | MQTT_TX_BUF_SIZE |
| abyRxBuf | Array | MQTT_RX_BUF_SIZE |
| abyMsgBuf | Array | MQTT_MSG_BUF_SIZE |

### 6.5.11.2 Global variables

MQTT_CONSTANTS

| Parameter | Datatype | Value | Description |
| :--- | :--- | :--- | :--- |
| MQTT_MAX_IP_ADDRESS_L <br> EN | Word | 15 | Maximum length of the IP <br> address. |
| MQTT_MAX_PEM_KEY_LEN | Word | 2048 | Maximum length of the PEM <br> key. |
| MQTT_MAX_PEM_CERT_LE <br> N | Word | 3072 | Maximum length of the PEM <br> certificate. |
| MQTT_MAX_FILE_PATH_LEN | Word | 255 | Maximum length of the file path <br> to the certificate files. |
| MQTT_MAX_CLIENT_ID_LEN | Word | 250 | Maximum length of the client <br> id. |
| MQTT_MAX_USER- <br> NAME_LEN | Word | 250 | Maximum length of the user- <br> name. |
| MQTT_MAX_PASS- <br> WORD_LEN | Word | 250 | Maximum length of the pass- <br> word. |
| MQTT_MAX_TOPIC_LEN | Word | 255 | Maximum length of the topic. <br> MQTT_CLIENT_STRUCT_SIZ <br> E <br> Word <br> MQTT_TX_BUF_SIZE |
| Word | 1024 | Size of the internal connection <br> structure representing the con- <br> nection state. |  |
| MQTT_RX_BUF_SIZE | Word | 1024 | Size of the internally used <br> output buffer. |
| Size of the internally used input |  |  |  |
| buffer. |  |  |  |

### 6.5.12 PLCopen libraries

### 6.5.12.1 Common function block state machine

6.5.12.1.1 General

Most of the V3 function blocks follow the behavior model and style as recommended by PLC Open.

- Clear separation between "Edge triggered FBs" ("Execute") or "Level triggered FBs" ("Enable")
- Binary status outputs: "Done", "Busy", "Error" (exclusive)
- Standardized state machine
- CamelCase naming for function block and all inputs and outputs


Example: Edge_Triggerd_Function_Block_EthOwnIP according to PLCopen
Currently the following "function block state machines" are used:

- "Edge Triggered" (Input "Execute"), for example EthOwnIP
- "Level Controlled" (Input "Enable")
- "Level Controlled Continuous" (Input "Enable", no "Done" output, for example PID)


In contrast to AC500 V2 POUs, either "Done" or "Error" is set, not both outputs at the same time in case of an error.

The state machines are explained in the following chapters.

### 6.5.12.1.2 Edge triggered (AbbETrig)



After a rising edge at the input "Execute" the state goes from "Dormant" to "Busy". In the first cycle all inputs are sampled and stored.

When the task is completed successfully the state goes from "Busy" to "Done".
In case of an error the state goes to "Error".
The states "Done" or "Error" are stable for minimum one cycle and as long as "Execute" is "TRUE". With a falling edge of "Execute", the state goes via Reset to "Dormant".

## Description of standard inputs and outputs:

- "Execute"

A rising edge starts the operation, the output "Busy" goes to "TRUE". In the first cycle all other inputs are read and stored, afterwards they are ignored.
A falling edge does not stop the operation.
After "Done $=$ TRUE" or "Error $=$ TRUE" and "Execute $=$ FALSE" all outputs will be reset.

- "Busy"

Operation is running (while outputs "Done" and "Error" are "FALSE")

- "Done"

Operation is completed without error (while outputs "Busy" and "Error" are "FALSE"). This output is "TRUE" for at least one cycle or until "Execute" is set to "FALSE"

- "Error"

Operation is stopped with error (while outputs "Busy" and "Done" are "FALSE").
This output is "TRUE" for at least one cycle or until "Execute" is set to "FALSE".
The output "ErrorID" gives more details about the error.

### 6.5.12.1.3 Level controlled (AbbLCon)



After a rising edge at the input "Enable" the state goes "Dormant" to "Busy". All inputs are sampled and considered continuously.

When the task is completed successfully the state goes from "Busy" to "Done".
In case of an error the state goes to "Error".
The states "Done" or "Error" are stable for minimum one cycle and as long as "Enable is TRUE". With a falling edge of "Enable", the state goes via Reset to "Dormant".
The Busy state can be aborted from outside by setting the "Enable" input to "FALSE".
After Aborting is done the state goes back to "Dormant".
Description of standard inputs and outputs:

- "Enable"

A rising edge ("Enable = TRUE") starts the operation, the output "Busy" goes to "TRUE". All other inputs are read and considered continuously. A falling edge ("Enable = FALSE") aborts the operation.
During Aborting the Busy is still "TRUE". Afterward all outputs are reset.

- "Busy"

Operation is running (while outputs "Done" and "Error" are "FALSE")

- "Done"

Operation is completed without error (while outputs "Busy" and "Error" are "FALSE"). This output is "TRUE" for at least one cycle or until "Enable" is set to "FALSE"

- "Error"

Operation is stopped with error (while outputs "Busy" and "Done" are "FALSE"). This output is "TRUE" for at least one cycle or until "Enable" is set to "FALSE". The output "ErrorID" gives more details about the error.

### 6.5.12.1.4 Level controlled continous (AbbLConC)

This state machine is a special case of "Level Controlled". Only difference is that this function block type is never done, for example a PID which never stops.

Therefore these function blocks have no "Done" output.
Description of standard inputs and outputs:

- "Enable"

A rising edge ("Enable = TRUE") starts the operation, the output "Busy" goes to "TRUE". All other inputs are read and considered continuously.
A falling edge ("Enable = FALSE") aborts the operation.
During Aborting the "Busy" is still "TRUE". Afterward all outputs are reset.

- "Busy"

Operation is running (while output "Error is FALSE")

- "Error"

Operation is stopped with error (while output "Busy is FALSE").
This output is "TRUE" for at least one cycle or until "Enable" is set to "FALSE".
The output "ErrorID" gives more details about the error.

### 6.5.12.1.5 Error_ID

Each library contains an enumeration "ERROR_ID", which is valid for this library but not across all libraries


## Inputs/Outputs Documentation

## ERROR_ID (ENUM)

TYPE ERROR_ID
InOut:

| Name | Initial | Comment |
| :--- | :--- | :--- |
| NO_ERROR | 0 | No error |
| ERR_FAlLED | 1 | General error |
| ERR_PARAMETER | 2 | Parameter error |
| ERR_JOB_PARAMETER | 3 | Job could not be created |
| ERR_TIMEOUT | 5 | Timeout error |
| INVALID_INPUT_ETH | $16 \mathbb{1 0 4 0 2 0}$ | Wrong value on input \#2 (ETH) |
| INVALID_INPUT_IPADR | $16 \# 4030$ | Wrong value on input \#3 (IP address) |
| FATAL_ERROR | 16\#5FFF | Fatal error state machine |

Only the following errors are unique:

- 16\#5FFF FATAL_ERROR from state machine
- $16 \# 4000$ errors are used for input errors, same scheme like in V2:


## 4000hex...4FFFhex - Block Input Error

The error $4 x x x h e x$ is used in case of detected Function Block input parameter errors. The error is structured as follows:

```
4 X X X HEX
```



```
\(0=\) Invalid value
1 = Value too low
\(2=\) Value too high
\(3=\) Wrong combination of the parameters
1...FF = Number of the input
```


### 6.5.12.1.6 Compatibility with V2 function blocks

In order to ensure compatibility with V2 applications a lot of ABB function blocks are delivered together with a compatible version in classic style:
Names in CAPITAL letters, input "EN" and outputs "DONE", "ERR" and "ERNO":


The classic blocks internally use the PLCopen style function blocks. The inputs and outputs are mapped in the following way:


### 6.5.13 Pumping library V3

6.5.13.1 Overview of the PS5608 pumping library

The PS5608 - pumping library V3 stands as an upgrade of PS571 - pumping library V2. Specifically designed for pumping stations equipped with up to eight pumps, this library encompasses a comprehensive set of function blocks. This library helps in configuring and monitoring the various pumps in pumping stations.
Depending on the pumping station requirement and objectives, the available process modes typically include pressure control, flow control, and level control.

- Pressure control: This mode focuses on maintaining a specific pressure level within the system. It is ideal for applications where pressure regulation is paramount, such as in water supply systems or hydraulic systems.
- Flow control: Flow control mode is employed to regulate the rate at which the fluid is pumped through the system. It is essential in applications where maintaining a consistent flow rate is critical, such as in chemical manufacturing or irrigation systems.
- Level control: Level control mode is employed to manage the height or volume of liquid in a reservoir or tank. It is commonly used in wastewater treatment plants, industrial storage tanks, and sewage systems.
Depending on the specific operational needs and desired level of control and efficiency for the pumping stations, it offers various operational modes, each tailored to meet different requirements:
- Multimode - all VFD pumps: In this mode, all pumps are equipped with Variable Frequency Drives (VFDs). VFDs allow for precise control of pump speed, making it ideal for applications where fine-tuning flow rates and pressure is essential. This mode provides high efficiency and flexibility.
- Traditional mode - One pump VFD, other DOL pumps: This configuration combines the advantages of a VFD pump with the simplicity of direct-on-line (DOL) pumps. Typically, one pump is equipped with a VFD for precise control, while the others operate using a traditional DOL starter. This mode balances control and cost-effectiveness.
- DOL Mode - All DOL Pumps: In this mode, all pumps operate using direct-on-line (DOL) starters, which provide basic on/off control without speed variation. It's a cost-effective choice for applications where precise control is less critical.

Basic block diagram of a pumping station is as shown below.


The key features supported by the library includes:

- Configuration functions for pump station that permits:
- Users to customize pump parameters like pump station name, number of pumps, process mode of pumps, pump combination and pump speed.
- Communication with all pumps in the field that are run by DOL/VFD motors by configuring the pump settings and collecting the pump operation status.
- Pumping functions of the pump that supports:
- Precise management of emptying and filling operations by comparing actual fluid level with the set fluid level for level control.
- Uniform fluid level distribution across multiple tanks and reservoirs by providing ON/OFF command to the pumps for level control.
- Maintenance of flow in the network for flow control.
- Stabilization of pressure in the network for pressure control.
- Closed loop control of pressure and flow using PID control.
- Pump turn off/on sequence generation based on demand.
- Auxiliary functions of the pump that include:
- Cleaning up of the pump using antijam functionality.
- Capability to calculate and monitor energy consumed and saved.
- Calculation of pump flow rate using pump head flow and pressure flow rate curve characteristics.
- Computation of maintenance schedule for the pump.
- Offering analog and binary protection mechanisms to safeguard pumps against operational risks.
- Conserving energy by enabling pumps to enter sleep mode during periods of low demand.
- Soft fill functionality that ensures gradual pump start up to prevent water hammer and system stress.
- Simulation functions of the library that help in system test and validation consists of:
- DOL (Direct-on-line pumps) simulation
- VFD (Variable Frequency Drive) simulation
- Pumptank simulation for tank behavior.

This PLC library for pumping stations is a comprehensive solution that together optimize pump control, energy efficiency, maintenance, protection, and system simulation. It stands as an invaluable tool in the field of industrial automation, ensuring reliability, efficiency, and cost-effectiveness across a wide range of pumping applications.

### 6.5.13.2 Comparison of V2 and V3 pumping library

The below table compares the function blocks in the V2 library package and corresponding adapted function blocks in the V3 library package.

Table 768: Comparison of V2 and V3 pumping library

| PS571-Pumping library package (V2) |  | PS5608-Pump library package (V3) |  |
| :---: | :---: | :---: | :---: |
| Library name | Function block | Library | Function block |
| PMP_AC500_V25.lib | PMP_CONFIGURA- TION | ABB_Pump_AC500.c ompiled-library <br> ABB_Pum- <br> pEco_AC500.com-piled-library | PmpConfiguration |
|  | PMP INTERFACE_DOL |  | PmpInterfaceDol |
|  | PMP_INTERFACE_VFD |  | PmpInterfaceVfd |
|  | PMP_LEVEL_COMPARĀTOR |  | PmpLevelComparator |
|  | PMP_LEVEL_DISTRIBUTOR |  | PmpLevelDistributor |
|  | PMP_FLOW_DISTRIBUTOR |  | PmpFlowDistributor |
|  | PMP PRES-SURĒ_DISTRIBUTOR |  | PmpPressureDistributor |
|  | PMP_PID |  | PmpPid |
|  | $\begin{aligned} & \text { PMP_SEQUENCE_G } \\ & \text { EN } \end{aligned}$ |  | PmpSequenceGen |
|  | PMP_ANTIJAM |  | PmpAntiJam |
|  | $\begin{aligned} & \text { PMP_ENERGY_CAL } \\ & \text { C } \end{aligned}$ |  | PmpEnergyCalc |
|  | $\begin{aligned} & \hline \text { PMP_FLOW_CALC_ } \\ & \mathrm{HQ} \end{aligned}$ |  | PmpFlowCalcHQ |


| PS571-Pumping library package (V2) | PS5608-Pump library package (V3) |
| :---: | :---: |
| PMP PQ | PmpFlowCalcPQ |
| PMP MAINTAINENC̄E | PmpMaintainence |
| PMP PROTECTION_ANALOG | PmpProtectionAnalog |
| PMP PROTECTION_BINARY | PmpProtectionBinary |
| PMP_SLEEP | PmpSleep |
| PMP_SOFT_FILLING | PmpSoftFilling |
| PMP_DOL_SIMU | PmpDolSimu |
| PMP_DRIVE_SIMU | PmpDriveSimu |
| PMP_TANK_SIMU | PmpTankSimu |

Table 769: Comparison of V3 Pump- and PumpEco libraries

| Name of the library | ABB_Pump_AC500 library | ABB_PumpEco_AC500 library |
| :--- | :--- | :--- |
| Runtime license | Needs pump runtime license | Needs pump eCo runtime license |
| PLCs supported | Runs in all PLC's | Runs only in eCo PLC's |
| Number of pumps sup- <br> ported | 8 | 4 |

### 6.5.13.3 Compontents of the library

### 6.5.13.3.1 Overview

The library contains function blocks needed for functions of pumping application (pressure control, flow control or level control). It also contains function blocks for some advanced functions needed for different applications

```
--\square POUs
```

--\square POUs
- PUMPING
- PUMPING
--
--
Auxdiary Functions
Auxdiary Functions
* 圁 PmpAntiJam (FB)
* 圁 PmpAntiJam (FB)
* 宣 PmpEnergyCalc (FB)
* 宣 PmpEnergyCalc (FB)
+ 首 PmpFlowCalcHQ (FB)
+ 首 PmpFlowCalcHQ (FB)
+ 荁 PmpFlowCalcPQ (FB)
+ 荁 PmpFlowCalcPQ (FB)
\# - PmpMaintenance (FB)
\# - PmpMaintenance (FB)
+ - - PmpProtectionAnalog (FB)
+ - - PmpProtectionAnalog (FB)
+ 目 PmpProtectionBinary (FB)
+ 目 PmpProtectionBinary (FB)
+ 兰] PmpSleep (FB)
+ 兰] PmpSleep (FB)
* 首 PmpSoftFilling (FB)
* 首 PmpSoftFilling (FB)
-- Configuration Functions
-- Configuration Functions
+ 目 PmpConfiguration (FB)
+ 目 PmpConfiguration (FB)
* 鱼 PmpInterfaceDol (FB)
* 鱼 PmpInterfaceDol (FB)
* 言 PmpInterfaceVFD (FB)
* 言 PmpInterfaceVFD (FB)
- Pumping Functions
- Pumping Functions
-M Emptying and Filling
-M Emptying and Filling
目 PmplevelComparator (FB)
目 PmplevelComparator (FB)
\# 目 PmplevelDistributor (FB)
\# 目 PmplevelDistributor (FB)
-\square Process Controls
-\square Process Controls
-D Flow Control
-D Flow Control
\# PmpFlowDistributor (FB)
\# PmpFlowDistributor (FB)
Pressure Control
Pressure Control
* 目 PmpPressureDistributor (FB)
* 目 PmpPressureDistributor (FB)
* 目 PmpPid (FB)
* 目 PmpPid (FB)
* - PmpSequenceGen (FB)
* - PmpSequenceGen (FB)
-
-
Simulation Functions
Simulation Functions
+ 目 PmpDolSimu (FB)
+ 目 PmpDolSimu (FB)
+ 亩 PmpDriveSimu (FB)
+ 亩 PmpDriveSimu (FB)
* 亶 PmpTankSimu (FB)

```
    * 亶 PmpTankSimu (FB)
```

Fig．355：Overview of pumping library

## 6．5．13．3．2 Installation

The library is part of the Automation Builder 2．6．1 or higher．Use theLibrary Manager to add the library into project．
For more details on the package，refer to the release notes of the latest Automation Builder．

## 6．5．13．3．3 Compatibility

| Requirement | Specification |
| :--- | :--- |
| Hardware | AC500 V3 midrange and eCo PLCs |
| Software | Automation Builder 2．6．1 or higher |
| Library | ABB＿Pump＿AC500．compiled library <br> ABB＿PumpEco＿AC500．compiled library |
| License | Runtime license required |

### 6.5.13.3.4 Required sensors

A minimal sensor equipment is required for the following measurements:

- Actual pressure measurement for pressure control.
- Actual flow measurement for flow control.
- Actual level measurement for level control.

The pressure sensor can be at the input and/or at the output. The flow sensor should be only at the output. Often, at the input a sensor, typically pressure, is placed for advanced protection and diagnosis.

### 6.5.13.4 Control philosophy of pumping library

The Pumping library is used for controlling the following three water pumping processes:

- Pressure control
- Flow control
- Level control - Emptying or filling

The figure below gives an overview of the pumping library and explains how the library processes the data from the user and then controls the processes.


To run any process, the pumping library follows the four stages described below:

### 6.5.13.4.1 Stage 1: Comparator

The set process value (e.g., pressure in psi, flow rate in $\mathrm{m}^{3} / \mathrm{h}$ ) is compared with the measured actual value. The distributor uses the output of the comparator to decide the number of pumps and their operating speeds to meet the demand.

- Pressure control and Flow control:
-Pressure control and Flow control: In pressure and flow controls, the function block "PmpPid acts" as a comparator which gives the "PidOut" in [\%] to the function block "PmpPressureDistributor" or "PmpFlowDistributor". In the distributor the PID output is then converted as reference speed set point to the pumps.
- Emptying or Filling mode:
- Emptying or Filling mode: In emptying or filling modes, the function block
"PmpLevelComparator" is used to compare the actual level of the tank with the set levels to start or stop the pumps. This information is sent to the distributor by the output "NumberOfPumpDemand". For example, in filling mode, at lower actual level more pumps are needed to run, to fill the tank.


### 6.5.13.4.2 Stage 2: Sequence generator

The "PmpSequenceGen" function block is used. The following functions are included:

- Decides which pump is ready to run in the auto mode.
- Decides which pump to start in the sequence, based on the least actual runtime hours. Exception - when the pump station is traditional type (with master pump on VFD, rest on DOL), then the pump ID = 1 which is attached to the VFD will always run as master and is first to start and last to stop.
- Decides which pump to stop in the sequence, based on the highest actual runtime hours. Exception - when the pump station is traditional type (with master pump on VFD, rest on DOL), then the pump ID = 1 which is attached to the VFD will always run as master and is first to start and last to stop.
- Assigns a master status to the pump in the sequence.
- Indicates how many pumps are ready for automation and how many are running.

The figure below explains a use case where the demand increases and then decreases. The sequence is generated based on these criteria by the function block "PmpSequenceGen".

| DEMAND | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Master | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Next Pump to Start | 2 | 3 | 4 | 5 | 6 | 7 | 8 |  | 2 | Whichever pump has minimum runtime will start next |  |  |  |  |  |
| Next Pump to Stop | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 1 |
| Pumps Running | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  |
|  |  | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |  |
|  |  |  | 3 | 3 | 3 | 3 | 3 | 3 | 4 | 5 | 6 | 7 | 8 |  |  |
|  |  |  |  | 4 | 4 | 4 | 4 | 4 | 5 | 6 | 7 | 8 |  |  |  |
|  |  |  |  |  | 5 | 5 | 5 | 5 | 6 | 7 | 8 |  |  |  |  |
|  |  |  |  |  |  | 6 | 6 | 6 | 7 | 8 |  |  |  |  |  |
|  |  |  |  |  |  |  | 7 | 7 | 8 |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | 8 |  |  |  |  |  |  |  |

### 6.5.13.4.3 Stage 3: Distributor

The distributor function block distributes the start/ stop command and speed reference to all VFD operated pumps.

- Pressure control: For pressure control, the function block "PmpPressureDistributor" is used. The PID output coming from the function block "PmpPid" (comparator) is converted as reference speed setpoint to the master pump. The function block works based on the reference speed of the master pump. If the reference speed is higher, a greater number of pumps are needed to meet the demand. If the reference speed is lower, the already running pumps in the network need to be stopped. The function block:
- Calculates the speed reference of the master pump and then starts/ stops the follower pumps based on the settings in the inputs "StartSpeedFwr" or "StopSpeedFwr".
- Distributes the speed references to VFD follower pumps based on the settings in the input "FollowerMode".
- Flow control: For flow control, the function block "PmpFlowDistributor" is used.
- In the flow distributor the number of pumps to run is decided by the following ratio: "FlowSetpoint"/ (Number of pumps in station * Nominal flow of pump).
- The function block distributes the speed references to VFD follower pumps based on the settings in the input "FollowerMode".
- Emptying/Filling mode: For emptying or filling operation, the function block PmpLevelDistributor is used. The function block starts/ stops the pumps based on the input "NumberOfPumpDemand" (demand) which it receives from function block "PmpLevelComparator".
- If the actual level is in the normal range, then the speed reference of the pumps are based on the input "NormalSpeed".
- If for example while filling the actual level reaches below the "LowLevel", then all pumps run at "HighSpeed", to quickly fill the pump. Similar analogy is followed when the pump is in emptying mode.


### 6.5.13.4.4 Stage 4: DRIVE or DOL system

The output of the distributors ("AutoStartCmd" and "AutoSpeedRef") is connected to function block "PmpInterfaceVFD" and "PmpInterfaceDol". The function blocks establish the link between library and field devices i.e., pumps.

- For pumps driven by VFD motors, the function block "PmpInterfaceVFD" can be connected with the drives communication function blocks to establish communication and control with the VFD.
- For pumps driven by DOL motor, the function block "PmpInterfaceDol" can be used for start/stop controls.
For more information, see the detailed control philosophy in the following sections.


### 6.5.13.5 Features in pumping library

Pumping library is mainly designed to use in various application of water industries for example, wastewater treatment, stormwater overflow tank, water association and well stations etc. Maximum 8 pumps can be configured for pressure, flow and level control.

### 6.5.13.5.1 Types of pumping stations

This section explains the possible combination of library features with different types of pumping stations:

1. "Multi pumping" stations - maximum 8 pumps each fed by variable frequency drive.
2. "Traditional pumping" station - maximum 8 pumps, where one pump (mandatory) fed by variable frequency drive and rest by the direct online motor.
3. "DOL pumping" station - maximum 8 pumps running by direct online motors.

## Multi pumping station

This multi pumping station configuration has the best overall and operation economics:

- The library is working in the PLC which controls the operation of the variable frequency drives based on the process requirements.
- All the pumps are connected to the motor which is fed by the VFD. This type of pumping station is most flexible and has optimal process control efficiency performance.
- Based on the load requirements the number of pumps operating can vary and the pumps in operation can be kept as much as possible in their "best efficiency point".



## Traditional pumping station

This pumping station configuration has following features:

- One lead pump is connected to the VFD, and others are connected to the DOL motors.
- The library is working in the PLC which controls the operation of the variable frequency drive of the lead pump. The operation of the DOL motors is based on the process requirements.
- Based on the load requirements the number of pumps operating can vary but the lead pump with the VFD always operates.



## Direct-on-line (DOL) pumping station

This pumping station configuration has all the pumps running by the direct-on-line (DOL) motors. This is the most investment cost-effective configuration. It has following features:

- The pump library runs on the AC500 PLC.
- The pump library controls the ON/OFF the pumps based on the process demands.



### 6.5.13.5.2 Process control vs. pump combinations

The table below explains the applicability of the three process controls and their pump combinations.

| Process control | Pump combinations |  |  |
| :--- | :--- | :--- | :--- |
|  | DOL | Multi pump | Traditional |
| Flow | $\boldsymbol{\checkmark}$ | $\boldsymbol{\checkmark}$ | $\boldsymbol{\checkmark}$ |
| Level | $\boldsymbol{\checkmark}$ | $\boldsymbol{\checkmark}$ | $\boldsymbol{\checkmark}$ |
| Pressure | $\boldsymbol{x}$ | $\boldsymbol{\checkmark}$ | $\boldsymbol{\checkmark}$ |

### 6.5.13.5.3 Process control types

There are three main application and control function blocks:

- Pressure control function blocks
- Flow control function blocks
- Level control function blocks


## Pressure control function blocks

Pressure control process flow diagram.

Pressure control is used in applications with individual or multiple water consumers like in water distribution networks. A typical pressure control application diagram is shown below.


1 Suction tank
2 Parallel operating pumps P1 ... P8
3 Pressure setpoint, discharge pressure
4 Consumer
5 Water distribution network
6 Water tower buffer tank
H Water head between the levels in the suction tank and water tower buffer tank

Using pressure
Pressure control in the pumping library can help in the following operations:

- Water consumption fluctuates always and may be discontinuous over a time period, because of which it is difficult to have a continuous flow rate like in the flow control process.
- The number of pumps and the pump speed is decided by the water consumption in the water distribution system.
- Water is supplied to the pump station through suction pipeline from a suction tank.
- The pumps may be operating in parallel.
- The discharge pressure is measured in the discharge pipeline. The measured discharge pressure is controlled according to the pressure set point.
- Each pump must be started with a minimum speed to build up the pressure which is required to produce a minimum flow.
- The pressure to deliver a minimum flow depends of the water head $(\mathrm{H})$ between the water levels in the suction tank and target tank.
- If there is pump station shutdown due to protection shutdown input, the pumps must be stopped sequentially to prevent water hammering.

Variable speed drives should be stopped through a speed ramp. Otherwise, the water hammering will quickly wear out the pipeline connections and the pump station equipment.

Pump combina- Pressure control mode works for two types of pump combinations: Multi pump and traditional tions for pressure control pump.
The combination of function blocks to execute a pressure control is shown in the figures below.

- Multi pump - In this mode all pumps are run using VFDs.


Traditional pump - In this mode only one pump will run using the VFD and the rest of the pumps will run using the direct-on-line motors. The pump, which is fed by VFD will always be master or lead pump.

Based on the load requirement the number of pumps operating can change but the pump fed by VFD will always be operating.


The pressure control mode cannot be supported by a DOL pump.
The combination with any VFD pump is necessary.

## Control philosophy of pressure control mode

The pressure control mode follows this sequence of operation:

1. When the process is started, the first pump runs at "MinimumSpeed" for the time defined in the "TimeToRunMinSpeed" duration. With this process the pipe starts filling gradually and then the normal operation of PID control takes over.
2. The PID compares the required pressured and the actual pressure to generate the output in terms of percentage. This output must be connected to the function block "PmpPressureDistributor".
3. The function block "PmpPressureDistributor" receives the "PidOut". The distributor converts the PID output in terms of speed for the master pump.
4. The function block "PmpSequenceGen" has the "MasterPump" information.
5. The "PidOut" is scaled in terms of speed in this method:

Speed reference of "MasterPump" = ("NominalSpeed" of "MasterPump") * ("PidOut"/100)
6. As the PID output increases, the speed of the master pump increases. A higher PID output indicates a high demand. In case one pump is not able to cater the requirement, then more pump (followers) are needed to start and supply water to maintain the pressure.
7. If speed reference of the "MasterPump" increases such that it is more than the "StartSpeedFwr[1]", then the first follower in the network will start. The information about which pump to start comes from function block "PmpSequenceGen" at the output "NextPumpToStart".
8. If the speed reference of the "MasterPump" increases further such that it is more than "StartSpeedFwr[2]", then second follower pump starts. This sequence is followed till there is an increase in demand. See the timing diagram below.


Speed set point of Master Pump = PidOut \% * Nominal speed of master pump
Start Delay: Delay time [s] to start the next pump
9. Similar when the demand decreases, the speed reference of master decreases. See the timing diagram below. As the speed reference of the master pump goes below "StopSpeedFwr[1]", the first follower stops. The ID of next pump to stop comes from "PmpSequenceGen" as "NextPumpToStop". If the speed reference of the "MasterPump" decreases further such that it is less than "StopSpeedFwr[2]", then second follower pump stops. This sequence is followed till there is a decrease in demand.

The follower pumps in the above figure is considered to be in the "FollowerMode"= 1, copy master speed. They can also run at their individual speed if "FollowerMode"= 2, fixed speed.


Stop Delay: Delay time [s] to stop the next pump

### 6.5.13.5.4 Flow control process flow diagram

Flow control is used in applications where continuous flow distribution to one or more target tanks or reservoirs is needed.

A typical flow control application diagram is shown below.


1 From supply station
2 Suction tank
3 Parallel operating pumps P1 ... P8
4 Flow setpoint, discharge pressure
5 Water distribution network
6 Target tank/Reservoir
7 To next station or distribution network
H Water head between the levels in the suction tank and target tank

### 6.5.13.5.5 Using flow control

Flow control in the pumping library can help in followings pumping operations:

- Supplying water to the pump station through the suction pipeline from a suction tank.
- Operating pumps in parallel.
- Controlling continuous water flow over a time period.
- Measuring the discharge flow in the discharge pipeline.
- Defining the number of pumps and pump speed based on water flow demand (flow set point).
- Starting each pump with minimum speed to build up the pressure required to produce a minimum flow.
The pressure depends on the water head ( H ) between the water levels in the suction tank and the target tank.
- In case of station shutdown stops the pumps sequentially to prevent water hammering.
- Stops variable speed drives through a speed ramp.


### 6.5.13.5.6 Pump combinations for flow control

Flow control works for all the three types of pump combinations: Multi pump, traditional pump, and DOL pump.

- Multi pump - In this mode all pumps are run using VFDs. See the example diagram below.

- Traditional pump - In this mode only one pump is run using the VFD and rest are run using the Direct On Line (DOL) motors. See the example diagram below.

- DOL Pump - In this mode all the pumps are operated by DOL motors. The function block PMP_PID is not configured as there is no possibility to change the speed of the pump. See the example diagram below.



### 6.5.13.5.7 Control philosophy of flow control mode

The above figures show the combination of function blocks to execute the flow control. As it can be seen,

- When the process is started, the first pump runs at the "MinimumSpeed" for the time defined in the "TimeToRunMinSpeed" duration. This is to gradually start filling the pipe. After this the normal operation of PID control takes over.
- The PID acts as a comparator: the PID compares the required flow and the actual flow to generate output in terms of percentage. This needs to be connected to the "PmpFlowDistributor".
- "PmpFlowDistributor" takes the "PidOut". Then converts this output in terms of speed for the master pump. The information of master pump comes from the "MasterPump" of the "PmpSequenceGen". The scaling of the "PidOut" in terms of speed is done in following ways. Speed reference "MasterPump" = ("NominalSpeed" of "MasterPump") * ("PidOut"/ 100)
- This speed reference is given to the master and the follower (if Follower Mode $=1$ ). If Follower Mode $=2$, then follower pumps will run at their fixed speed given in the input "FollowerSpeed" [1..8].
- The distributor then decides about the number of pumps to operate based on the following formula, Number of pumps = "FlowSetpoint""NominalFlow", here it is assumed that all pumps are of same rating.


It is assumed that all the Pumps are of the same capacity.

- As the flow set point increases, the demand of pumps increases.
- Example: if each pump is of capacity = 100 cubic meters per hour and the flow set point is 400 cubic meters per hour, then four pumps will start together. This is unlike the pressure control where the pumps do not start together but based on speed reference of the master pump.

- Similarly when the demand decreases, the speed reference of the master decreases. As the speed reference of the master pump goes below than "StopSpeedFwr[1]", the first follower stops. The ID of next pump to stop comes from "PmpSequenceGen" as "NextPumpToStop".
- If the speed reference of the "MasterPump" decreases further such that it is less than "StopSpeedFwr[2]", then second follower pump stops. This sequence is follows till there is a decrease in demand.


The follower pumps in the above figures are considered to be in the "FollowerMode" = 1, copy master speed.
They can also run at their individual speed if "FollowerMode" $=2$, fixed speed.

### 6.5.13.5.9 Level control function blocks

## Level control process flow diagram

Level control is used in applications where continuous flow distribution into one target tank or reservoir is needed.
A typical flow control application diagram is shown below.


1 From supply station
2 Suction tank
3 Parallel operating pumps P1 ... P8
4 Water distribution network
5 Target tank/Reservoir
6 To next station or distribution network
H Water head between the levels in the suction tank and target tank

## Using level control

Level control in the pumping library can help in followings pumping operations:

- Emptying the target tank/reservoir by the next pump station or water distribution network.
- Filling the suction tank from a supply station.
- Defining the number of running pumps based on water level in the water tank.
- Start/stop the pumps with predefined start/stop levels.
- In case of a station shutdown, stop pumps sequentially to prevent water hammering.


## Pump combinations for level control

Level control works for all the three types of pump combinations: Multi pump, traditional pump, and DOL pump.

- Multi pump - In this mode all pumps are run using VFDs. See the example diagram below.

- Traditional pump - In this mode only one pump is run using the VFD and rest using the Direct On Line (DOL) motors. See the example diagram below.

- DOL Pump - In this mode all the pumps are operated by DOL motors. In this case the PMP_PID is not configured as there is no possibility to change the speed of the pump. See the example diagram below.



## Control philosophy of level control mode

The level control is based on two modes.

- Emptying mode
- Filling mode

The level control mode follows this sequence of operation:

- The "PmpLevelComparator" compares the actual level ("ActLevel") with the set start levels ("StartLevel" [1...8]). Based on which the comparator decides the demand - "NoOfPumpDemand". This demand goes as an input information to the "PmpLevelDistributor".
- The "PmpLeveIDistributor" then starts the required pumps. In case the pumps are driven by VFD, the distributor also gives the speed reference to each pump. The information of which pump to start in the sequence is given to the distributor by "PmpSequenceGen".

Time diagram:
Emptying mode
Emptying Mode


Time diagram:
Time diagram
Filling mode
Filling Mode

6.5.13.6 Auxiliary functions

The various auxiliary functions supported in this library are described in the following chapters

### 6.5.13.6.1 Antijam overview

The pump antijam function is used to prevent building up of solids on the pump impellers or pip-ing. The function consists of a programmable sequence of forward and reverse runs of the pump to shake off any residue on the impeller or piping. This function is mostly used with booster and wastewater pumps. See timing diagram below:


The function block PmpAntijam performs the antijam function by running the pump at high speeds without any ramp up/down time.

### 6.5.13.6.2 Pump energy calculation functionality

The function block "PmpEnergyCalc" calculates the energy consumption of the pumping station run by variable frequency drive and gives the information on cumulative flow in cubic meters. The function includes following calculations:

- Total energy consumed.
- Total energy saved.
- Total cost saved.
- Reduced CO2 emission.


### 6.5.13.6.3 Pump flow calculation functionality using $H Q$ and $P Q$ curves

The function blocks "PmpFlowCalcHQ" and "PmpFlowCalcPQ" calculates the flow rate of pumps using the pump head flow (HQ) and pump pressure flow rate ( PQ ) curve characteristics respectively. These functions are applicable only for pumps run by variable frequency drives.
The flow calculation function provides a reasonably accurate calculation of the flow without the installation of a separate flow meter. The flow is calculated based on the pump characteristics. The user can define the PQ (Power vs Flow) or HQ (Head vs Flow) values for the performance curve at the inputs of the function block for the five coordinate points. The values of the PQ and $H Q$ are generally provided by the pump manufacturer and all points defined must be within the operating range of the pump. These points are taken at the maximum operation speed and scaled down by the block according to the actual speed. By this method a good accuracy is achieved with limited parameterization effort.

### 6.5.13.6.4 Pump maintenance functionality

The function blocks "PmpMaintenance" is used to record the maintenance schedule of the pump. The function block maintains an hourly counter to track the pump maintenance interval and generates timely reminders and alarms.

### 6.5.13.6.5 Pump protection functionality using analog/binary inputs

The function blocks "PmpProtectionAnalog" and "PmpProtectionBinary" monitors analog inputs and binary inputs respectively and generates signals to shut down or to block the starting of pump or the process.
The function block sends the signals to shut down or to block the starting of pump through inputs "ProtectionShutdown" or "StartBlocker" of these distributor and pump interface function blocks: "PmpLeveIDistributor", "PmpFlowDistributor" "PmpPressureDistributor", "PmpInterfaceVfd" and "PmpInterfaceDol".

### 6.5.13.6.6 Pump sleep functionality

The function block "PmpSleep" stops all pumps in the pressure control process during low demand. The function works only for pressure control (when function block "PmpConfiguration" has the input ProcessMode $=1$ (Pressure control) with multi pump station or traditional pump station.


### 6.5.13.6.7 Pump soft fill functionality

The function block "PmpSoftFilling" enables slow filling operation in pipes in the beginning of the process to avoid damages. If this operation is selected, it is executed before the actual process control takes over.

Note:

- This function block cannot run when the "PumpComb" $=3$, DOL mode in
"PmpConfiguration". Soft filling needs the VFD drive to run and execute gradual filling of the water.
- In case of master pump failure when process is running (pumps running):
- If softfill - disabled, next pump to start in sequence is becoming master and running in the normal mode.
- If softfill- enabled, next pump to start in sequence is becoming master but it is not going to the normal mode.

Softfill will only be performed for the master pump when there are no pumps running and the process is started for the first time.

### 6.5.13.7 Simulation functions

For the testing of the program in test environment where the real drives or pumps and tanks are missing, these three simulation blocks are provided which can be used.

The outputs of the function block may or may not be as accurate as real process. Use this function block only for simulation testing and not on the actual system.

### 6.5.13.7.1 Pump DOL simulation block

The function block "PmpDolSimu" is a simple simulation of the direct online pumps. This block can be used to test the behavior of the system when there is start and stop of pumps and external faults are introduced.

### 6.5.13.7.2 Pump drive simulation block

The function block "PmpDriveSimu" is used to simulate the variable frequency drive (VFD) pumps and for PLC communication. This block can be used to simulate the behavior of the VFDs to check the behavior of system during maximum speed or power, external faults and to check Start and stop sequence of the pumps.

### 6.5.13.7.3 Pump tank simulation block

The function block "PmpTankSimu" is used to simulate the water tank. This simulation can be used in all the three process control modes:

- Pressure
- Flow
- Level

This block can be used to check the start and stop sequence of the pumps and fault conditions during process control mode when the tank is not available by providing the dimensions of the tank, it can be used in combination with "PmpDolSimu" and "PmpDriveSimu" function blocks to simulate a test environment.

### 6.5.14 Reference, function blocks

Reference documentation
② Chapter 6.5.3.3 "View embedded documentation of all libraries" on page 3775

- BlockGetData
- _BlockGetPool
- _CloneMessage
- CreateArrayReceiver
- CreateIdAreaReceiver
- CreateMaskReceiver
- CreateMessage
- CreateSingleIdReceiver
- _DeleteReceiver
- _DisableSyncService
- DriverClose
- DriverGetSize
- DriverOpenH
- DriverOpenP
- EnableSyncService
- FlatCreateH
- FlatCreateP
- _FlatDelete
- FlatDisable
- FlatEnable
- FlatGetSize
- FlatRead
- FlatTest
- _FlatUpdate
- _FreeMessage
- GetBaudrate
- GetBusAlarm
- GetBusload
- GetBusState
- GetCiAState
- _GetDiagnosis
- _GetLostCounter
- GetMessageDataPointer
- GetMessageld
- GetMessageLength
- GetMsgCount
- GetNetld
- _GetReceiveCounter
- _GetReceiveErrorCounter
- GetReceivePoolSize
- GetReceiveQueueLength
- GetTimeStamp
- GetTransmitCounter
- GetTransmitErrorCounter
- GetTransmitPoolSize
- _GetTransmitQueueLength
- Is29BitldMessage
- IsRTRMessage
- IsSendingActive
- IsTransmitMessage
- JobAbort
- JobClose
- JobExecute
- JobGetld
- JobGetParams
- JobGetState
- JobOpen
- _JobOpenEx
- JobReset
- JobSetState
- LostMessages
- MsgAddRef
- MsgClone
- MsgGetData
- _MsgReceive
- _MsgRelease
- MsgReleaseEx
- MsgSend
- PoolCreateH
- PoolCreateP
- PoolDelete
- _PoolExtendH
- _PoolGetBlock
- PoolGetBlockSize
- PoolGetCurCapacity
- PoolGetNumBlocksLeft
- PoolGetSize
- PoolPutBlock
- _Read
- _ReadArrayReceiver
- RegisterldArea
- ResetBusAlarm
- RLstAddPrio
- RLstCheckPrio
- RLstCreateH
- _RLstCreateP
- _RLstDelete
- RLstGetHighestPrio
- RLstGetSize
- RLstRemovePrio
- SetCiAState
- StorageGetIndexId
- StorageGetTableId
- _UnregisterldArea
- WorkerRegister
- WorkerUnregister
- Write
- XChgCreateH
- XChgCreateP
- XChgDelete
- XChgExtendH
- _XChgGetSize
- XChglsEmpty
- XChgMsgLeft
- AbbETrig
- AbbETrig2
- AbbETrig3
- AbbLCon
- AbbLCon 2
- AbbLCon3
- AbbLConA
- AbbLConC
- AbbLConC2
- AbbLConC3
- AbbLConCA
- AbsoluteAddressInfo
- AbstrTreeNode
- AC500Device
- AC500DeviceDiag
- AC500DeviceInfo
- AC500DiagEvent
- ACCESS MODE
- AccessRights
- AcknowledgeRequestBuilder
- ActionController
- AdapterDiagnosis
- ADAPTERSTATE
- AdapterState
- AddMultiplicatedVector
- $A D D R$
- aDDR TO ID
- aDDR TYPE
- AddressArea
- AddressGeneratedlnfo
- AddressInfoType
- AddressInfoUnion
- AddressLeafTreeNode
- AffectedSourcesHelp
- AINFO TYPE
- ALARM ID
- AlarmFctWriteLatchVariable
- ALARMGROUP ID
- AlarmIndices
- AlarmLatchAdapter
- AlarmSelectionInfo
- AlarmSelectionInfoDefault
- AlarmState
- AlarmStateTransition
- AlarmStorageConvertFromTimestamp
- AlarmStorageConvertToTimestamp
- AlarmStorageConvertValueToLREAL
- AlarmStorageConvertValueToREAL
- AlarmStorageConvertValueToString
- AlarmStorageGetDefaultText
- AlarmStorageGetMessageCount
- AlarmStorageGetTextld
- AlarmStorageGetTextListName
- AlarmStorageLatchVariable
- AlarmStorageLimit
- AlarmStorageReader
- AlarmStorageRow
- AlarmStorageStaticData
- AlarmToAcknowledge
- AlarmType
- AllocAndCopyPString
- Allocator
- AllScalarsUnion
- AnalogDeviceDescType
- AnalyzeExpression
- AnalyzeExpressionCombined
- AnalyzeExpressionTable
- APP MEMORY SEGMENT
- APP NAME
- AppCallGetProperty
- AppCallGetProperty2
- AppCallGetProperty2Release
- AppCallGetProperty3
- AppCallSetProperty
- AppCallSetProperty2
- AppendToString
- AppFindApplicationByName
- AppGenerateException
- AppGetApplicationByAreaAddress
- AppGetApplicationFlags
- AppGetApplicationInfo
- AppGetAreaAddress
- AppGetAreaOffsetByAddress
- AppGetAreaPointer
- AppGetAreaSize
- AppGetCurrent
- AppGetFirstApp
- AppGetNextApp
- AppGetProjectInformation
- AppGetSegment
- AppGetSegmentAddress
- AppGetSegmentSize
- APPLICATION
- APPLICATION INFO
- Apply Attributes
- AppNumOfActiveSessions
- AppRegisterPropAccessFunctions
- AppReset
- AppRestoreRetainsFromFile
- AppStartApplication
- AppStopApplication
- AppStoreRetainsInFile
- $A R$ Info
- AREA TYPE
- ARInfo
- ARP Packet
- ARRAY_RECV_ENTRY
- ArrayDimension
- ArrayRangeHelper
- AsciiUpper
- AskCredentialsHelper
- ASM IWORKER
- ASM STATE
- Assert
- AssignerBase
- ASYNC JOB ERROR ID
- AsyncAdd
- AsyncBaseClass
- AsyncGetJobReturnValue
- ASYNCJOB EVENTPARAM
- ASYNCJOB HOOKPARAM
- ASYNCJOB PARAM
- ASYNCJOB TASKPARAM
- AsyncKill
- AsyncProperty
- AsyncRemove
- AsyncRemoveAll
- atan2 func
- AtomicReadLInt
- AtomicReadLReal
- AtomicReadLTime
- AtomicReadLWord
- AtomicReadULInt
- AtomicWriteLInt
- AtomicWriteLReal
- AtomicWriteLTime
- AtomicWriteLWord
- AtomicWriteULInt
- ATTRIB
- Axis Parameter
- Axis_Ref
- BACnetAccumulator
- BACnetAcknowledgeAlarm
- BACnetAddListElement
- BACnetAnaloginput
- BACnetAnalogOutput
- BACnetAnalogValue
- BACnetAsyncTransactionToken
- BACnetAveraging
- BACnetBackupBACnetDevice
- BACnetBinaryInput
- BACnetBinaryOutput
- BACnetBinaryValue
- BACnetBitStringGetBit
- BACnetBitStringSetBit
- BACnetCalendar
- BACnetCanceIPendingConfirmedRequest
- BACnetClientAcknowledgeAlarm
- BACnetClientAddListElement
- BACnetClientBackupBACnetDevice
- BACnetClientBase
- BACnetClientConfPrivateTransfer
- BACnetClientConfTextMessage
- BACnetClientCreateObject
- BACnetClientDeleteObject
- BACnetClientDeviceCommControl
- BACnetClientGetAlarmSummary
- BACNetClientGetEnrollmentSummary
- BACnetClientGetEventInfo
- BACnetClientLifeSafetyOperation
- BACnetClientReadAlIPropertyDataContents
- BACnetClientReadProperty
- BACnetClientReadPropertyMultiple
- BACnetClientReadRange
- BACnetClientReadStreamFile
- BACnetClientReinitializeDevice
- BACnetClientRemoveListElement
- BACnetClientRestoreBACnetDevice
- BACnetClientSubscribeCOV
- BACnetClientSubscribeCOVProperty
- BACnetClientTimeSynchronization
- BACnetClientUTCTimeSynchronization
- BACnetClientWriteProperty
- BACnetClientWritePropertyMultiple
- BACnetClientWriteStreamFile
- BACnetCloseClientCustomer
- BACnetCommand
- BACnetConfPrivateTransfer
- BACnetConfTextMessage
- BACnetCreateObject
- BACnetCreateObjectResult
- BACnetDateRange
- BACnetDateTime
- BACnetDateTimeCmp
- BACnetDateTimeToString
- BACnetDeleteObject
- BACnetDevice
- BACnetDeviceCommControl
- BACnetDevObjPropReference
- BACnetEventEnrollment
- BACnetEventLog
- BACnetFile
- BACnetGetAlarmSummary
- BACnetGetClientDeviceCommunication
- BACnetGetEnrollmentSummary
- BACnetGetEventInfo
- BACnetGlobalGroup
- BACnetGroup
- BACnetIntegerValue
- BACnetIPdatalink
- BACnetLargeAnalogValue
- BACnetLifeSafetyOperation
- BACnetLifeSafetyPoint
- BACnetLifeSafetyZone
- BACnetLoop
- BACnetMSTPdatalink
- BACnetMultistateInput
- BACnetMultistateOutput
- BACnetMultistateValue
- BACnetNotificationClass
- BACnetObjectBase
- BACnetOpenClientCustomer
- BACnetPersistenceInfo
- BACnetPositiveIntegerValue
- BACnetProgram
- BACnetPropertyAttributeExistent
- BACnetPropertyAttributePersistent
- BACnetPropertyAttributes
- BACnetPropertyAttributeWritable
- BACnetPulseConverter
- BACnetReadAllPropertyDataContents
- BACnetReadFile
- BACnetReadProperty
- BACnetReadPropertyMultiple
- BACnetReadRange
- BACnetRegisterClientCommunicationStateCallback
- BACnetRegisterClientDataPoint
- BACnetRegisterClientEventNotification
- BACnetRegisterClientUnsubscribeCompletionCallback
- BACnetReinitializeDevice
- BACnetRemoveListElement
- BACnetRestartAllClients
- BACnetRestoreBACnetDevice
- BACnetSchedule
- BACnetServer
- BACnetServerConfCOVNotification
- BACnetServerConfEventNotification
- BACnetServerPluginBase
- BACnetServerPluginCallbackBase
- BACnetServerPluginHookBase
- BACnetSetClientDeviceCommunication
- BACnetSetClientDeviceFixAddress
- BACnetSetClientDeviceFixSubscribeCovTime
- BACnetSetClientGlobalCommTimingParameters
- BACnetSetClientGlobalMaxDeviceActions
- BACnetSetpointReference
- BACnetStructuredView
- BACnetSubscribeCOV
- BACnetSubscribeCOVProperty
- BACnetTimeStamp
- BACnetTimeStampUnion
- BACnetTrendLog
- BACnetTrendLogMultiple
- BACnetUnconfPrivateTransfer
- BACnetUnconfTextMessage
- BACnetUnregisterClientDataPoint
- BACnetUnregisterClientEventNotification
- BACnetWhoHas
- BACnetWhols
- BACnetWriteFile
- BACnetWriteGroup
- BACnetWriteProperty
- BACnetWritePropertyMultiple
- BASE64
- BaseMap
- BaseVector
- BBMD Info
- BCD TO BYTE
- BCD TO DWORD
- BCD TO INT
- BCD TO WORD
- BehaviourModel
- BehaviourModelBase
- BIT_AS_BYTE
- BIT AS DWORD
- BIT AS WORD
- BitCpy
- BitRelativeAddressInfo
- BLINK
- BlkClass
- BLOB
- BlobAlloc
- BlobCopyToData
- BlobFree
- BlockGetData
- BlockGetPool
- BmpPoolBegin
- BmpPoolEnd
- BmpPoolRegister
- BmpPoolUnRegister
- BOLT
- BoolElement
- BoolElementFactory
- BranchNamedTreeNode
- BranchTreeNode
- BranchTreeNodeOpcUA
- BrowseOperation
- BTagAlignment
- BTagElementType
- BTagReaderCreate
- BTagReaderCreateDynamic
- BTagReaderDestroy
- BTagReaderGetComplexContent
- BTagReaderGetContent
- BTagReaderGetString
- BTagReaderGetTagld
- BTagReaderGetTagLen
- BTagReaderInit
- BTagReaderIsDataTag
- BTagReaderMoveNext
- BTagReaderPeekNext
- BTagReaderSkipContent
- BTagSwapHeader
- BTagWriterAppendBlob
- BTagWriterAppendDummyBytes
- BTagWriterAppendFillBytes
- BTagWriterAppendRaw
- BTagWriterAppendWString
- BTagWriterCreate
- BTagWriterCreateDynamic
- BTagWriterCreateSavePoint
- BTagWriterCreateSavePointDynamic
- BTagWriterDestroy
- BTagWriterDestroySavePoint
- BTagWriterEndTag
- BTagWriterFinish
- BTagWriterInit
- BTagWriterInit2
- BTagWriterRestoreSavePoint
- BTagWriterStartTag
- BTagWriterSwitchBuffer
- Buffer
- BufferPool
- BufferPoolFactoryArgs
- BufferPoolFactoryBase
- BufferToString
- BuildAndEnqueueV3Request
- BUS INFO
- BUS STATE
- BUS TYPE
- BusScanConfHeader
- BUSSTATE
- BYTE_AS_BIT
- BYTE_TO_BCD
- BYTE TO GRAY
- BYTE TO HEXinASCII
- ByteBuffer
- ByteOrder
- C TS Type
- CAADiagDeviceDefault
- CAADiagTreeBase
- CAAReconfigureBase
- CalcHesseRepresentation
- CalcRootLin
- CalcRootParable
- CalculatePropertyBufferSize
- CallbackNetVar
- CallbackTaskCodeNC
- CallFunctionByIndex
- CallGlueDeserializeParameters
- CallGlueFunctionParameterSet
- CallGlueSerializeReturnValues
- CamExtended
- CANbus
- CANbus Diag
- CANDiagnosis
- CANOPEN KERNEL ERROR
- CANOPEN KERNEL STATE
- CANOpenDiagnosisInfo
- CANopenEvent
- CANopenManager
- CANopenManager_Diag
- CanReconfigure
- CANRemoteDevice
- CANRemoteDevice Diag
- CANRemoteModule Diag
- CartesianToPolar
- CaseSensitiveNamedTreeNode
- CB CALLBACK
- $\quad$ CCB
- CD522DoubleWordCounter
- CD522Encoder32Bit
- CD522FreqOut
- CD522FreqScan
- CD522FreqScan PLUS
- CD522In
- CD522OneWordCounter
- CD522Out
- CD522PulseOut
- CD522PwmOut
- CD522ReadInput
- CD522SsiCnt
- CD522SsiCnt PLUS
- CD522TwoWordCounters
- CD522WriteOutput
- CDSV3Request
- Ceil
- CeilF
- CERT_INFO
- ChainBuffer
- ChanneIDiagnosisData
- ChannelErrorType
- ChannelProperties
- ChannelProperties Type
- CharBufferPtr
- CharBufferString
- CHARCURVE
- CharCurve DINT
- CharCurve LREAL
- CharToUpper
- CHCAddressComponent
- CHCAddressType
- CHCPeerAddress
- CHCProtocolDataUnit
- Check
- CheckExpSubmodule
- CIFX APPLICATION CHANNEL INFO
- CIFX BOARD
- CIFX BOARD INFORMATION
- CIFX_CHANNEL
- CIFX CHANNEL INFO BLOCK
- CIFX CHANNEL INFORMATION
- CIFX COM DIAGNOSTICS
- CIFX COMMON STATUS BLOCK
- CIFX COMMON STATUS BLOCK MASTER
- CIFX COMMUNOICATION CHANNEL INFO
- CIFX_DEV_INFO
- CIFX_DIRECTORY_ENTRY
- CIFX ERROR FIELD
- CIFX GetBusActivationBeforeReset
- CIFX GETSLAVECONNECTINFO REQ
- CIFX GETSLAVEHANDLE CONF
- CIFX GETSLAVEHANDLE REQ
- CIFX HANDSHAKE CHANNEL INFO
- CIFX INDICATION PARAM
- CIFX MASTER DIAG
- CIFX MAX PACKET
- CIFX MEMORY INFORMATION
- CIFX_PACKET
- CIFX ResetConfigApplication
- CIFX SYSTEM CHANNEL INFO
- CIFX SYSTEM INFO BLOCK
- CIFX xChannelBusState
- CIFX xChanneIClose
- CIFX xChannelCommonStatusBlock
- CIFX_xChannelConfigLock
- CIFX_xChannelControlBlock
- CIFX xChanneIDownload
- CIFX xChannelExtendedStatusBlock
- CIFX xChannelFindFirstFile
- CIFX xChannelFindNextFile
- CIFX xChannelGetMBXState
- CIFX_xChannelGetPacket
- CIFX_xChanneIGetPacketTimeout
- CIFX xChannelGetSendPacket
- CIFX xChannelHostState
- CIFX xChannellnfo
- CIFX xChannelIOInfo
- CIFX xChannelIORead
- CIFX_xChannelIOReadSendData
- CIFX_xChannellOWrite
- CIFX xChanneIOpen
- CIFX xChannelOpen2
- CIFX xChanneIPLCActivateRead
- CIFX xChanneIPLCActivateWrite
- CIFX xChanneIPLCIsReadReady
- CIFX_xChanneIPLCIsWriteReady
- CIFX_xChanneIPLCMemoryPtr
- CIFX xChanneIPutPacket
- Cifx xChanneIRegisterRecvCallback
- CIFX xChanneIReset
- CIFX xChanneISetPacketTimeout
- CIFX xChannelUpload
- CIFX xChannelUserBlock
- CIFX_xChannelWatchdog
- CIFX xDriverClose
- CIFX xDriverEnumBoards
- CIFX xDriverEnumChannels
- CIFX xDriverGetErrorDescription
- CIFX xDriverGetInformation
- CIFX xDriverMemoryPointer
- CIFX_xDriverOpen
- CIFX_xMemCpy
- CIFX xSysdeviceClose
- CIFX xSysdeviceDownload
- CIFX xSysdeviceFindFirstFile
- CIFX xSysdeviceFindNextFile
- CIFX xSysdeviceGetMBXState
- CIFX xSysdeviceGetPacket
- CIFX xSysdeviceInfo
- CIFX xSysdeviceOpen
- CIFX xSysdevicePutPacket
- CIFX xSysdeviceReset
- CIFX_xSysdeviceUpload
- CiModCi52x
- CiModCiClusterDiag
- CiModClusterDiag
- CiModClusterStatus
- CiModCmdQueue
- CiModCmdQueueInput
- CiModDataAC522
- CiModDataAI523
- CiModDataAI531
- CiModDataAI561
- CiModDataAI562
- CiModDataAI563
- CiModDataAO523
- CiModDataAO561
- CiModDataAX521
- CiModDataAX522
- CiModDataAX561
- CiModDataCD522
- CiModDataCl521
- CiModDataCl522
- CiModDataDA501
- CiModDataDA502
- CiModDataDC522
- CiModDataDC523
- CiModDataDC532
- CiModDataDC561
- CiModDataDC562
- CiModDataDI524
- CiModDataDI561
- CiModDataDI562
- CiModDataDI571
- CiModDataDI572
- CiModDataDO524
- CiModDataDO526
- CiModDataDO561
- CiModDataDO562
- CiModDataDO571
- CiModDataDO572
- CiModDataDO573
- CiModDataDX522
- CiModDataDX531
- CiModDataDX561
- CiModDataDX571
- CiModDiag
- CiModDiagModInfo
- CiModDiagTableType
- CiModInput
- CiModParaAC522
- CiModParaAI523
- CiModParaAI531
- CiModParaAI561
- CiModParaAI562
- CiModParaAI563
- CiModParaAO523
- CiModParaAO561
- CiModParaAX521
- CiModParaAX522
- CiModParaAX561
- CiModParaCD522
- CiModParaCl521
- CiModParaCl522
- CiModParaDA501
- CiModParaDA502
- CiModParaDC522
- CiModParaDC523
- CiModParaDC532
- CiModParaDC561
- CiModParaDC562
- CiModParaDI524
- CiModParaDI561
- CiModParaDI562
- CiModParaDI571
- CiModParaDI572
- CiModParaDO524
- CiModParaDO526
- CiModParaDO561
- CiModParaDO562
- CiModParaDO571
- CiModParaDO572
- CiModParaDO573
- CiModParaDX522
- CiModParaDX531
- CiModParaDX561
- CiModParaDX571
- CIPClass
- CIPCommonService
- CIPHER LIST
- CLASS INFO
- ClassCreate
- ClassDelete
- ClassFree
- CLClient
- CLClientOptions
- CLClientState
- CLIENT ACCEPT
- CLIENT REPLY
- CLOCK
- CLOCK_DT
- CloneMessage
- Close
- CLRequestState
- CLServer
- CLServerOptions
- CM5610 2RS
- CM5640 2ETH
- CM579EcatBusDiag
- CM579EcatBusGetDCMaxDeviation
- CM579EcatBusSetState
- CM579EcatBusStart
- CM579EcatBusStop
- CM579EcatCoeRead
- CM579EcatCoeWrite
- CM579EcatGetExtSyncInfo
- CM579EcatMasterGetCPULoad
- CM579EcatMasterGetFrameLossCount
- CM579EcatMasterGetMemInfo
- CM579EcatMasterGetThresholdCount
- CM579EcatMasterGetTimingInfo
- CM579EcatRegisterRead
- CM579EcatRegisterWrite
- CM579EcatScanTopology
- CM579EcatScanTopologyStop
- CM579EcatSIvDiag
- CM579EcatSIvGetDCInfo
- CM579EcatSIvGetMDPModules
- CM579EcatSIvGetState
- CM579EcatSIvReadESCVersion
- CM579EcatSIvReadLostLinkCnt
- CM579EcatSIvReadRxErrorCnt
- CM579EcatSIvSetState
- CM579EcatSoeRead
- CM579EcatSoeWrite
- CM579PnioBlockHeader
- CM579PnioCommState
- CM579PnioDeviceState
- CM579PnioDevName
- CM579PnioGetCntrIState
- CM579PnioGetDevState
- CM579PniolM0Data
- CM579PniolM0SwRev
- CM579PniolMOVersion
- CM579PniolmplicitRead
- CM579PnioRead
- CM579PnioReadIM0Data
- CM579PnioStartCom
- CM579PnioStopCom
- CM579PnioWrite
- CM582ProfibusDeviceInfoType
- CM592BaseCommonReq
- CM592BaseStatusBlockReq
- CM592CommErrorInfo
- CM592CommStatus
- CM592Control
- CM592DPV1Masc1Read
- CM592DPV1Masc1Write
- CM592ExtDiagData
- CM592ReadInput
- CM592ReadOutput
- CM592SlaveDiagnosis
- CM592SlaveStates
- CM592StateBits
- CM592StationStatus 1
- CM592StationStatus 2
- CM592StationStatus 3
- CM592SystemDiagnosis
- Cm598Base
- Cm598CanInfo
- Cm598CanInfoType
- Cm598CanMessageDataType
- Cm598CanMessageType
- Cm598CanMsgRec
- Cm598CanMsgRecDirect
- Cm598CanMsgRecEvt
- Cm598CanMsgRecMult
- Cm598CanMsgSend
- Cm598CanopenComErrorType
- Cm598CanopenNmt
- CM598CanopenNodeStates
- Cm598CanopenResetErrors
- Cm598CanopenSdoRead
- Cm598CanopenSdoWrite
- Cm598CanopenSetNodeMode
- Cm598CanopenState
- Cm598CanopenStateBitsType
- Cm598CanopenStateType
- Cm598CanopenSyncState
- Cm598CanopenSysDiag
- CM598DeviceInfoType
- Cm598NmtCmd
- Cm598NodeModeCmd
- Cm598NodeStateType
- CMAddComponent
- CMAddComponent2
- CMC_Axis_Actual
- CMC Axis Control Parameter
- CMC Axis InOut
- CMC Axis 10
- CMC Axis Simu
- CMC Axis User
- CMC Basic Kernel
- CMC_Basic_Kernel64
- CMC Binary2Modulo
- CMC Get Units From Inc
- CMC Linear Kernel
- CMC Load InOut
- CMC Load Motion Kernel
- CMC Load Super
- CMC_Modulo2Binary
- CMC PidT1
- CMC Pos Control
- CMC SInterPolation
- CMC SIPosiLoop
- CMC XYGroupRef
- CMExitComponent
- CMGetComponentByName
- CMGetCoreVersion
- CMInitComponent
- CmpLogAsyncFB
- CmpTIsAccept
- CmpTIsBufferDataReceived
- CmpTIsBufferDataSent
- CmpTIsBufferDataToSendAvailable
- CmpTIsBufferOpen
- CmpTIsClose
- CmpTIsConnect
- CmpTIsCreateContext
- CmpTIsCreateContext2
- CmpTIsFreeContext
- CmpTIsMethod
- CmpTIsRead
- CmpTIsShutdown
- CmpTIsWrite
- CMRemoveComponent
- CMShutDown
- CMUtIcwstrcpy
- CMUtISafeStrCpy
- CMUtIStrICmp
- CMUtIUtf8ToW
- CMUtlwstrcpy
- CMUtIWToUtf8
- CNCT
- CNT_IO_EXT
- COBID
- CodeMClose
- CodeMDecrementUnitCounter
- CodeMDecrypt
- CodeMEncrypt
- CodeMGetContentByFirmcode
- CodeMGetContentByFirmcode2
- CodeMGetExpirationTime
- CodeMGetFeatureMapByFirmcode
- CodeMGetFirst
- CodeMGetInfo
- CodeMGetName
- CodeMGetNext
- CodeMGetQuantity
- CodeMGetUnitCounter
- CodeMOpen
- CodeWriter
- COLLECTION ERROR
- COM CFG FORMAT ENUM
- COM MOD FCT22 TYPE
- COM_MOD_FCT23_TYPE
- COM_PORT_ID
- CombineDateTime
- ComGetCaaSerialComConfig
- ComGetldByName
- CommandManager
- CommStatus
- CommunicationSettings
- Compare
- CompareString
- CompareWString
- CompatibilitySafeGetPrepareExitCommProcessingLastCall
- CompatibilitySafeSetPrepareExitCommProcessingFurtherCallNecessary
- ComponentBase
- ComponentPseudo
- ComponentRenamed
- ComponentSimple
- CONCAT
- ConfigError
- ConfigGetConnector
- ConfigGetFirstChild
- ConfigGetFirstConnector
- ConfigGetNextChild
- ConfigGetNextConnector
- ConfigGetParameter
- ConfigGetParameterLength
- ConfigGetParameterValueByte
- ConfigGetParameterValueDword
- ConfigGetParameterValuePointer
- ConfigGetParameterValueWord
- ConnectionSetup
- Connector
- ConnectorFlagController
- ConnectorState
- ContainerSizeProvider
- ContentFeatureFlags
- ControllerConfigUtil
- ControllerState
- ConvertNSecToTick
- ConvertSystimedateToUTC
- ConvertSystimedateUsingLDate
- ConvertSysTimeValueToLWord
- ConvertTickToNSec
- ConvertTickToUSec
- ConvertTimestampToLDateAndTime
- ConvertToIBranchTreeNode
- ConvertTolLeafTreeNode
- ConvertToITreeWalker
- ConvertUSecToTick
- ConvertUTF16toUTF8
- ConvertUTF8toUTF16
- Copy
- CopyBufferData
- CopyVUM User
- CopyVUM User3
- COUNT
- COUNT_TO_UDINT
- COUNT TO UINT
- COUNT TO ULINT
- CPU PROD READ ASYNC
- CpuCoreBindingDesc
- CpuCoreBits
- CRC16 CCITT
- CRC16 generic
- CRC16 Modbus
- CRC16 standard
- CRC16Finish
- CRC16Init
- CRC16Update
- CRC32
- CRC32Finish
- CRC32Init
- CRC32Update
- CRC32Update2
- CREATEID
- CreateBuffer
- CreateIdAreaReceiver
- CreateMaskReceiver
- CreateMessage
- CreatePacket
- CreatePacketPool
- CreateSegment
- CreateSegmentPool
- CreateSingleIdReceiver
- CreateTextFromString
- CreateTextFromWString
- CredentialsHandling
- CrossProduct
- CrossProductNormed
- CryptoAsymmetricDecrypt
- CryptoAsymmetricEncrypt
- CryptoDeletePrivateKey
- CryptoDeriveKey
- CryptoExportAsymmetricKey
- CryptoExportRawAsymmetricKey
- CryptoGenerateAsymmetricKeyPair
- CryptoGenerateHash
- CryptoGenerateRandomNumber
- CryptoGetAlgorithmByld
- CryptoGetAsymmetricKeyLength
- CryptoGetFirstAlgorithm
- CryptoGetNextAlgorithm
- CryptoHMACSign
- CryptoHMACVerify
- CryptolmportAsymmetricKey
- CryptolmportRawAsymmetricKey
- CryptoKeyAgreement
- CryptoKeyExit
- CryptoKeyInit
- CryptoLoadPrivateKey
- CryptoRtsByteStringExit
- CryptoRtsByteStringInit
- CryptoRtsByteStringInit2
- CryptoSignatureGenerate
- CryptoSignatureVerify
- CryptoStorePrivateKey
- CryptoSymmetricDecrypt
- CryptoSymmetricEncrypt
- CTD
- CtrIFilterContinuous
- CtrIFilterDiscrete
- CtrIIntegrator
- CtrILag
- CtrILagDelay
- CtrlLead
- CtrILeadLag
- CtrIPidParallel
- CtrlPiPulse
- CtrlRampStep
- CTU
- CTUD
- CustomRequestQueue
- CustomRequestResponse
- CWCHAR
- DATA
- DatabaseSizeLimit
- DataCopyToBlob
- Dataltem
- DataltemAndPtrVectors
- DataltemBase
- DataltemList
- DataltemListPublic
- DataltemListPublicPersistant
- DataltemLocation
- DataltemPtrVector
- DataltemVector
- DataRepresentation
- DataServer StructReplacementInfo
- DataServerFlags
- DataServerRequestldentification
- Datasource
- DataSourceError
- DataSourceMonitoringState
- Datasources
- DatasourcesAction
- DatasourcesActionRecord
- DataSourceShutdownFlags
- DatasourceShutdownInfo
- DatasourcesMgr
- DataSourcesQualityChecker
- DataSourceState
- DataSourceTransactionResult
- DateConcat
- DateSplit
- DateTime
- DateTimeFromWeek
- DateTimeProvider
- DAY
- DayOfWeek
- DAYS
- DCP DeviceData
- DCP DeviceRole
- DCP Error
- DCP FilterData
- DCP FilterMode
- DCP FilterOptions
- DCP Get
- DCP_GetOptions
- DCP Identify
- DCP Reset
- DCP ResetMode
- DCP Set
- DCP SetData
- DCP SetOptions
- DeallocStackAllocatedContentBuffer
- DebugltfAddrToltfPtr
- Decode
- DECODE IOL STATUS
- DecodeClass
- DecodeEvent
- DefaultAlarmFilterCriteria
- DELETE
- Delete
- DeleteBuffer
- DeleteReceiver
- DERIVATIVE
- Derivative
- DeserializeHexReal
- DetermineVarTypeOfArray
- DetermineVarTypeOfStruct
- DetermineVarTypeOfTypeDesc
- DEVICE
- DEVICE INFO
- DEVICE STATE
- DEVICE TRANSITION STATE
- DEVICE TYPE
- DeviceAR
- DeviceAR State
- DeviceConfigUtil
- DeviceInfo
- Devicelterator
- DeviceState
- Diag
- DIAG_HISTORY_TXT_TYPE
- DIAG TXT TYPE
- DIAG VAL TYPE
- DiagHistory
- DiagHistoryValToTxt
- DiagMessageFactory
- DIAGNOSIS INFO
- DiagnosisDataBuffer
- DiagnosisDataReader
- DiagnosisDirection
- DiagnosisInformationUSI
- DiagnosisRecordIndex
- DiagnosisSeverity
- DiagnosisSource
- DiagnosisSpecifier
- DiagSignalEventlec
- DiagValToTxt
- DiagValTypeGetDeviceType
- DiagValTypeGetTextList
- DiagVerifyTextListCallback
- Dialog FormatTypes
- DialogChecklnit
- DialogCheckInitForExponential
- DialogCheckInitForExponentialW
- DialogCheckInitW
- DINT TO SIGNED
- DintElement
- DintElementFactory
- DintSetBitBased
- DintSetFull
- DintToDintMap
- DintVector
- DirClose
- DirCopy
- DirCreate
- DirectAssigner
- DirectlOBits16
- Direct/OBits 8
- DirFileTime
- Dirlnfo
- DirList
- DirOpen
- DirRemove
- DirRename
- DisableSyncService
- DM1 Read
- DM1 Write
- DM2_Read
- DM2 Write
- DownloadDestination
- DP ADDR
- DP AINFO
- DP DEVICE ID
- DP DIAG
- DP_StationStatus1
- DP_StationStatus1_Diag
- DP StationStatus2
- DP StationStatus2 Diag
- DP StationStatus 3
- DP StationStatus3 Diag
- DPSlaveDiag
- DPT10
- DPT10IEC_to_KNX
- DPT10 KNX to IEC
- DPT16 IEC to KNX
- DPT16 KNX to IEC
- DPT19
- DPT19 IEC to $K N X$
- DPT19 KNX to IEC
- Driver
- DriverCfg
- DriverClose
- DriverDiag
- DriverGetSize
- DriverInitInfo
- DriverOpenH
- DriverOpenP
- DRV PDRIVE PRM REQ ERROR
- DRV PDRIVE PRM TYPE
- DrvControlACS
- DrvControlCANCiA402
- DrvControIDCS
- DrvControlModbusACS
- DrvControlModbusDCS
- DrvControlModbusEng
- DrvDataType
- DrvDataTypeInternal
- DrvModbusRead
- DrvModbusReadWrite23
- DrvModbusRtu
- DrvModbusRtuBroadcast
- DrvModbusTcp
- DrvModbusWrite
- DrvModFct23Type
- DrvModMastType
- DrvModPara32Bit
- DrvPdPrmDpv1DataType
- DrvPnRead
- DrvPnWrite
- DrvScaling
- DS DISK STATUS
- DS EOL INFO
- DS LIFETIME_USED
- DT TO INT64
- DT TO ISO8601
- DT TO REAL8
- DT to Timestamp
- DT to Timestamp2
- DTC
- DTCBufferWriter
- DTCLogger
- DTConcat
- DTCProvider
- DTR CONTROL
- DTSplit
- DTU GETDATEANDTIME PARAMS
- DTU GETTIMEZONEINFORMATION PARAMS
- DTU SETDATEANDTIME_PARAMS
- DTU_SETTIMEZONEINFORMATION_PARAMS
- Dummy
- DummyJob
- DURATION
- DURATION TO LTIME
- DURATION TO TIME
- DWORD AS BIT
- DWORD TO BCD
- DWORD TO GRAY
- DWORD TO HANDLE
- DWORD TO IDENT
- DWORD_TO_PVOID
- DwordVector
- DynamicTextChangeLanguage
- DynamicTextGetCurrentLanguage
- DynamicTextGetDefaultText
- DynamicTextGetDefaultTextW
- DynamicTextGetText
- DynamicTextGetTextW
- DynamicTextlterateIndices
- DynamicTextLoadDefaultTexts
- DynamicTextRegisterFile
- DynamicTextRegisterPath
- DynamicTextReloadTexts
- DynamicTextRetrieveIndices
- DynamicTextUnRegisterFile
- DynamicTraceLoader
- DynamicTraceLoaderRemote
- EAlarmStorageReaderErrors
- EAlarmTableParts
- EAlarmType
- ECAT 402ParameterHoming APP
- ECAT_CiA402_Control_App
- ECAT_CiA402_TouchProbe_App
- ECAT CiA Object App
- ECAT DecodeCouplerError
- ECAT HomingOnTouchProbe APP
- ECAT Read Byte App
- ECAT Read Coe List App
- ECAT_Read_DInt_App
- ECAT_Read_Int_App
- ECAT Write Byte App
- ECAT Write Coe List App
- ECAT Write DInt App
- ECAT Write Int App
- EcatBusDiag
- EcatBusGetDCMaxDeviation
- EcatBusSetState
- EcatCoeRead
- EcatCoeWrite
- EcatDeviceldentification
- EcatDeviceInfoData
- EcatDeviceTypeIdentification
- EcatGetExtSyncInfo
- EcatMasterGetCPULoad
- EcatMasterGetFrameLossCount
- EcatMasterGetMemInfo
- EcatMasterGetThresholdCount
- EcatMasterGetTimingInfo
- EcatRegisterRead
- EcatRegisterWrite
- EcatScanTopology
- EcatScanTopologyStop
- EcatSIvDiag
- EcatSIvGetDCInfo
- EcatSIvGetMDPModules
- EcatSIvGetState
- EcatSIvReadESCVersion
- EcatSIvReadLostLinkCnt
- EcatS/vReadRxErrorCnt
- EcatS/vSetState
- EcatSoeRead
- EcatSoeWrite
- EcatStartCom
- EcatState
- EcatStopCom
- EcatSync
- EcatVendor
- EcatVendorIDList
- EcatVendorName2Device
- ECM_IF_DC_CONTROL_STATUS_E
- ECM_IF_GET_SLAVE_DC_INFO_FLAGS_E
- ECO DisableMultiPin
- EColorSetting
- ECUSTATE
- EDBActiveIndex
- EDBType
- eDeviceState
- EdgeTriggeredBehaviourModelBase
- EdgeTriggeredTimingControlledBehaviourModelBase
- EEthernetState
- eFastCounter
- EFillingStyle
- EFilterCriteriaActivity
- EFilterLatchContent
- EFilterTimeRangeType
- ElmageStyle
- EIP CloseClass3Connection
- EIP OpenClass3Connection
- EIP SendClass3ConnectedMessage
- EIP SendUnconnectedMessage
- ElaborateLatchFilterCriteria
- ElaborateTimeRangeFilterCriteria
- Element
- EMCY DATA
- EMCY ERROR
- eModulName
- EnableSyncService
- Encode
- EncodeSpec
- ENDIANESS
- EnqueuedRequest
- ENUM61850 BASIC TYPES
- ENUM61850 CLOCK SYNC MODE
- ENUM61850 DataPoint Type
- ENUM61850 SIM MODE
- EnumAttributes
- EnumValues
- EnXYChartClientActity
- EnXYChartDataProviderAxisVar
- EnXYChartDataProviderCurveVar
- EnXYChartDataProviderVar
- EnXYChartUpdateType
- EOF
- eParaState
- ERectSetting
- ErrFlexConf
- ERROR
- ERRORID
- ERROR INFO
- ErrorCode
- ErrorCode1 RW
- ErrorCodesOB
- ErrorToString
- EShadowStyle
- ESpecial_FP_Value
- ETC_ADS_loLinkRead
- ETC ADS loLinkWrite
- ETC CO Emergency
- ETC CO ERROR
- ETC CO MODE
- ETC CO SdoInfoGeEntryDescription
- ETC_CO_SdoInfoGetObjectDescription
- ETC_CO_SdoInfoGetODList
- ETC CO SdoRead
- ETC CO SdoRead4
- ETC CO SdoRead Access
- ETC CO SdoRead Channel
- ETC CO SdoReadDWord
- ETC_CO SdoWrite
- ETC CO SdoWrite4
- ETC CO SdoWrite Access
- ETC CO SdoWriteDWord
- ETC FoE Download
- ETC FoE Upload
- ETC LASTERROR
- ETC MASTER STATE
- ETC SDO INFO LIST_TYPE
- ETC SDO INFO OBJECT CODE
- ETC SLAVE STATE
- ETC SoE Cmd
- ETC SOE ERROR
- ETC SoE IDNRead
- ETC SoE IDNRead4
- ETC_SoE_IDNWrite
- ETC SoE_IDNWrite4
- ETC VoE SendReceive
- ETCDeviceldentMode
- ETCERRORCODES
- ETCMasterStack
- ETCSlave
- ETCSlave Diag
- ETCSlaveStack
- ETH MOD FCT22 TYPE
- ETH MOD FCT23 TYPE
- EthDNSResolve
- EtherCATDevice
- EthercatMaster GetVersion
- EtherCatReconfigure
- EthernetState
- EthlcmpPing
- EthOwnIP
- EthOwnIPInfo
- EthSetOwnIP
- EthSetRtoMin
- ETHx ICMP PING
- ETHX MOD CONFIG
- ETHX MOD INFO
- ETHx MOD MAST
- ETHx OWN IP
- ETHx_OWN_IP_INFO
- ETraceAddressFlags
- ETraceGradientType
- ETrendStorageGraphType
- ETrendStoragePenStyle
- ETrendStorageReaderErrors
- ETrendStorageReaderStep
- ETrig
- ETrigA
- ETrigATI
- ETrigATITo
- ETrigATo
- ETrigTI
- ETrigTIA
- ETrigtito
- ETrigTo
- ETrigToA
- ETrigToTI
- ETrigTotiA
- EVENT
- EVENT CLASS
- EVENT SOURCE
- EventClose2
- EventCreate
- EventCreate2
- EventCreateEventID
- EventDelete2
- EventElementData
- EventGetClass
- EventGetEvent
- EventIdToString
- EventOpen
- EventParam
- EventParam2
- EventPost
- EventPost2
- EventPostByEvent
- EventPostByEvent2
- EventQueueAndElement
- EventRegisterCallback
- EventRegisterCallback2
- EventRegisterCallbackFunction
- EventRegisterCallbackFunction2
- EventRegisteredCallbacks
- EventUnregisterCallback
- EventUnregisterCallbackFunction
- EventUnregisterCallbackFunction2
- EVTPARAM Application
- EVTPARAM_BeforeCheckFirmware
- EVTPARAM_CIFX_GetFirmware
- EVTPARAM CIFX LoadFirmware
- EVTPARAM CIFX PacketParam
- EVTPARAM CIFX xChannelClose
- EVTPARAM CIFX xChannelOpen
- EVTPARAM CmpApp
- EVTPARAM_CmpAppAllBootAppsLoaded
- EVTPARAM_CmpAppComm
- EVTPARAM CmpAppCommCycle
- EVTPARAM CmpAppConfig
- EVTPARAM CmpAppDeny
- EVTPARAM CmpAppDenyDelete
- EVTPARAM CmpAppDenyLoadBootproject
- EVTPARAM_CmpAppDenyStart
- EVTPARAM_CmpAppDenyStop
- EVTPARAM CmpAppException
- EVTPARAM CmpAppExit
- EVTPARAM CmpAppOEMServiceTag
- EVTPARAM CmpAppOperatingStateChanged
- EVTPARAM CmpAppPrepareLoadBootproject
- EVTPARAM_CmpAppRegisterBootproject
- EVTPARAM_CmpAppReset
- EVTPARAM CmpAppResetAllApplications
- EVTPARAM CmpAppRetainBackupState
- EVTPARAM CmpAppSourceDownload
- EVTPARAM CmpAppStateChanged
- EVTPARAM CmpAppStop
- EVTPARAM CmpCodeMeterLicenseActivated
- EVTPARAM_CmplecTask
- EVTPARAM CmplecTask2
- EVTPARAM CmploMgr
- EVTPARAM CmpLogAdd
- EVTPARAM CmpMgr DisableOperation
- EVTPARAM CmpMgr LicenseRequest
- EVTPARAM CmpMgr PrepareExitCommProcessing
- EVTPARAM_CmpMgr_Shutdown
- EVTPARAM_CmpOPCUAServerSessionsChanged
- EVTPARAM CmpSrv
- EVTPARAM CmpTraceMgr Packet
- EVTPARAM CmpTraceMgr Record
- EVTPARAM CmpUserMgrDatabaseChanged
- EVTPARAM DownloadProgress
- EVTPARAM PacketConfirmation
- EVTPARAM PacketIndication
- EVTPARAM PacketUnhandled
- EVTPARAM SysEthernet
- EVTPARAM UploadProgress
- Expert
- ExpressionResult
- ExpSubmodule
- EXTRACT
- F TRIG
- FactoryBase
- FaultStatus
- FbChangeVisu
- FbCloseDialog
- FBFileTransfer
- fblEC61850 Subs ASN1 CheckData
- fblEC61850 Subs ASN1 Decoder
- fblEC61850 Subs ASN1 Decoder CheckDataNum
- fbIEC61850 Subs ASN1 Decoding Data
- fbIEC61850_Subscriber
- FblterateClients
- FbOpenDialog
- FbOpenDialogExtended
- FctGetAngle
- FctGetAngleFromProp
- FctGetNextVisu
- FctGetPreviousVisu
- FctIncreaseElemRectForLine
- FctPointIntersectsRectangle
- FD CLR
- FILE DIR ENTRY
- FILENAME
- FillNodeInfolnt
- FilterErrorCode
- FIND
- FindBlock
- FindByte
- FlatClass
- FlatCreateH
- FlatCreateP
- FlatDelete
- FlatDisable
- FlatEnable
- FlatGetSize
- FlatRead
- FlatTest
- FlatUpdate
- FLOAT
- FLOAT_TO_LREAL
- FLOAT_TO_REAL
- Floor
- FloorF
- Flush
- FMI
- fmod
- FORMAT MODE
- FormatDateTime
- FormatTimestamp
- FormatTimestamp2
- FormatTypedValue
- FrameManager
- FramePaintState
- FrameRegistrationData
- FreeMessage
- FreeStackAllocatedMemory
- FREQ MEASURE
- FromBACnetBitString
- FromBACnetBoolean
- FromBACnetDate
- FromBACnetDateRange
- FromBACnetDateTime
- FromBACnetDevObjPropReference
- FromBACnetSetpointReference
- FromBACnetString
- FromBACnetTime
- FromBACnetTimeStamp
- FSLState
- funIEC61850 GetReportHeaderLen
- funIEC61850 MMSTYPE TO STRING
- funIEC61850 Subs Bits SwapRight
- funIEC61850 Subs InitDatapoint
- funIEC61850 SubsCheckDataNum
- GEN
- GEN MODE
- Generic Service
- Get Attribute List
- Get Attribute Single
- Get Attributes All
- GET_CANOPEN_KERNEL_STATE
- GET_LOCAL NODEID
- GET STATE
- GetAlarmIDName
- GetAllEtherCATCouplers
- GetAttribute
- GetBACnetDataTypeSize
- GetBACnetPropertyDataType
- GetBaudrate
- GETBIT
- GetBitStringFromContents
- GetBitValue
- GetBooleanProperty
- GetBoolFromContents
- GetBufferSize
- GetBusAlarm
- GetBusError
- GetBusload
- GetBusScan
- GetBusState
- GetCallback
- GetChar
- GetCiAState
- GetClass
- GetClassInfo
- GetCompany
- GetConfigType
- GetControllerNode
- GetDatabaseFileSize
- GetDatabaseFileSize2
- GetDateAndTime
- GetDateFromContents
- GetDateRangeFromContents
- GetDateTime
- GetDateTimeFromContents
- GetDayOfWeek
- GetDeviceError
- GetDeviceNameString
- GetDeviceNameWString
- GetDeviceNode
- GetDevObjPropReferenceFromContents
- GetDiagnosis
- GetElapsedTimeInNSec
- GetElapsedTimeInUSec
- GetHandleOfCallback
- GetID
- GetIDeviceInstByloAddr
- GetInfo
- GETIO PART
- GetIPAddress
- GetLatchVarColumnID
- GetLibVersion
- GetLibVersionNumber
- GetLINTValue
- GetLINTValue2
- GetLINTValue3
- GetLocalDateTime
- GetLostCounter
- GetLrealFromContents
- GetLRealSpecialVal
- GetMessageDataPointer
- GetMessageld
- GetMessageLength
- GetMsgCount
- GetNetld
- GetNextNode
- GetNodeDepth
- GetNumberActiveCallbacks
- GetNumberProperty
- GetObjectIDFromContents
- GetParent
- GetPos
- GetProperty
- GetRealFromContents
- GetRealSpecialVal
- GetReceiveCounter
- GetReceiveErrorCounter
- GetReceivePoolSize
- GetReceiveQueueLength
- GetRoot
- GetSetpointReferenceFromContents
- GetSignedFromContents
- GetSize
- GetSpecificDeviceError
- GetSubmoduleDiagnosis
- GetSupplierVersion
- GetText
- GetTextListInfo
- GetTextProperty
- GetTextProperty2
- GetTextW
- GetTick
- GetTime
- GetTimeFromContents
- GetTimeStamp
- GetTimeStampsDifference
- GetTimeZoneInformation
- GetTitle
- GetTransmitCounter
- GetTransmitErrorCounter
- GetTransmitPoolSize
- GetTransmitQueueLength
- GetTypeNodeOptions
- GetUnsignedFromContents
- GetVersion
- GetVersionProperty
- GetWStringFromContents
- GloballmagePool
- GlobalTextList
- GRAY TO BYTE
- GRAY_TO_DWORD
- GRAY TO WORD
- GUARDING TYPE
- Guid
- HaModAIO
- HaModCallbackStop
- HaModControl
- HaModCtd
- HaModCtu
- HaModCtud
- HaModDataSync
- HaModDerivative
- HaModDiag
- HaModDIO
- HaModEthFrame
- HaModEthFrameHeader
- HaModIntegral
- HaModPid
- HaModPidFixCycle
- HaModRampInt
- HaModRampReal
- HaModStatus
- HaModStatusLifecom2
- HaModStatusPIc
- HaModTof
- HaModTon
- HaModVisuData
- HaModVisuEthinfo
- HANDLE
- HANDLE TO DWORD
- HANDLE TO LWORD
- HANDLE TO WORD
- HandleChannelError
- HandleReply
- HandleStateVariables
- HandleStore
- HasAlarmStorageRecordLimit
- HashCodeFromString
- HashCodeFromWString
- HashTable
- HashTableFactory
- HasNullOrEmptySecurityPolicy
- HEADER TAG
- HeapInspectionInfo
- HEXinASCII TO BYTE
- HexStrToLReal
- HexStrToReal
- HighByte
- HighWord
- HistoricalActiveAlarmRowID
- history
- HOSTNAME
- HOUR
- HvacAirDensity
- HvacAirEnthalpy
- HvacDewConcentration
- HvacDewRelHumidity
- HvacDewTemp
- HvacOnTime
- HvacPumpEnergyCalc
- HvacRuntimeCounter
- HvacVaporSaturationPressure
- HYSTERESIS
- Hysteresis DINT
- Hysteresis LREAL
- IAbbAsyncJob
- IAbbETrig2
- IAbbETrig3
- IAbbLCon2
- IAbbLCon3
- IAbbLConC2
- IAbbLConC3
- IAbortable
- IAC500Device
- IAC500DeviceCM5610 2RS
- IAC500DeviceCM5640 2ETH
- IAC500DeviceCM56xCoupler
- IAC500DeviceCoupler
- IAC500DeviceNetxCoupler
- IAC500DeviceOptionBoard
- IAC500DeviceSerialProtocol
- IAC500DeviceTA5130KNXPB
- IAC500DeviceTA5131RTC
- IAC500DeviceTA5141RS232I
- IAC500DeviceTA5142RS485
- IAC500Diag
- IAC500DiagGet
- IAC500NetworkInterface
- IAC500OnboardEtherCATMaster
- IActionController
- IActionController2
- IActionProvider
- IAddressResolver
- IAddrInfoLeafTreeNode
- IAlarm
- IAlarm 2
- IAlarm3
- IAlarm4
- IAlarm5
- IAlarmAdditionalLatchVariables
- IAlarmAdditionalLatchVariables2
- IAlarmClass
- IAlarmConfiguration7
- IAlarmConfiguration8
- IAlarmGroup
- IAlarmGroup3
- IAlarmHandler
- IAlarmHandler2
- IAlarmHandler3
- IAlarmHandler4
- IAlarmHandler5
- IAlarmHandlerRemoteMonitor
- IAlarmManagerClient
- IAlarmManagerClient2
- IAlarmMetaObjectStub5
- IAlarmNotifiable
- IAlarmRemote
- IAlarmStateChangedEventListener
- IAlarmStateChangedListener
- IAlarmStateChangedListener2
- IAlarmStorageListener
- IAlarmStorageReaderConsumer
- IAlarmStorageReaderConsumer2
- IApplicationRectangleProvider
- IARPCallback
- IARPEthernetClient
- IArrayNotifiable
- IAsyncProperty
- IBACnetClient
- IBACnetEventConsumer
- IBACnetObjectBase
- IBACnetPersistence
- IBACnetPropertyConfiguration
- IBACnetServer
- IBACnetServerPlugin
- IBACnetServerPluginCallback
- IBACnetServerPluginHook
- IBACnetStaticObjectBase
- IBase
- IBaseTreeNode
- iBasicEthercat
- iBasicKernel
- IBehaviourModel
- IBoolElement
- IBranchTreeNode
- IBuffer
- IBufferPool
- IBufferPoolFactoryArgs
- IBus
- ICallback
- ICallOnDialogBlocks
- ICallOnVisuBlocks
- ICANopenEventHandler
- ICanOpenStack
- ICascadedDisposalProvider
- ICDSV3Request
- ICDSV3RequestBuilder
- ICDSV3RequestCallback
- ICDSV3RequestHandling
- ICDSV3Response
- ICDSV3ServiceWriter
- ICertificateVerifier
- ICleanupActionProvider
- IClient
- IClientObjectInfo
- IClippingLayer
- ICmpEventCallback
- ICmploDrv
- ICmploDrvBusControl
- ICmploDrvBusControl2
- ICmploDrvPbSlaveActivation
- ICmploDrvProfibus
- ICmploDrvProfibusConfig
- ICollection
- ICompactTextListInfo2
- ICompleteSurroundingRectInfo
- IConfigurationProvider
- IConfigurationProvider2
- IConnection
- IContainerPaintSelf
- IContainsValue
- ICredentials
- ICredentialsProvider
- ICredentialsUserPassword
- ICursor
- ICursor2
- ICursor3
- ICursorAsync
- ICustomAlarmToOpcUaMapping
- ICustomEventHandler
- ICustomRequest
- ICustomRequestCallback
- ICustomRequestHandling
- ICustomResponse
- ICyclicActionProvider
- ID
- ID TO ADDR
- IData
- IDataGroup
- IDataltem
- IDataltem2
- IDataltemBase
- IDataltemCompound
- IDataltemInfo
- IDataltemList
- IDataltemListExpanding
- IDataltemListInternal
- IDataltemListPersistant
- IDataltemProvider
- IDataltemProvider2
- IDataltemProviderSupportsPartialLists
- IDataltemValueAssigner
- IDataServer
- IDataServer2
- IDataServer3
- IDataServer4
- IDataServer5
- IDataServer6
- IDataServerActionRecord
- IDataServerFrameManager
- IDatasource
- IDatasource2
- IDatasource3
- IDataSourceDriver
- IDataSourceDriver2
- IDataSourceDriver3
- IDataSourceDriver4
- IDataSourceDriver5
- IDatasourcesActionRecord
- IDatasourcesActionRecordInternal
- IDatasourcesFrameManager
- IDatasourcesMgr
- IDatasourcesMgr2
- IDatasourcesMgr3
- IDatasourcesMgr4
- IDatasourcesResourceEntryAllocator
- IDatasourceTraceAddressInfoProvider
- IDateTimeLanguageTextTarget
- IDateTimeProvider
- IDENT
- IDENT TO DWORD
- IDENT TO WORD
- IDevice
- IDevice2
- IDeviceCM579EtherCAT
- IDeviceCM579Profinet
- IDeviceCM582Profibus
- IDeviceCM589Profinet
- IDeviceCM592Profibus
- IDeviceCM598Can
- IDeviceSM560
- IDialogCloseListener
- IDialogCloseListenerWithTag
- IDialogManager10
- IDialogManager2
- IDialogManager3
- IDialogManager4
- IDialogManager5
- IDialogManager6
- IDialogManager7
- IDialogManager8
- IDialogManager9
- IDintElement
- IDintSet
- IDirectoryFileElement
- IDisposable
- IDoubleLinkedList
- IDrawingInterface3
- IDrawSequentially
- IDriver
- IDriverCDSV3
- IDriverCredentialsHandling
- IDriverCredentialsHandlingEveryLogin
- IDriverCustomRequest
- IDriverLargeRequest
- IDriverRequest
- IDriverRequestFactory
- IDriverRequestFactoryCustom
- IDriverRequestFactoryLargeRequests
- IDriverRequestFactoryReusable
- IDriverRequestReusable
- IDriverSpecificData
- IDriverSpecifiedSettings
- IDriverStateMachine
- IDriverStateMachine2
- IDriverStateMachineNotifyBeforeShutdown
- IDriverStateMachineShutdownTimeout
- IDriverTraceAddressInfoProvider
- IEC60870 5104 Connection
- IEC60870 BACKGROUND SCAN
- IEC60870_DISABLE
- IEC60870_DoubleCommand
- IEC60870 DoublePointInformation
- IEC60870 GET ADDRESS
- IEC60870 IntegratedTotal
- IEC60870 MeasuredValue
- IEC60870 REC C DC
- IEC60870 REC C SC
- IEC60870 REC C SE
- IEC60870 REC C TS NA 1
- IEC60870 REC M DP
- IEC60870 REC M IT
- IEC60870 REC_M_ME
- IEC60870 REC M ME 1
- IEC60870 REC M SP
- IEC60870 REC P ME
- IEC60870 SEND C CI NA 1
- IEC60870 SEND C CI NA 12
- IEC60870 SEND C CS NA 1
- IEC60870_SEND_C_CS_NA_1_2
- IEC60870_SEND_C_DC
- IEC60870 SEND C IC NA 1
- IEC60870 SEND C IC NA 12
- IEC60870 SEND C RD NA 1
- IEC60870 SEND C RP NA 1
- IEC60870 SEND C RP NA 12
- IEC60870_SEND_C_SC
- IEC60870_SEND_C_SE
- IEC60870 SEND C TS NA 1 ACT
- IEC60870 SEND C TS NA 1 ACTCON
- IEC60870 SEND DISABLE
- IEC60870 SEND M DP
- IEC60870 SEND M DP ET
- IEC60870 SEND M EI NA_1
- IEC60870 SEND M IT
- IEC60870 SEND M IT 1
- IEC60870 SEND M IT 16
- IEC60870 SEND M IT 16 ET
- IEC60870 SEND M IT 1 ET
- IEC60870 SEND M ME
- IEC60870_SEND_M_ME_1
- IEC60870 SEND M ME 16
- IEC60870 SEND M ME 16 ET
- IEC60870 SEND M ME 1 ET
- IEC60870 SEND M SP
- IEC60870 SEND M SP 16
- IEC60870 SEND M SP 16 ET
- IEC60870 SEND M SP 1 ET
- IEC60870 SEND P_ME
- IEC60870 SetPoint
- IEC60870 SingleCommand
- IEC60870 SinglePointInformation
- IEC60870 STATE
- IEC60870 TIME
- IEC60870Commands
- IEC60870Disable
- IEC60870DisableSend
- IEC60870GetConfigAddress
- IEC60870GetConfigValues
- IEC60870GetConnectionStatistics
- IEC60870GetPinData
- IEC60870GetStatesOfPinParam
- IEC60870GetStatesOfPins
- IEC60870GetTestInformation
- IEC60870SendCommand
- IEC60870SendPinData
- IEC60870SetParameterValues
- IEC60870SetPinData
- IEC60870StartScan
- IEC61850 ArrayBits SwapLeft
- IEC61850 ASN1 Decoder
- IEC61850 ASN1 DECODING
- IEC61850 ASN1 Decoding Data
- IEC61850 ASN1 EncodingBlock
- IEC61850_ASN1_EncodingSize
- IEC61850_ASN1_EncodingSpecific
- IEC61850 ASN1 EncodingStruct
- IEC61850 ASN1 GetNextTag
- IEC61850 ASN1 NewDecoder
- IEC61850 ByteBits SwapLeft
- IEC61850 ByteBits SwapRight
- IEC61850_CDC_ACD
- IEC61850_CDC_ACT
- IEC61850 CDC ALM
- IEC61850 CDC APC
- IEC61850 CDC ASG
- IEC61850 CDC ASS
- IEC61850 CDC BCR
- IEC61850 CDC_BRCB
- IEC61850 CDC BSC
- IEC61850 CDC CMD
- IEC61850 CDC CMV
- IEC61850 CDC CSD
- IEC61850 CDC CTE
- IEC61850 CDC CURVE
- IEC61850_CDC_DEL
- IEC61850 CDC DPC
- IEC61850 CDC DPL
- IEC61850 CDC DPS
- IEC61850 CDC GoCB
- IEC61850 CDC HDEL
- IEC61850 CDC HMV
- IEC61850 CDC HWYE
- IEC61850 CDC INC
- IEC61850 CDC ING
- IEC61850 CDC INS
- IEC61850 CDC ISC
- IEC61850 CDC LPL
- IEC61850 CDC MV
- IEC61850 CDC ORG
- IEC61850_CDC_SAV
- IEC61850_CDC_SEC
- IEC61850 CDC SEQ
- IEC61850 CDC SPC
- IEC61850 CDC SPG
- IEC61850 CDC SPS
- IEC61850 CDC SPV
- IEC61850 CDC STV
- IEC61850 CDC TMS
- IEC61850 CDC URCB
- IEC61850 CDC WDPL
- IEC61850 CDC WYE
- IEC61850 Check HexString
- IEC61850 CheckBufferlx
- IEC61850 CheckByteOrder
- IEC61850 CheckClients
- IEC61850 CheckDataPoint
- IEC61850 CheckDoubleDP
- IEC61850 CheckEntryID
- IEC61850_CheckEnumRange
- IEC61850_CheckTrgOp
- IEC61850 CLIENT ACCEPT
- IEC61850 ClientConnectionFB
- IEC61850 CONCAT3
- IEC61850 CONCAT4
- IEC61850 CONCAT5
- IEC61850_CONCAT6
- IEC61850_CpyAndSwap
- IEC61850 CreateBasicNames
- IEC61850 DatasetFB
- IEC61850 DateTime
- IEC61850 DecodeNull
- IEC61850 DeleteDataSet
- IEC61850 DWORD TO HEXSTRING
- IEC61850_Encoding_Array_Count
- IEC61850 Encoding Array Struct
- IEC61850 Encoding Component
- IEC61850 Encoding Component Struct
- IEC61850 Encoding ComponentSingle
- IEC61850 Encoding DirectoryNames
- IEC61850_Encoding_ListOfData
- IEC61850_Encoding_ListOfData_Struct
- IEC61850 Encoding ListOfVariable
- IEC61850 Encoding Value
- IEC61850 ENUM ASN1 TAGS
- IEC61850 ENUM ATTR NAMES
- IEC61850 ENUM DA ALM STATE
- IEC61850 ENUM DA_ANGID
- IEC61850_ENUM DA_ANGIDCMV
- IEC61850 ENUM DA ANGLEREFERENCEKIND
- IEC61850 ENUM DA ASS STVAL
- IEC61850 ENUM DA BEH
- IEC61850 ENUM DA CBOPCAP
- IEC61850 ENUM DA CMDQUAL
- IEC61850 ENUM DA CONTROLOUTPUTKIND
- IEC61850 ENUM DA_CTE_HISRS
- IEC61850_ENUM_DA_CTE_RSPER
- IEC61850 ENUM DA CTLMODELKIND
- IEC61850 ENUM DA CTLMODELS
- IEC61850 ENUM DA CURVECHARKIND
- IEC61850 ENUM DA DAWEEKDAYKIND
- IEC61850 ENUM DA DBPOS
- IEC61850 ENUM DA DIR
- IEC61850 ENUM DA DIRMOD
- IEC61850 ENUM DA ENUMERATED
- IEC61850 ENUM DA FAILMOD
- IEC61850 ENUM DA FANCTL
- IEC61850 ENUM DA_FAULTDIRECTIONKIND
- IEC61850 ENUM DA GNST
- IEC61850 ENUM DA HEALTH
- IEC61850 ENUM DA HVID
- IEC61850 ENUM DA HVREFERENCEKIND
- IEC61850 ENUM DA LEVMOD
- IEC61850 ENUM DA LIVDEAMOD
- IEC61850_ENUM_DA_MOD
- IEC61850_ENUM DA MONTHKIND
- IEC61850 ENUM DA MULTIPLIER
- IEC61850 ENUM DA MULTIPLIERKIND
- IEC61850 ENUM DA OCCURRENCEKIND
- IEC61850 ENUM DA OPMOD
- IEC61850 ENUM DA ORCAT
- IEC61850_ENUM_DA_ORIGINATORCATEGORYKIND
- IEC61850_ENUM_DA_PERIODKIND
- IEC61850 ENUM DA PHASEANGLEREFERENCEKIND
- IEC61850 ENUM DA PHASEFAULTDIRECTIONKIND
- IEC61850 ENUM DA PHASEREFERENCEKIND
- IEC61850 ENUM DA PHSID
- IEC61850 ENUM DA POLQTY
- IEC61850 ENUM DA POWCAP
- IEC61850 ENUM DA_RANGE
- IEC61850 ENUM DA RANGEKIND
- IEC61850 ENUM DA RETRMOD
- IEC61850 ENUM DA RSTMOD
- IEC61850 ENUM DA RVAMOD
- IEC61850 ENUM DA SBOCLASSES
- IEC61850_ENUM_DA_SBOCLASSKIND
- IEC61850 ENUM DA SCHTYP
- IEC61850 ENUM DA SEQT
- IEC61850 ENUM DA SEQUENCEKIND
- IEC61850 ENUM DA SETCHARACT
- IEC61850 ENUM DA SEV
- IEC61850 ENUM DA SEVERITYKIND
- IEC61850 ENUM DA SHOPCAP
- IEC61850 ENUM DA SIUNIT
- IEC61850 ENUM DA SIUNITKIND
- IEC61850 ENUM DA SPV CHAPERRS
- IEC61850 ENUM DA SPV SPACS
- IEC61850 ENUM DA SWOPCAP
- IEC61850 ENUM DA SWTYP
- IEC61850 ENUM DA TCMD
- IEC61850 ENUM DA_TMS_HISRS
- IEC61850_ENUM_DA_TMS_RSPER
- IEC61850 ENUM DA TRGMOD
- IEC61850 ENUM DA TRMOD
- IEC61850 ENUM DA TYPRSCRV
- IEC61850 ENUM DA UNBLKMOD
- IEC61850 ENUM DA WEIMOD
- IEC61850 ENUM ELEMENTTYP
- IEC61850 ENUM FC
- IEC61850 ENUM MMS CONFIRMED REQ PDU
- IEC61850 ENUM MMS CONFIRMED RESP PDU
- IEC61850 ENUM MMS DataType
- IEC61850_ENUM MMS_OBJECTCLASS
- IEC61850 ENUM MMS PDU
- IEC61850 ENUM QUALITY
- IEC61850 ENUM SERVICES
- IEC61850 ENUM TRGOPT
- IEC61850 EthernetAdapter
- IEC61850 GetDatapoint
- IEC61850 GetDataPointLen
- IEC61850_GetDatapointRef
- IEC61850 GetDefinition
- IEC61850 GetDirectory
- IEC61850 GetDirectory All
- IEC61850 GetFC
- IEC61850 GetReportLen
- IEC61850_GetURCBDataLen
- IEC61850_GetValue
- IEC61850 GetValues All
- IEC61850 Goose ASN1 Decoder
- IEC61850 GOOSE MReq
- IEC61850 GooseDecodeData
- IEC61850 HEXSTRING TO DWORD
- IEC61850 HistDataBuffer_In
- IEC61850 HistDataBufferFB
- IEC61850 Init BReportBlock
- IEC61850 Init DataPoints
- IEC61850 Init GoCB
- IEC61850 Init UBReportBlock
- IEC61850 InitDSLastValPtr
- IEC61850 INT_TO STRING
- IEC61850_MMS_Data_InterpreterFB
- IEC61850 MMS ErrorPDU
- IEC61850 MMS InterpreterFB
- IEC61850 MMSGetBlockLen
- IEC61850 ReadDWord
- IEC61850 ReadISOHeader
- IEC61850 ReadString
- IEC61850 ReadWord
- IEC61850 SetDatasetVal
- IEC61850 SetDSError
- IEC61850 SetISOEntry
- IEC61850 SetISOLen
- IEC61850 SetReportValue
- IEC61850 SetStructIndex
- IEC61850 SetTrgOpt
- IEC61850_SetValue
- IEC61850 SimpleClock
- IEC61850 STR TO BYTE
- IEC61850 String Split
- IEC61850 SWAP 2 BYTE
- IEC61850 SWAP 3 BYTE
- IEC61850 SWAP 4 BYTE
- IEC61850 SysMemCpy
- IEC61850 TimeStampR
- IEC61850 Version
- IEC61850 WordBits SwapLeft
- IEC61850_WordBits_SwapRight
- IEC61850ServerFB
- IEC BACNET ERROR TO STRING
- IEC CYCLE STRUCT
- IEC STATE
- lecOpcUaMapping
- lecTaskCreate
- lecTaskCreate2
- lecTaskDelete2
- lecTaskDelete3
- lecTaskDisableScheduling
- lecTaskDisableWatchdog
- lecTaskDisableWatchdog2
- lecTaskEnableScheduling
- lecTaskEnableWatchdog
- lecTaskEnableWatchdog2
- lecTaskGetCurrent
- lecTaskGetDesc
- lecTaskGetFirst
- lecTaskGetInfo3
- lecTaskGetNext
- lecTaskGetProfiling
- lecTaskGetWatchdogHitCount
- lecTaskReload
- lecTaskResetStatistics
- lecVarAccBrowseCallback
- lecVarAccBrowseDirection
- lecVarAccBrowseDown2
- lecVarAccBrowseGetNext2
- lecVarAccBrowseRecursive
- lecVarAccBrowseUp2
- lecVarAccess
- lecVarAccessUalnformationModeIMetaData
- lecVarAccExitVarInfo
- lecVarAccGetFirstInterface
- lecVarAccGetFirstInterface2
- lecVarAccGetNextInterface
- lecVarAccGetNextInterface2
- lecVarAccGetNode4
- lecVarAccGetNodeFullPath4
- lecVarAccGetNodeName4
- lecVarAccGetSymbolSetMask
- lecVarAccInitVarInfo
- lecVarAccInitVarInfo2
- lecVarAccInvalidateNode
- lecVarAccNodeInfoAddBrowseInfo
- lecVarAccNodeInfoAddReference
- lecVarAccNodeInfoGetBrowseInfo
- lecVarAccNodeInfoGetReference
- lecVarAccNodeInfoRemoveBrowseInfo
- lecVarAccNodeInfoRemoveReference
- lecVarAccRegisterInstance
- lecVarAccRegisterInstance2
- lecVarAccRegisterInstance3
- lecVarAccRegisterInstanceBase
- lecVarAccRegisterInstanceBase2
- lecVarAccSetSymbolconfigCrc
- lecVarAccSymbolSetDescription
- lecVarAccUnregisterInstance
- lecVarAccUpdateSymbolSets
- IEdgeTriggered
- IElement
- IEthernet
- IETrig
- IETrigA
- IETrigATI
- IETrigATITo
- IETrigATo
- IETrigTI
- IETrigTITo
- IETrigTo
- IExitActionProvider
- IExpandSubNodeAdapterSingleRelease
- IExternalUserDatabaseProvider
- IExternalUserDatabaseProvider2
- IFactory
- IFBCommand
- IFDBConfig
- IFileListProvider
- IFrame
- IFrameElement2
- IFrameElement3
- IFrameManager
- IFrameManager2
- IFrameManagerBase
- IGeneralCommand
- IGeneric
- IGestureEventHandler
- IGestureEventHandler2
- IGestureEventHandler3
- IGlobalClientManagerListener
- IGridProvider
- IHasContinuousBehaviour
- iIEC61850 LogicalDevice
- IlecVarAccess
- IlecVarAccess10
- IlecVarAccess11
- IlecVarAccess12
- IlecVarAccess13
- IlecVarAccess14
- IlecVarAccess15
- IlecVarAccess2
- IlecVarAccess3
- IlecVarAccess4
- IlecVarAccess5
- IlecVarAccess6
- IlecVarAccess7
- IlecVarAccess8
- IlecVarAccess9
- IlecVarAccessOpcUaMetaData
- IInputOnElementEventHandler
- IInputRectangle
- IInputRectangleMgr
- IInputRectangleProvider
- IInstance
- IInstancePathBuildingNode
- IIntElement
- IloDrvEIPAcylicServices
- IIPAddress
- IIPAddressSet
- IIPv4Address
- Ilterator
- IKeyEventHandler
- ILayeredVisualElement
- ILayerManager
- ILCon
- ILConC
- ILConTI
- ILConTIC
- ILContiTo
- ILConTo
- ILeafTreeNode
- ILevelControlled
- ILinkedListlterator
- ILintElement
- IList
- IList2
- IListlterator
- ILocalAssigner
- ILocalizedDateTimeNames
- ILogger
- ILRealToStringFormatter
- ImagePool
- ImagePool 3DPath
- ImagePoolDialogs
- IMap
- IMap2
- IMemberIndex
- IMouseEventHandler
- IMultitouchElement
- INADDR
- INamedTreeNode
- INetworkInterface
- INFO
- InfoValues
- InitializeBACnetBitString
- InitializeBACnetBoolean
- InitializeBACnetDate
- InitializeBACnetDateRange
- InitializeBACnetDateTime
- InitializeBACnetDateTimeUnspecified
- InitializeBACnetDevObjPropReference
- InitializeBACnetSetpointReference
- InitializeBACnetString
- InitializeBACnetTime
- InitializeBACnetTimeStamp
- InitializeEmptyPropertyInstance
- InitializePropertyInstance
- INode
- INode TO IBus
- INode TO IDevice
- INode TO IDevice2
- INode_TO_IStack
- INodeName
- INotificationClient
- InputDataSave
- INSERT
- Inspect Heap
- InstanceBase
- InstanceData
- InstancePathBuildingBranchNode
- InstancePathBuildingNode
- InstancePathBuildingNodeType
- InstancePathNodeFinder
- INT64
- INT64 TO DT
- INT64-TO ISO8601
- INT64-TO LOCALTIME
- INT64 TO LTIME
- INT64 TO REAL8
- INT64 TO TIME
- INT64 TO UTC
- INT TO BCD
- INT TO SIGNED
- INTEGRAL
- Integral
- IntElement
- IntElementFactory
- InterfaceVersion
- InternalState
- INullElement
- IO PROD ENTRY READ
- IO SYSTEM TYPE
- IOBus Download
- IOBus GetBusInfo
- IOBus GetBusStatistcis
- IOBus GetDownloadState
- IOBus GetHotplugOK
- IOBus_GetIODriverVersion
- IOBus_GetModState
- IOBus GetModuleInfo
- IOBus GetModuleLinkStatistics
- IOBus GetModuleStatistics
- IOBus GetModuleVersion
- IOBus GetPlugged
- IOBus GetProductionData
- IOBus GetRun
- IOBUS INFO
- IOBUS LINKSTATISTICS
- IOBUS MOD STATE
- IOBUS MODUL_STATE
- IOBUS MODULINFO
- IOBUS PARA STATE
- IOBUS PRODDATA
- IOBUS STATISTICS
- IOBus SwitchLinkStatistics
- IOBUS TU STATE
- IOBUS_VERSIONINFO
- loCnt
- IoConfigChanneIMap
- loConfigConnector
- IoConfigConnectorMap
- IoConfigParameter
- IoConfigTaskMap
- loDiag
- loDriverVersion
- loDrvAnalogBase
- loDrvBase
- IoDrvCIFX
- IoDrvCIFXEthernetIP
- IoDrvCIFXEthernetIP Diag
- loDrvCIFXProfinetDevice
- IoDrvCIFXProfinetDeviceDiag
- IoDrvCM579EtherCAT
- IoDrvCM579EtherCATDiag
- IoDrvCM579Profinet
- IoDrvCM579ProfinetDiag
- IoDrvCM582Profibus
- IoDrvCM582ProfibusDiag
- IoDrvCM589Profinet
- IoDrvCM589ProfinetDiag
- loDrvCM592Profibus
- IoDrvCM592ProfibusDiag
- IoDrvCM598
- IoDrvCM598Diag
- IoDrvCpuModuleDiag
- IoDrvDigitalOptionBoardBase
- IoDrvEtherCAT
- IoDrvEthercat Diag
- IoDrvEthernet
- IoDrvEthernetAC500
- loDrvEthernetAC500Diag
- loDrvEthernetDiag
- IoDrvEtherNetIP
- loDrvEtherNetIP_diag
- IoDrvEtherNetIPAdapter
- IoDrvEtherNetIPAdapter Diag
- loDrvinfo
- IoDrvloBusModuleDiag
- IoDrvJ1939Diag
- loDrvKNX
- IoDrvKNXDiag
- IoDrvOnboardEtherCAT
- IoDrvOnboardEtherCAT Diag
- loDrvOnboardIO
- IoDrvOnboardIODiag
- loDrvS500ModuleDiag
- loDrvSM560
- loDrvSM560Diag
- loDrvTA5101
- IoDrvTA5101Diag
- IoDrvTA5105
- IoDrvTA5105Diag
- IoDrvTA5110
- IoDrvTA5110Diag
- IoDrvTA5120
- IoDrvTA5120Diag
- loDrvTA5122
- IoDrvTA5122Diag
- IoDrvTA5123
- IoDrvTA5123Diag
- IoDrvTA5126
- IoDrvTA5126Diag
- IoDrvTA5128
- IoDrvTA5128Diag
- IoInformation
- IOL_AdditionalCode
- IOL_AdjustableSwitchingSensor
- IOL AdSS Function
- IOL AdSS Status
- IOL AdSS TeachFunction
- IOL AdSS TeachMode
- IOL CALL
- IOL_DataStorage
- IOL_DiagEntry
- IOL Error
- IOL ErrorCode
- IOL Event
- IOL EventCode
- IOL EventCode Device
- IOL EventCode Port
- IOL_EventQualifier
- IOL EventQualifier Instance
- IOL EventQualifier Mode
- IOL EventQualifier Source
- IOL EventQualifier Type
- IOL FieldbusStatus
- IOL GetEvent ChanneIDiagnosis
- IOL_GetEvent_UDINT
- IOL_IdentificationAndDiagnosis
- IOL IdentificationAndDiagnosis Function
- IOL IdentificationObjects
- IOL Index
- IOL IOLM Info
- IOL IOLM InfoRecord
- IOL IQ Behavior
- IOL Masterldent
- IOL Masterldent Features 1
- IOL MasterType
- IOL MeasurementDataChannel
- IOL_PN_PortControl
- IOL PortConfigList
- IOL PortConfiguration
- IOL PortConfigurationRecord
- IOL PortError
- IOL PortMode
- IOL PortQualityInfo
- IOL_PortStatus
- IOL_PortStatusInfo
- IOL PortStatusList
- IOL PortStatusRecord
- IOL PortType
- IOL PQI
- IOL ProfileIdentifier
- IOL_TransmissionRate
- IOL_ValidationBackup
- loMgrConfigGetConnector
- IoMgrConfigGetConnectorByDriver
- IoMgrConfigGetConnectorList
- IoMgrConfigGetDriver
- IoMgrConfigGetFirstChild
- IoMgrConfigGetFirstConnector
- IoMgrConfigGetFirstParameter
- IoMgrConfigGetNextChild
- IoMgrConfigGetNextConnector
- IoMgrConfigGetNextParameter
- IoMgrConfigGetParameter
- IoMgrConfigGetParameterValueByte
- IoMgrConfigGetParameterValueDword
- loMgrConfigGetParameterValuePointer
- IoMgrConfigGetParameterValueWord
- IoMgrConfigResetDiagnosis
- IoMgrConfigSetDiagnosis
- IoMgrCopyInputBE
- IoMgrCopyInputLE
- IoMgrCopyOutputBE
- IoMgrCopyOutputLE
- IoMgrGetBusCycleType
- IoMgrGetConfigApplication
- IoMgrGetDriverProperties
- loMgrGetFirstDriverInstance
- IoMgrGetModuleDiagnosis
- loMgrGetNextDriverInstance
- loMgrldentify
- IoMgrLockEnter
- loMgrLockLeave
- loMgrReadInputs
- loMgrReadParameter
- IoMgrReconfigure
- IoMgrRegisterConfigApplication
- loMgrRegisterInstance2
- IoMgrScanModules
- IoMgrSetDriverProperties
- IoMgrSetDriverProperty
- IoMgrStartBusCycle
- IoMgrStartBusCycle2
- IoMgrUnregisterConfigApplication
- loMgrUnregisterInstance
- IoMgrUpdateConfiguration
- IoMgrUpdateConfiguration2
- IoMgrUpdateMapping
- IoMgrUpdateMapping2
- IoMgrWatchdogTrigger
- loMgrWriteOutputs
- loMgrWriteParameter
- loModuleDiag
- IoModuleHotSwapInfo
- loModuleInfo
- loModuleInfoExt
- IOnlineChangeSafeLinkedListElement
- IOPCUAClientConnectionCallback
- IOPCUAClientDataAccessCallback
- IOPCUAClientDiscoveryCallback
- IOPCUAClientMethodCallback
- IOPCUAClientMonitoredItemCallback
- IOPCUAClientSubscriptionCallback
- IOPCUAClientViewCallback
- IOpcUaDataTypeMetaData
- IOpcUalnstanceMetaData
- IoProdEntryRead
- IOptionalMultitouchElement
- IOptionProvider
- IOxStatus
- IP_ADDR
- IP ADR DWORD TO STRING
- IP ADR STRING TO DWORD
- IP ElementImages
- IP VUM
- IPAADialog
- IPAADialog2
- IPacket
- IPacketPool
- IPacketQueue
- IPADDRESS
- IPAddressSet
- IPaintAfterAll
- IPaintAfterAll2
- IPaintAfterAllOwningElement
- IPaintAfterAllRectangleProvider
- IPaintAfterAllSelection
- IPaintSelectionInElement
- IPARRAY TO INADDR
- IPARRAY TO IPSTRING
- IPARRAY TO UDINT
- IPBSlaveDiag
- IPeer
- IPersistantRecipeListSupportsAdd
- IPoolExtender
- IPParameterValue
- IProvidesBitOffset
- IProvidesConcreteType
- IProvidesContainerSize
- IProvidesDifferentRemoteName
- IProvidesRootInfo
- IProvidesTabOrder
- IProxyMonitor
- IPseudoNode
- IPSTRING_TO_UDINT
- IPStringAndIntElement
- IPStringElement
- iPTOKernel
- IPv4Address
- IQueryInterfaceElement
- IQueue
- IQueue2
- IRecipeCheckOnStart
- IRecipeDefinition2
- IRecipeDefinition3
- IReconfigureProvider
- IRectangleListManager
- IRectangleListManager2
- IRectangleListManager3
- IRectangleListManager4
- IRectangleProvider
- IRequest
- IRequestResult
- IRequiresInitMeasureString
- IResetActionProvider
- IResolveCallbackHandler
- IRoleEtherCAT
- IRoleEtherCATMaster
- IRootAddressNodeWithDatasourceIndex
- IRow
- IRow2
- IRow3
- IRowAsync
- IRowBase
- IRowIdIterator
- IRowPlanchet
- IRowPlanchetAsync
- IRPCCLClient
- IRPCCLClientCallback
- IRPCProvider
- IRPCProviderCallback
- IRtsServiceHandler
- IRtsServiceHandler2
- Is29BitldMessage
- IS MULTICAST GROUP
- IsAcceptedLeafNode
- ISampleActionProvider
- ISavepoint
- ISavepointAsync
- IsBACnetBACnetDateTimeUnspecified
- IsBACnetDateTimeUnspecified
- IsBACnetObjectAMEVCreatable
- IsBACnetPropertyAMEVASBWritable
- IsBroadcast
- IScrollValueProvider
- ISearchCallbacks
- ISegment
- ISegmentPool
- ISelectableInside
- ISelectionManager
- IServer
- IServiceReader
- IServiceWriter
- IsHandleValid
- ISimpleList
- ISimpleTree
- IsInvalidMemoryAddress
- ISizeProvider
- IsLeapYear
- IsLegalUTF8
- IsLibReleased
- IsLRealNaN
- IsLRealNegInfinity
- IsLRealNumber
- IsLRealPosInfinity
- ISO8073 FB
- ISO8327 FB
- ISO8327 ReadHeader
- ISO8601
- ISO8601 TO DT
- ISO8601_TO_LTIME
- ISO8601_TO_TIME
- ISO8650 FB
- ISO8823 FB
- ISOLayer FB
- ISortedList
- ISortedList2
- IsP2P
- ISpecialEventHandler
- IsRealNaN
- IsRealNegInfinity
- IsRealNumber
- IsRealPosInfinity
- IsRTRMessage
- IsSendingActive
- IsSpaceCharacter
- IStack
- IStack2
- IStartActionProvider
- IStorage
- IStorage2
- IStorageAsync
- IsTransmitMessage
- IStream
- IStringElement
- ISupportsReaIDrawing
- ISwipeableFrame
- ITable
- ITable2
- ITable3
- ITable 4
- ITableAsync
- ITaskFinishedCallback
- ItemAccess
- ItemName
- ItemNameType
- ITempShutdownResult
- ItemQuality
- ITextListInfo
- ITextListWrapper
- ITimeElement
- ITimeLimited
- ITimeOutConstraint
- ITimingControlled
- ITimingController
- ITLSContext
- ITraceAddressInfoWriter
- ITransaction
- ITransactionAsync
- ITransformationImpIProvider
- ITree
- ITreeNode
- ITreeWalker
- ITrendRecording2
- ITrendRootPageManager2
- ITrendStorageAccessReadOperator
- ITrendStorageAccessReadOperator2
- ITrendStorageReaderConsumer
- ITrendStorageWriter4
- ITrendStorageWriterListener
- ITSNContext
- ITypedElement
- ITypeDesc
- ITypeDesc2
- ITypeDesc3
- ITypeDesc4
- ITypeDescExecutable
- ITypeDescSubrange
- ITypeDescWithAttributes
- ITypeDescWithBaseType
- ITypeDescWithReferenceType
- ITypedList
- ITypedTree
- IUdintElement
- IUintElement
- IUlintElement
- IUseDataContextSubNodes
- IUserConfiguration
- IUserMgmtEventHandler
- IValueChangedListener
- IVariableInfoCollectsSpecialAddressInfos
- IVariableInfoCollectsTopleveIAddressInfo
- IVariableInformation
- IVariableInformation2
- IVariableInformation3
- IVariableInformation4
- IVariableInformation5
- IVerifyCertCallback
- IVisualElementLayer
- IVisualElementOfflineScaling
- IVisualElementProvidesChildElements
- IVisualElementProvidesSubElements
- IVisualElementWithoutBlobInit
- IVisualisationAccessRights
- IVisualizationClient
- IVisualizationClientFilter
- IVisualizationClientlteration
- IVisualizationClientRaw
- IVisualizationStyle
- IVisualTableColumnSimpleBase
- IVisuGlobalClientManager2
- IVisuGlobalClientManager3
- IVisuManager
- IVisuManager2
- IVisuManager3
- IVisuManagerBase
- IVisuStreamFileNameInfo
- IVisuStreamHandler
- IVisuStreamReader
- IVisuStreamSetFileName
- IVisuStreamWriter
- IVisuUserEventManager
- IVisuUserManagement
- IVisuUserManagement2
- IVisuUserManagement3
- IVisuUserManagement4
- IVisuUserMgmt
- IVisuUserMgmt2
- IVisuUserMgmt3
- IVisuUserMgmt4
- IVisuUserMgmtCyclicCall
- IVisuUserMgmtProvider
- IWORKER
- IWStringElement
- IXYChartAlloc
- IXYChartDataProvider
- IXYChartDataProvider2
- IXYChartDataProvider3
- IXYChartDataProviderAxis
- IXYChartDataProviderCurve
- IXYChartDataProviderCurveCheckPointer
- IXYChartFont
- IXYChartGenericVariable
- IXYChartGeneric Variable2
- IXYChartStringApproxMeasurer
- IXYChartVisuStructLevelLine
- J1939ECUBase
- J1939LocalECU
- J1939LocalECUDiag
- J1939RemoteECU
- J1939RemoteECUDiag
- Jitter Distribution
- JOB STATE
- JobAbort
- JobClass
- JobClose
- JobExecute
- JobGetld
- JobGetParams
- JobGetState
- JobOpen
- JobOpenEx
- JobReset
- JobSetState
- JoinDate Time
- JSON ARR REF
- JSON OBJ REF
- JsonAddArray
- JsonAddBool
- JsonAddIInt
- JsonAddObject
- JsonAddReal
- JsonAddString
- JsonArrayAddArray
- JsonArrayAddBool
- JsonArrayAddlnt
- JsonArrayAddObject
- JsonArrayAddReal
- JsonArrayAddString
- JsonArrayGetArray
- JsonArrayGetBool
- JsonArrayGetInt
- JsonArrayGetObject
- JsonArrayGetReal
- JsonArrayGetString
- JsonArrayRemoveEntry
- JsonCreateArray
- JsonCreateObject
- JsonFreeArray
- JsonFreeObject
- JsonGetArray
- JsonGetBool
- JsonGetInt
- JsonGetObject
- JsonGetReal
- JsonGetString
- JsonParseArrayFromString
- JsonParseObjectFromString
- JsonRemoveEntry
- JsonSerializeArray
- JsonSerializeObject
- KeyValuePair
- LAMP_FLASH
- LAMP INFO
- LAMP STATUS
- LargeReadRequest
- LargeWriteRequest
- LatchVariable
- LCon
- LConC
- LConTI
- LConTIC
- LContito
- LConTo
- LCTD
- LCTU
- LCTUD
- LeafTreeNode
- LeafTreeNodeOpcUA
- LeafTreeNodeTypeMember
- LeafTreeNodeTypeMemberOpcUA
- LED ID
- LEFT
- LegacyRTSVisuStructEvent2
- LEN
- LicenseFunctions
- LIMITALARM
- LimitAlarm DINT
- LimitAlarm LREAL
- LIN TRAFO
- LINE_3D
- LinearTrafo
- LinkedList
- LinkedListElementBase
- LinkedListFactory
- LinkedListlterator
- LinkState Link
- LINT TO SIGNED
- LintElement
- LintElementFactory
- List
- ListBase
- Listener
- ListFactory
- Listlterator
- ListOfDevices
- Load_Ref
- LocalDateTime
- LOG ENTRY
- LOG ENUM STATE
- LogAdd
- LogAdd2
- LogClose
- LogComponent
- LogCreate
- LogDelete
- LogFileDelete
- LogFileReduce
- LogGeneric Input
- LogGeneric Output
- LOGGER MODE
- LoggingHelper
- LogHandling
- Loglec60870 Input
- Loglec60870_Output
- LogMultifile
- LogOpen
- LogOptions
- LogToDevice
- LostMessages
- LowByte
- LowWord
- LREAL_TO_FLOAT
- LRealToHexStr
- LRealToStr
- LTIME TO DURATION
- LTIME TO INT64
- LTIME TO ISO8601
- LTIME_TO_REAL8
- LTOF
- LTON
- LTP
- LTrig
- LWORD TO HANDLE
- LWORD TO PVOID
- MAC_ADDRESS_COMPARE
- MAKE_EVENTID
- MakeNormed3D
- MappingDesc ArrayArbitrary
- MappingDesc ArraySubRange
- MATH FFT
- MATH INPUT SHAPER ZV
- MATH INPUT SHAPER_ZVD
- MATH_LINEAR_REGRESSION
- MATH MOVING AVG
- MATH NOTCH FILTER
- MAUType
- MC ABB iTypes Enum
- MC AccelerationProfile
- MC BufferMode
- MC_Cam_ld
- MC_CamIn
- MC CamOut
- MC CamTableSelect
- MC CombineAxes
- MC Direction
- MC Gearln
- MC GearInPos
- MC GearOut
- MC Halt
- MC HaltPhasing
- MC_HaltSuperlmposed
- MC_Homing_Direction
- MC Homing Edge
- MC Homing Mode
- MC LimitLoad
- MC LimitMotion
- MC LoadControl
- MC LoadProfile
- MC_LoadSuperimposed
- MC_MoveAbsolute
- MC MoveAdditive
- MC MoveContinuousAbsolute
- MC MoveContinuousRelative
- MC MoveRelative
- MC MoveSuperlmposed
- MC_MoveVelocity
- MC_PhasingAbsolute
- MC PhasingRelative
- MC PositionProfile
- MC Power
- MC PProfile
- MC ReadActualPosition
- MC_ReadActualVelocity
- MC_ReadAxisError
- MC ReadBoolParameter
- MC ReadParameter
- MC ReadStatus
- MC Reset
- MC SetOverride
- MC_SetPosition
- MC Source
- MC StepAbsSwitch
- MC StepDirect
- MC StepLimitSwitch
- MC StepRefPulse
- MC Stop
- MC TorqueControl
- MC_TProfile
- MC VelocityProfile
- MC WriteBoolParameter
- MC WriteParameter
- MCA Cam Extra
- MCA CamBounds
- MCA CamEditor
- MCA_CamEditorControlStruct
- MCA_CamEditorFeedbackStruct
- MCA CamGetInterpolationPosition
- MCA CamGetInterpolationValues
- MCA CamInDirect
- MCA CamInfo
- MCA CamStructureSelect
- MCA CAMTappet
- MCA CamTappet
- MCA DigitalCamSwitch
- MCA doWrap
- MCA DriveBasedHome
- MCA_DriveCheckReadyToEnable
- MCA DriveEnablelmmediateApply
- MCA DriveParametersSave
- MCA DriveSetOperatingMode
- MCA EncoderAxisMotionState
- MCA GearInDirect
- MCA GetCamMasterPhase
- MCA_GetE1x0DriveErrorDescription
- MCA_GetTappetValue
- MCA HomeAbsSwitch
- MCA HomeLimitSwitch
- MCA HomeToTorque
- MCA HomeToTouchProbe
- MCA Indexing
- MCA_JogAxis
- MCA_LinkOption
- MCA MoveBuffer
- MCA MoveByExternalReference
- MCA MoveByExtRefRelative
- MCA MoveFeed
- MCA MoveLink
- MCA MoveMode
- MCA_MoveRelativeOpti
- MCA MoveVelocity PV
- MCA MoveVelocityContinuous
- MCA Parameter
- MCA Parameter Struct
- MCA PhasingByMaster
- MCA Pos_Ref
- MCA_QuickStop
- MCA ReadCamFile
- MCA ReadCamTablePoint
- MCA ReadParameterList
- MCA ReadTappetTablePoint
- MCA SetOperatingMode
- MCA SetPositionContinuous
- MCA_TappetAction
- MCA TappetType
- MCA TorqueControl CSTOL
- MCA TorqueControl PT
- MCA TouchProbeECAT
- MCA TouchProbeECATFiltering
- MCA TouchProbeOBIO
- MCA_WriteCamFile
- MCA_WriteCamTablePoint
- MCA WriteParameterList
- MCA WriteTappetTablePoint
- MCA XYGroupCircMode
- MCA XYGroupEnable
- MCA XYGroupHome
- MCA XYGroupHomingSequence
- MCA XYGroupPause
- MCA XYGroupReset
- MCA XYGroupState
- MCA XYGroupStop
- MCA_XYMoveCircular
- MCA XYMoveLinear
- MD5
- MD5 FF
- MD5 GG
- MD5 HH
- MD5 II
- MD5_Transform
- MeasureFrequence
- MeasuringPoint
- MemBuffer
- MemCmp
- MemCpy
- MemFill
- MemForceSwap
- MemMove
- MemoryManager
- MemoryToString
- MemSet
- MESSAGE
- MessageBox Struct
- MessageBoxDialog_Types
- METRICS
- MID
- MILLISECOND
- MinMaxLabels
- MINUTE
- Mod Client FCT23
- MODE
- ModRtuGenDevDataType
- ModRtuGenDevDataTypeInternal
- ModRtuMast
- ModRtuMastDev
- ModRtuMastTypeInternal
- ModRtuRead
- ModRtuReadWrite23
- ModRtuToken
- ModRtuTokenDev
- ModRtuTokenType
- ModRtuWrite
- ModTcpConfig
- ModTcpInfo
- ModTcpMast
- ModTcpMast2
- ModTcpServOnOff
- Module
- Module Diag
- MODULESTATE
- ModuleState
- ModuleStatus
- MonitorDBStatus
- MonitorFilterByDateTime
- MonitorFilterByLatch
- Monitoring2ByteCode
- Monitoring2ByteCodeUnion
- MonitoringService
- MonitoringServiceHelper
- MonitoringState
- MonitorPopulateFilterCriteria
- MonLog Binary
- MonLog CurrentTime
- MonLog DumpDataltem
- MonLog_DumpDataltemBase
- MonLog_DumpDataltemWriting
- MonLog Error
- MonLog ErrorAndClose
- MonLog LogLine
- MonLog OkAndClose
- MONTH
- MQTT_CONNECTION
- MQTT_MESSAGE
- MQTT QOS
- MattConnectWithCertBuffer
- MqttConnectWithCertFile
- MattConnectWithCertStore
- MattDisconnect
- MattGetReceivedPacket
- MqttPing
- MqttPublish
- MqttSubscribe
- MattUnsubscribe
- MRC PortState
- MsgAddRef
- MsgClass
- MsgClone
- MsgGetData
- MsgGetRXChg
- MsgReceive
- MsgRelease
- MsgReleaseEx
- MsgSend
- MSK_ECM_IF_EXT_SYNC_INFO_FLAGS
- NamedTreeNode
- NamespaceNodeFlags
- NCAPDUFaultStatus
- NestingPathEntry
- NestingPathInformation
- NET INFO
- NetClientCloseChannel
- NetClientOpenChannel
- NetClientOpenChanneIResult
- NetClientSend
- NetDiagnosis
- NetVarDataltem Udp
- NetVarManager Udp FB
- NetVarOD Service Udp
- NetVarPDO Rx Udp
- NetVarPDO Tx Udp
- NetVarTelegramm Udp
- NetVarTlgHeader Udp
- NetVarUDPDiagStruct
- NetVarUDPError
- NetworkScanSettings
- NETX DEV DIAG
- NETX SYSTEM CHANNEL
- NETX UDINT TO STRINGHEX
- NetxEcatInit
- NetxEcatIsCompatible
- NMT
- NMT ERROR BEHAVIOUR
- NodeFlags
- NOP
- Norm3D
- NSC AddrComponent
- NSC_CompleteNodeInfo
- NSC_NodeAddress
- NSC NodeInfoExt
- NSC NodeInfoInt
- NSClientClose
- NSClientGeneralResolveCallback
- NSClientOpen
- NSClientResolveAll
- NSClientSearchNodeFlags
- NSClientSysMemAllocator
- NSClientTaskBase
- NSClientTaskResolveAlINodes
- NSClientTaskSearchForSpecificNode
- NSClientUtil DumpAddress
- NSClientUtil_DumpAddressHelp
- NSClientUtil_DumpCallback
- NSClientUtil DumpNodeInfo
- NSClientUtil DumpStartSearchNodeParams
- NSClientUtil Log1
- NSClientUtil Log2
- NSClientUtil Log3
- NSClientWrapper
- NtpSourceInfoData
- NtpSourceMode
- NtpSourceState
- NullElement
- OBIO PTO Motion Parameter
- OBIO PTOMotionKernel
- OBIO PTOSingleLineMotionKernel
- OBIO_PWM_Motion_Parameter
- OBIO_PWMMotionKernel
- OBIOBasicPoint2Point
- OBIOEncoderCounter
- OBIOForwardCounter
- OBIOFreqOut
- OBIOInterpolationType
- OBIOInterruptInfo
- OBIOInterruptPara
- OBIOLimitSwitch
- OBIOMotionPTO
- OBIOMotionPwm
- OBIOMotionSinglePTO
- OBIOPulseTrainOutput
- OBIOPulseTrainSingleOutput
- OBIOPwm
- OBIOSineSquarePoint2Point
- OBIOTouchProbe
- OLM OnlineLicenseManager
- OnChangePasswordDialogClosed
- OnLoginDialogClosed
- OnUserManagementDialogClosed
- OPCAClientCredentials UserPassword
- OpcUa ActivateSessionRequest
- OpcUa ActivateSessionResponse
- OpcUa AddNodesltem
- OpcUa_AddNodesRequest
- OpcUa_AddNodesResponse
- OpcUa AddNodesResult
- OpcUa AddReferencesltem
- OpcUa AddReferencesRequest
- OpcUa AddReferencesResponse
- OpcUa AggregateConfiguration
- OpcUa_AggregateFilter
- OpcUa_AggregateFilterResult
- OpcUa Annotation
- OpcUa Anonymous/dentityToken
- OpcUa ApplicationDescription
- OpcUa ApplicationType
- OpcUa Argument
- OpcUa_ArrayType
- OpcUa_AttributeOperand
- OpcUa Attributes
- OpcUa AxisInformation
- OpcUa AxisScaleEnumeration
- OpcUa Boolean
- OpcUa BrowseDescription
- OpcUa BrowseDirection
- OpcUa BrowseNextRequest
- OpcUa BrowseNextResponse
- OpcUa BrowsePath
- OpcUa BrowsePathResult
- OpcUa BrowsePathTarget
- OpcUa BrowseRequest
- OpcUa BrowseResponse
- OpcUa_BrowseResult
- OpcUa_BrowseResultMask
- OpcUa BuildInfo
- OpcUa BuiltInType
- OpcUa Byte
- OpcUa ByteString
- OpcUa CallMethodRequest
- OpcUa CallMethodResult
- OpcUa CallRequest
- OpcUa CallResponse
- OpcUa CancelRequest
- OpcUa_CancelResponse
- OpcUa_Channe/SecurityToken
- OpcUa CharA
- OpcUa CloseSecureChannelRequest
- OpcUa CloseSecureChannelResponse
- OpcUa CloseSessionRequest
- OpcUa CloseSessionResponse
- OpcUa ComplexNumberType
- OpcUa_ContentFilter
- OpcUa_ContentFilterElement
- OpcUa ContentFilterElementResult
- OpcUa ContentFilterResult
- OpcUa CreateMonitoredltemsRequest
- OpcUa CreateMonitoredltemsResponse
- OpcUa CreateSessionRequest
- OpcUa_CreateSessionResponse
- OpcUa_CreateSubscriptionRequest
- OpcUa CreateSubscriptionResponse
- OpcUa DataChangeFilter
- OpcUa DataChangeNotification
- OpcUa DataChangeTrigger
- OpcUa DataTypeAtributes
- OpcUa_DataValue
- OpcUa_DateTime
- OpcUa Decoder
- OpcUa DeleteAtTimeDetails
- OpcUa DeleteEventDetails
- OpcUa DeleteMonitoredltemsRequest
- OpcUa DeleteMonitoredltemsResponse
- OpcUa_DeleteNodesItem
- OpcUa_DeleteNodesRequest
- OpcUa DeleteNodesResponse
- OpcUa DeleteRawModifiedDetails
- OpcUa DeleteReferencesltem
- OpcUa DeleteReferencesRequest
- OpcUa DeleteReferencesResponse
- OpcUa_DeleteSubscriptionsRequest
- OpcUa_DeleteSubscriptionsResponse
- OpcUa DiagnosticInfo
- OpcUa Double
- OpcUa DoubleComplexNumberType
- OpcUa ElementOperand
- OpcUa EncodeableObjectBody
- OpcUa EncodeableType
- OpcUa_Encoder
- OpcUa_EndpointConfiguration
- OpcUa EndpointDescription
- OpcUa EndpointUrIListDataType
- OpcUa EnumDefinition
- OpcUa EnumField
- OpcUa EnumValueType
- OpcUa EUInformation
- OpcUa EventFieldList
- OpcUa EventFilter
- OpcUa EventFilterResult
- OpcUa EventNotificationList
- OpcUa_ExpandedNodeld
- OpcUa ExtensionObject
- OpcUa ExtensionObject Body
- OpcUa ExtensionObjectEncoding
- OpcUa FilterOperator
- OpcUa FindServersOnNetworkRequest
- OpcUa FindServersOnNetworkResponse
- OpcUa_FindServersRequest
- OpcUa_FindServersResponse
- OpcUa Float
- OpcUa GenericAttributes
- OpcUa GenericAttributeValue
- OpcUa GetEndpointsRequest
- OpcUa GetEndpointsResponse
- OpcUa_Guid
- OPcUa_Handle
- OpcUa HistoryData
- OpcUa HistoryEvent
- OpcUa HistoryEventFieldList
- OpcUa HistoryModifiedData
- OpcUa HistoryReadRequest
- OpcUa_HistoryReadResponse
- OpcUa_HistoryReadResult
- OpcUa HistoryReadValueld
- OpcUa HistoryUpdateDetails
- OpcUa HistoryUpdateRequest
- OpcUa HistoryUpdateResponse
- OpcUa HistoryUpdateResult
- OpcUa_HistoryUpdateType
- OpcUa IdentifierType
- OpcUa Int
- OpcUa Int16
- OpcUa Int32
- OpcUa Int64
- OpcUa IssuedIdentityToken
- OpcUa LiteralOperand
- OpcUa_LocalizedText
- OpcUa MdnsDiscoveryConfiguration
- OpcUa MessageSecurityMode
- OpcUa MethodAttributes
- OpcUa ModelChangeStructureDataType
- OpcUa ModificationInfo
- OpcUa ModifyMonitoredItemsRequest
- OpcUa_ModifyMonitoredItemsResponse
- OpcUa_ModifySubscriptionRequest
- OpcUa ModifySubscriptionResponse
- OpcUa MonitoredItemCreateRequest
- OpcUa MonitoredItemCreateResult
- OpcUa MonitoredItemModifyRequest
- OpcUa MonitoredItemModifyResult
- OpcUa MonitoredltemNotification
- OpcUa MonitoringMode
- OpcUa MonitoringParameters
- OpcUa NetworkGroupDataType
- OpcUa_NodeAttributes
- OpcUa_NodeClass
- OpcUa Nodeld
- OpcUa Nodeld anon
- OpcUa Nodelds
- OpcUa NodeReference
- OpcUa NodeTypeDescription
- OpcUa NotificationMessage
- OpcUa_ObjectAttributes
- OpcUa_ObjectTypeAttributes
- OpcUa OpenSecureChannelRequest
- OpcUa OpenSecureChannelResponse
- OpcUa OptionSet
- OpcUa ParsingResult
- OpcUa PerformUpdateType
- OpcUa_ProgramDiagnostic2DataType
- OpcUa_ProgramDiagnosticDataType
- OpcUa PublishRequest
- OpcUa PublishResponse
- OpcUa QualifiedName
- OpcUa QueryDataDescription
- OpcUa QueryDataSet
- OpcUa_QueryFirstRequest
- OpcUa_QueryFirstResponse
- OpcUa QueryNextRequest
- OpcUa QueryNextResponse
- OpcUa Range
- OpcUa ReadAtTimeDetails
- OpcUa ReadEventDetails
- OpcUa_ReadProcessedDetails
- OpcUa_ReadRawModifiedDetails
- OpcUa ReadRequest
- OpcUa ReadResponse
- OpcUa ReadValueld
- OpcUa RedundantServerDataType
- OpcUa ReferenceDescription
- OpcUa_ReferenceTypeAttributes
- OpcUa_RegisteredServer
- OpcUa RegisterNodesRequest
- OpcUa RegisterNodesResponse
- OpcUa RegisterServer2Request
- OpcUa RegisterServer2Response
- OpcUa RegisterServerRequest
- OpcUa RegisterServerResponse
- OpcUa_RelativePath
- OpcUa_RelativePathElement
- OpcUa RepublishRequest
- OpcUa RepublishResponse
- OpcUa RequestHeader
- OpcUa ResponseHeader
- OpcUa RolePermissionType
- OpcUa SamplingIntervalDiagnosticsDataType
- OpcUa SByte
- OpcUa SecurityTokenRequestType
- OpcUa SemanticChangeStructureDataType
- OpcUa_ServerDiagnosticsSummaryDataType
- OpcUa_ServerOnNetwork
- OpcUa ServerState
- OpcUa ServerStatusDataType
- OpcUa ServiceCounterDataType
- OpcUa ServiceFault
- OpcUa SessionDiagnosticsDataType
- OpcUa SessionlessInvokeRequestType
- OpcUa_SessionlessInvokeResponseType
- OpcUa_SessionSecurityDiagnosticsDataType
- OpcUa SetMonitoringModeRequest
- OpcUa SetMonitoringModeResponse
- OpcUa SetPublishingModeRequest
- OpcUa SetPublishingModeResponse
- OpcUa SetTriggeringRequest
- OpcUa_SetTriggeringResponse
- OpcUa_SignatureData
- OpcUa SignedSoftwareCertificate
- OpcUa SimpleAttributeOperand
- OpcUa StatusChangeNotification
- OpcUa StatusCode
- OpcUa StatusResult
- OpcUa_String
- OpcUa_StructureDefinition
- OpcUa StructureField
- OpcUa StructureType
- OpcUa SubscriptionAcknowledgement
- OpcUa SubscriptionDiagnosticsDataType
- OpcUa TimestampsToReturn
- OpcUa_TimeZoneDataType
- OpcUa_TransferResult
- OpcUa TransferSubscriptionsRequest
- OpcUa TransferSubscriptionsResponse
- OpcUa TranslateBrowsePathsToNodeldsRequest
- OpcUa TranslateBrowsePathsToNodeldsResponse
- OpcUa UInt
- OpcUa Uint16
- OpcUa UInt32
- OpcUa UInt64
- OpcUa UnregisterNodesRequest
- OpcUa UnregisterNodesResponse
- OpcUa UpdateDataDetails
- OpcUa UpdateEventDetails
- OpcUa UpdateStructureDataDetails
- OpcUa_UserIdentityToken
- OpcUa_UserNameldentityToken
- OpcUa UserTokenPolicy
- OpcUa UserTokenType
- OpcUa VariableAttributes
- OpcUa VariableTypeAttributes
- OpcUa Variant
- OpcUa VariantArrayType
- OpcUa VariantArrayUnion
- OpcUa VariantArrayValue
- OpcUa VariantMatrixValue
- OpcUa VariantUnion
- OpcUa_ViewAttributes
- OpcUa ViewDescription
- OpcUa WriteRequest
- OpcUa WriteResponse
- OpcUa WriteValue
- OpcUa X5091dentityToken
- OpcUa XVType
- OpcUaApplicationDescriptionClear
- OpcUaApplicationDescriptionInitialize
- OpcUaBrowsePathClear
- OpcUaBrowsePathInitialize
- OpcUaBrowsePathResultClear
- OpcUaBrowsePathResultInitialize
- OpcUaBrowseResultClear
- OpcUaBrowseResultInitialize
- OpcUaByteStringClear
- OpcUaByteStringCompare
- OpcUaByteStringConcatenate
- OpcUaByteStringCopyTo
- OpcUaByteStringInitialize
- OPCUAClient Browse
- OPCUAClient BrowseNext
- OPCUAClient_Call
- OPCUAClient Connect
- OPCUAClient Create
- OPCUAClient CreateMonitoredltems
- OPCUAClient CreateSubscription
- OPCUAClient Delete
- OPCUAClient DeleteMonitoredItems
- OPCUAClient_DeleteSubscription
- OPCUAClient Disconnect
- OPCUAClient FindServers
- OPCUAClient FindServersOnNetwork
- OPCUAClient GetConfig
- OPCUAClient GetEndpoints
- OPCUAClient ModifyMonitoredItems
- OPCUAClient ModifySubscription
- OPCUAClient Read
- OPCUAClient RegisterNodes
- OPCUAClient SetDataChangeFilterStatic
- OPCUAClient SetEventFilterStatic
- OPCUAClient SetMonitoringMode
- OPCUAClient SetPublishingMode
- OPCUAClient_TranslateBrowsePathsToNodelds
- OPCUAClient UnregisterNodes
- OPCUAClient Write
- OPCUAClientConnectionConfiguration
- OPCUAClientConnectionState
- OPCUAClientCredentials
- OPCUAClientMonitoredItemConfiguration
- OPCUAClientMonitoredItemState
- OPCUAClientSubscriptionState
- OPCUAClientUserToken
- OpcUaDataTypeDescription
- OpcUaDataValueClear
- OpcUaDataValueCompare
- OpcUaDataValueCopyTo
- OpcUaDataValueInitialize
- OpcUaDateTimeUtcNow
- OpcUaElementDescription
- OpcUaEndpointDescriptionClear
- OpcUaEndpointDescriptionInitialize
- OpcUaEventFieldListClear
- OpcUaEventFieldListInitialize
- OpcUaEventNotificationListClear
- OpcUaEventNotificationListInitialize
- OpcUaExpandedNodeldClear
- OpcUaExpandedNodeldCompare
- OpcUaExpandedNodeldCopyTo
- OpcUaExpandedNodeIdInitialize
- OpcUaExpandedNodeldIsNull
- OpcUaExtensionObjectClear
- OpcUaExtensionObjectCompare
- OpcUaExtensionObjectCopyTo
- OpcUaExtensionObjectCreate
- OpcUaExtensionObjectDelete
- OpcUaExtensionObjectInitialize
- OpcUaLocalizedTextClear
- OpcUaLocalizedTextCompare
- OpcUaLocalizedTextCopyTo
- OpcUaLocalizedTextInitialize
- OpcUaMetaDataType
- OpcUaMethodDescription
- OpcUaMethodMetaData
- OpcUaNodeldClear
- OpcUaNodeIdCompare
- OpcUaNodeldCopyTo
- OpcUaNodeldInitialize
- OpcUaNodeldlsNull
- OpcUaNodeMetaData
- OpcUaObjectDescription
- OpcUaObjectTypeDescription
- OpcUaOwnDataTypeMetaData
- OpcUaQualifiedNameClear
- OpcUaQualifiedNameCompare
- OpcUaQualifiedNameCopyTo
- OpcUaQualifiedNameInitialize
- OpcUaReadValueldClear
- OpcUaReadValueldInitialize
- OpcUaReferenceDescriptionClear
- OpcUaReferenceDescriptionInitialize
- OpcUaReferenceTypeDescription
- OpcUaServer MessageSecurityMode
- OpcUaServer Session Information
- OpcUaServer SessionEvents
- OpcUaServerGetFirstSession
- OpcUaServerGetNextSession
- OpcUaServerGetSessionInfo
- OpcUaServerNodeDescription
- OpcUaServerOnNetworkClear
- OpcUaServerOnNetworkInitialize
- OpcUaServerReferenceDescription
- OpcUaSimpleAttributeOperandClear
- OpcUaSimpleAttributeOperandInitialize
- OpcUaStackDecodeGetPosition
- OpcUaStackDecodeSetPosition
- OpcUaStackDecodeValue
- OpcUaStackEncodeGetPosition
- OpcUaStackEncodeSetPosition
- OpcUaStackEncodeValue
- OpcUaStackFinishDecode
- OpcUaStackFinishEncode
- OpcUaStackStartDecode
- OpcUaStackStartEncode
- OpcUaStatusChangeNotificationClear
- OpcUaStatusChangeNotificationInitialize
- OpcUaStatusCodeToString
- OpcUaStringAttachCopy
- OpcUaStringAttachReadOnly
- OpcUaStringAttachToString
- OpcUaStringClear
- OpcUaStringGetRawString
- OpcUaStringInitialize
- OpcUaStringlsEmpty
- OpcUaStringlsNull
- OpcUaStringStrLen
- OpcUaStringStrnCat
- OpcUaStringStrnCmp
- OpcUaStringStrnCpy
- OpcUaStringStrSize
- OpcUaTypeMetaData
- OpcUaTypeMetaDataUnion
- OpcUaVariableDescription
- OpcUaVariableTypeDescription
- OpcUaVariantClear
- OpcUaVariantCompare
- OpcUaVariantCopyTo
- OpcUaVariantInitialize
- OpcUaViewDescription
- OpcUaWellKnownDataTypeMetaData
- Open
- OpenChangePasswordDialog
- OpenChangePasswordDialog2
- OpenLoginDialog
- OpenLoginDialog2
- OpenUserManagementDialog
- OpenUserManagementDialog2
- OPERATION
- OPERATION FWK ACCESS ADDRESS
- OPERATION FWK ACCESS CONFIG
- OPERATION FWK ACCESS PARAMETER
- OPERATION FWK CALLINFO
- OPERATION FWK GET DEV STATUS PARAMETER
- OPERATION FWK SEND COMMAND
- OPERATION FWK SEND PARAMETER
- OPERATION FWK SET PARAMETER
- OPERATION FWK START SCAN
- OPERATION FWK STATUS PARAMETER
- OPERATION FWK TEST ADDRESS
- OPTION
- OS
- OurVarInfo
- PACK
- PackArrayOfBoolToArrayOfByte
- PackBitsToByte
- PackBitsToDword
- PackBitsToWord
- PackBytesToDword
- PackBytesToWord
- PacketPool
- PacketQueue
- PacketReader
- PacketWriter
- PackWordsToDword
- Pair DintDint
- Pair PStringDint
- Pair_PStringXWORD
- Pair_StringDint
- Pair StringString
- PARAMETER
- PARAMETERS INPUT SHAPING TYPE
- PARITY
- ParseCANID
- ParsePGN
- PB SlaveActivation
- PB SlaveConfigurationData
- $\quad P C B$
- $\quad \underline{D}$
- PERIOD
- PERIODE
- PERIODEINFO
- PersistantItemListUsage
- PFSYS TASK EXCEPTIONHANDLER
- PFSYS TASK FUNCTION
- PG TYPE
- PID
- PID FIXCYCLE
- PingRequest
- PINGROUP
- PLANE_H
- plc
- PLC IDENT
- PlcAbsoluteAddressInfo
- PlcAddressInfo
- PlcAddressInfoType
- PIcAddressInfoUnion
- PlcAppldentification
- PlcConnectionInitFlags
- PlcCredentiallnfo
- PlcCredentials
- PlcCryptType
- PlcDataltemAccess
- PIcDataltemQuality
- PIcDataltemQualitylinfo
- PlcDataltemType
- PlcOperationControl
- PIcPropertyByCallAddressInfo
- PM_VERSION
- PmBatt
- PmDiskLifetimeUsed
- PmDiskStatus
- PmDiskWriteProtection
- PmDispSetText
- PmEcoResetFRAM
- PmErrLedSet
- PmGetDeviceState
- PmGetPIcld
- PmLedSet
- PmNtpInfo
- PmpAntiJam
- PmpConfiguration
- PmpDolSimu
- PmpDriveSimu
- PmpEnergyCalc
- PmpFlowCalcHQ
- PmpFlowCalcPQ
- PmpFlowDistributor
- PmpInterfaceDol
- PmpInterfaceVFD
- PmPIcReboot
- PmpLevelComparator
- PmpLevelDistributor
- PmpMaintenance
- PmpPid
- PmpPressureDistributor
- PmpProtectionAnalog
- PmpProtectionBinary
- PmProdRead
- PmProdReadAsync
- PmpSequenceGen
- PmpSleep
- PmpSoftFilling
- PmpStationType
- PmpTankSimu
- PmRealtimeClock
- PmRealtimeClockDT
- PmSntplnfo
- PmSramCleared
- PmSramExport
- PmSramImport
- PmSysTime
- PmVersion
- PN ADDR
- PN AINFO
- PN_DEVICE_ID
- PN_PortConfiguration
- PN PortConfigurationRecord
- PNS CONFIG STATES
- PNS DIAG
- PNS DIAG LinkState
- PNS IF APDU STATUS CHANGED IND DATA T
- PNS IF APDU STATUS CHANGED IND T
- PNS_IF_AR_ABORT_IND_IND_T
- PNS_IF_AR_CHECK_IND_DATA_T
- PNS IF AR CHECK IND T
- PNS IF AR IN DATA IND T
- PNS IF CHECK IND
- PNS IF CHECK IND DATA T
- PNS IF CMD
- PNS_IF_EVENT_IND_T
- PNS_IF_GET_ASSET_IND_DATA_T
- PNS IF GET ASSET IND T
- PNS IF GET IP ADDR CNF DATA T
- PNS IF GET IP ADDR CNF T
- PNS IF GET STATION NAME CNF DATA T
- PNS IF GET STATION NAME CNF T
- PNS_IF_LOAD_REMANENT_DATA_REQ
- PNS_IF_READ_RECORD_IND_DATA_I
- PNS IF READ RECORD IND $T$
- PNS IF READ RECORD RSP DATA T
- PNS IF READ RECORD RSP T
- PNS IF RESET FACTORY SETTINGS IND T
- PNS IF START LED BLINKING IND T
- PNS_IF_STORE_REMANENT_DATA_IND_T
- PNS_IF_USER_ERROR_IND_DATA_T
- PNS IF USER ERROR IND T
- PNSlave
- PNSlaveDiag
- POINT
- POINT2 DINT
- POINT2 LREAL
- PolarToCartesian
- PolynomialValue
- PoolClass
- PoolCreateH
- PoolCreateP
- PoolDelete
- PoolExtendH
- PoolGetBlock
- PoolGetBlockSize
- PoolGetCurCapacity
- PoolGetNumBlocksLeft
- PoolGetSize
- PoolPutBlock
- Port
- PostEvent
- Printf
- PrintfW
- private iomgr memcpy
- PROC CMD
- PROC_STATE
- ProfinetByteData
- ProfinetConfigType
- ProfinetController
- ProfinetControllerDiag
- PROJECT INFO
- ProjectPointOnLine
- ProjectPointOnPlane
- PropertyAddressinfo
- PropertyAddrString
- PropertyAttributeExistenceString
- PropertyAttributePersistentString
- PropertyAttributesString
- PropertyAttributeWritableString
- PropertyIndexAddrString
- PropertyInfo
- PropertyInfoRemote
- PropertyLocation
- PROTOCOL DATA UNIT
- ProtocolDataUnit
- ProxyEnumState
- ProxyFbHistActiveAlarmsQueue
- ProxyFbHistAlarmsRowQueue
- ProxyStructError
- ProxyStructMonitor
- ProxyStructMonitorAlarmClassDesc
- ProxyStructMonitorAlarmDesc
- ProxyStructMonitorAlarmGroupDesc
- ProxyStructMonitorRequest
- prvCheckUserMgrlsActive
- prvGetTags
- PRVREC
- PRVREC MODE
- PStrCat
- PStrCmp
- PStrICmp
- PStrIFind
- PStringElement
- PStringElementFactory
- PStringToDintMap
- PStringToXWORDMap
- PStringVector
- PStrLen
- PStrLenUntil
- PStrNICmp
- PStrToUpper
- PT SIZE
- PtrToString
- PURPOSE
- PUTBIT
- PVOID
- PVOID TO DWORD
- PVOID TO LWORD
- PVOID TO WORD
- QOS INFO
- Queue
- QueueFactory
- QuickSortAddrltemHelpers
- $\quad$ TRIG
- RALARM
- RALARM MODE
- RAMP INT
- RAMP_REAL
- RCVREC
- RCVREC MODE
- RCX SET WATCHDOG TIME CNF T
- RCX SET WATCHDOG TIME REQ DATA T
- RCX SET WATCHDOG TIME REQ T
- RDIAG
- RDREC
- Read
- ReadArbitraryStringFromBuffer
- ReadEEpromData
- Readldentification
- ReadMemory
- ReadNbrSlaves
- ReadRequest
- ReadWriteEEprom
- REAL8
- REAL8 TO DT
- REAL8 TO LTIME
- REAL8 TO TIME
- REAL TO FLOAT
- RealToHexStr
- RealToStr
- ReceiveParameterGroup
- ReceiveWatchdog
- Recipe FileParameters
- RecipeMan FctTypeClassToDataType
- RecipeManCommands
- Reconfigure
- RECV EMCY
- RECV EMCY DEV
- RedundancyManager
- RedundancyState
- RegContext
- Register
- RegisterCallback
- RegisterldArea
- RemoteAdapter
- RemoteAdapter Diag
- RemoteAdapter diag
- RemotePlcRequestIdentification
- RemoteProcedureCall
- RemoteVarlnfo
- RemoteVarResolver
- Rename
- ReparselOMemoryAccessExpression
- REPLACE
- Replacementinfo
- REQUEST
- Request
- RequestFactory
- Reset
- RESET INIT
- RESET OPTION
- RESET RESET
- ResetBusAlarm
- ResetNodeInfo
- ResetNodeInfolnt
- ResolveHostname
- ReturnValues
- ReusableRequestInfo
- ReusableRequestOperations
- ReverseBits/nBYTE
- ReverseBitsInDWORD
- ReverseBitsInWORD
- ReverseBYTEsInDWORD
- ReverseBYTEsInWORD
- ReverseWORDsInDWORD
- RIGHT
- RLstAddPrio
- RLstCheckPrio
- RLstClass
- RLstCreateH
- RLstCreateP
- RLstDelete
- RLstGetHighestPrio
- RLstGetSize
- RLstRemovePrio
- RootAddressDatasourceIndex
- RootDatasourceIndex
- RootPseudo
- RootRenamed
- RootRenamedDatasourceIndex
- ROTATION DIFFERENCE
- RouterGetHostAddress
- RouterGetInstanceByName
- RouterGetName
- RouterGetParentAddress
- RPCDataRepresentation
- RPCNCARejectStatus
- $R S$
- RSM_HANDLE
- RSMClass
- RTC
- RtcGetTime ms
- RTCLK GETDATEANDTIME PARAMS
- RTCLK GETTIMEZONEINFORMATION PARAMS
- RTCLK PERIODE INFO
- RTCLK SETDATEANDTIME PARAMS
- RTCLK SETTIMEZONEINFORMATION PARAMS
- RTCLK SYSTEMTIME
- RTCLK TIME ZONE INFO
- RTR_AddrComponent
- RTR_NodeAddress
- RTS CMBOXENTRY
- RTS CODEMETER INFO
- RTS CONTROL
- RTS IEC CWCHAR
- RTS IEC HANDLE
- RTS IEC RESULT
- RTSIEC SIZE
- RTS_SOCKET_SO_VALUE_IP_MREQ
- RTS SOCKET SO VALUE LINGER
- RTS SOCKET SO VALUE TCP KEEPALIVE
- RTS SYSTIMEDATE
- RtsBrowselnfo
- RtsByteString
- RtsCertEncoding
- RtsCertTrustLevel
- RtsCryptoID
- RtsCryptoKey
- RtsCryptoKeyStorage
- RtsCryptoKeyType
- RtsCryptoType
- RtsKdfParameter
- RtsOID
- RtsOIDClear
- RtsOIDCreate
- RtsOIDGetID
- RtsOIDGetName
- RtsOIDStore
- RtsScryptParameter
- RtsServicehandlerBase
- RtsServicehandlerBase2
- RtsX509AltName
- RtsX509AltNameStore
- RtsX509AltNameType
- RtsX509CertCheckFlags
- RtsX509CertFilter
- RtsX509CertFilterContent
- RtsX509CertFilterType
- RtsX509Certlinfo
- RtsX509CertName
- RtsX509ExKeyUsage
- RtsX509NameEntry
- RtsX509VerifyFlags
- RXChgGetCurCapacity
- RXChgGetMsgSize
- S ALLOC
- SAdapterFlags
- SafeStringCopy
- SALARM
- ScalProd3D
- ScalProd3DStand
- ScannerState
- SchedGetCurrentTask
- SchedGetNumOfTasks
- SchedGetProcessorLoad
- SchedGetTaskEventByHandle
- SchedGetTaskHandleByIndex
- SchedGetTaskHandleByName
- SchedGetTaskInterval
- SchedPostExternalEvent
- SchedRegisterExternalEvent
- SchedSetTaskInterval
- Schedule
- SchedUnregisterExternalEvent
- SchedWaitBusy
- SchedWaitSleep
- SDO ABORT
- SDO ERROR
- SDO MODE
- SDO_READ
- SDO_READ4
- SDO READ DATA
- SDO WRITE
- SDO WRITE4
- SDO WRITE DATA
- SdoAbort
- SdoRead
- SdoWrite
- SECOND
- Segment
- SegmentPool
- Select
- SEMA
- SendEvent
- SeparateDateTime
- SerializeHexReal
- ServiceGroup
- SERVICEHANDLER PARAMETER
- ServiceHeader
- ServiceReader
- ServiceRequest
- ServiceRequestBase
- ServiceRequestRaw
- ServiceResponse
- ServiceWriter
- ServiceWriterSavepoint
- Set Attribute List
- Set Attribute Single
- Set_Attributes_All
- SETBIT
- SetBitValue
- SetCiAState
- SetCustomMapping
- SetDateAndTime
- SetError
- SETIO PART
- SetParent
- SetPos
- SetPropertyAgain
- SetResult
- SettgBeginUpdate
- SettgEndUpdate
- SettgGetIntValue
- SettgGetStringValue
- SettgGetWStringValue
- SettgRemoveKey
- SettgSetIntValue
- SettgSetStringValue
- SettgSetWStringValue
- SetTimeZoneInformation
- SettingsHelper
- SettingValue
- SetValueOfTypeDesc
- Severity
- SFCActionControl
- SFCActionType
- SFCSTepType
- sgn
- SIGNED
- SIGNED TO DINT
- SIGNED TO INT
- SIGNED_TO_LINT
- SIZE
- SIZE TO UDINT
- SIZE TO UINT
- SIZE TO ULINT
- SlaveStateBitFieldType
- SLOT ID
- Sm560Rec
- Sm560Send
- SntpSourceInfoData
- SntpSourceMode
- SntpSourceState
- SOCK ADAPTER INFORMATION
- SOCK ADAPTER INFORMATION2
- SOCK_ADAPTER INFORMATION3
- SOCK HOSTENT
- SOCK RECVMSG MSG
- SockAddr
- SOCKADDRESS
- SOCKET FD SET
- SOCKET TIMEVAL
- SocketType
- SolarBackTracking
- SolarEncoderCD522
- SolarEncoderIO
- SolarEncoderOnboardIO
- SolarEWAxisPrimary
- SolarEWAxisSecondary
- SolarHydraulicControl
- SolarModeCalibration
- SolarModeHoming
- SolarModeManual
- SolarModePosition
- SolarModeTracking
- SolarNoaa
- SolarNormalizeAnalog
- SolarNrel
- SolarNSAxisPrimary
- SolarNSAxisSecondary
- SolarPositionControl
- SortByAddrltemHelper
- SortedBranchNamedTreeNode
- SortedDirectoryFileElementFactory
- SortedInstancePathBuildingBranchNode
- SortedList
- SortedListFactory
- SortedPStringVector
- SplitDateTime
- SplitString
- SplitTextListld
- SQLSTATEMENT
- $\quad S R$
- SR Mode
- SRAM CLEARED
- SRAM EXPORT
- SRAM IMPORT
- SRAMArea
- SRstate
- Stack
- StackFactory
- Start
- StatDynMemory
- STATE
- StateFlags
- StateMachine
- StaticCredentialsProvider
- StaticMemBuffer
- Statistics DINT
- STATISTICS INT
- Statistics LREAL
- Statistics LTIME
- STATISTICS REAL
- STK INFO
- STK NODES
- STK SPEC
- STK STATE
- StkClose
- StkGetInfo
- StkOpen
- StkRegister
- StkUnregister
- STO BLOB
- STO METRICS
- STO TEXT
- Stop
- STOPBIT
- Storage
- StrCaseCmpA
- StrCaseCmpEndA
- StrCaseCmpStartA
- StrCaseCmpW
- StrCaseFindA
- StrCaseFindW
- StrCmpA
- StrCmpEndA
- StrCmpStartA
- StrCmpW
- StrConcatA
- StrConcatW
- StrCpyA
- StrCpyW
- StrCpyWtoA
- StrDeleteA
- StrDeleteW
- Stream
- STREAM STATE
- StrFindA
- StrFindW
- StringBuilder
- StringBuilderSysMemExtending
- StringElement
- StringElementFactory
- StringToDintMap
- StringToStringMap
- StringVector
- StrisNullOrEmptyA
- StrisNullOrEmptyW
- StrLenA
- StrLenW
- StrMidA
- StrMidW
- StrPadLeftA
- StrPadLeftW
- StrPadRightA
- StrPadRightW
- StrReplaceA
- StrReplaceW
- StrToLowerA
- StrToLReal
- StrToReal
- StrToUpperA
- StrTrimA
- StrTrimEndA
- StrTrimEndW
- StrTrimStartA
- StrTrimStartW
- StrTrimW
- StructFilteringName
- StuSprintf
- StuSprintfW
- StyleUtilFct GetBoolFromStyle
- StyleUtilFct GetBoolFromStyleEnumOrExplicitValue
- StyleUtilFct GetSimpleTypeFromStyleEnumOrExplicitValue
- StyleUtilFct_GetUDIntFromStyle
- SubmoduleDiagnosisEntry
- SubmoduleInfo
- Submodulelterator
- SubmoduleState AddInfo
- SubmoduleState ARInfo
- SubmoduleState Detail
- SubmoduleState IdentInfo
- SubmoduleStatus
- SUBSLOT_ID
- SubVector
- Swap
- SwapByteArrayByWords
- SwapDword
- SwapDwordlf
- SwapLocaIToIntel
- SwapLocalToMotorola
- SwapLword
- SwapLwordlf
- SwappedDirectAssigner
- SwappingInfo
- SwapWord
- SwapWordlf
- SWITCHBIT
- Symbol Translation
- SymbolicInfo
- SymbolicVarNodeAccessor
- SymbolicVarNodeFinder
- SymbolicVarsBaseHandleConverter
- Symbollnfo
- Symbols
- SymbolsBaseNode
- SymbolsBranchNode
- SymVarAccess
- SYNC INFO
- SYS FILE STATUS
- SYS FILETIME
- SYS TASK_INFO
- SYS TASK PARAM
- SYS TIME
- SysCpuAtomicAdd
- SysCpuAtomicAdd64
- SysCpuAtomicCompareAndSwap
- SysCpuCalllecFuncWithParams
- SysCpuResetBit
- SysCpuResetBit2
- SysCpuSetBit
- SysCpuSetBit2
- SysCpuTestAndReset
- SysCpuTestAndSet
- SysCpuTestAndSetBit
- SysDirClose
- SysDirCopy
- SysDirCreate
- SysDirCreate2
- SysDirDelete
- SysDirDelete2
- SysDirGetCurrent
- SysDirOpen
- SysDirRead
- SysDirRename
- SysDirSetCurrent
- SysEthernetAdapterClose
- SysEthernetAdapterOpen
- SysEthernetCapabilities
- SysEthernetEthFrameReceive
- SysEthernetEthFrameSend
- SysEthernetFrame
- SysEthernetFrameRelease
- SysEthernetGetCapabilities
- SysEthernetGetInterfaceCounters
- SysEthernetGetMediaCounters
- SysEthernetGetPortConfigAndStatus
- SysEthernetInterfaceCounters
- SysEthernetlpFrameReceive
- SysEthernetlpFrameSend
- SysEthernetMediaCounters
- SysEthernetPortConfigAndStatus
- SysEthernetSetAutoNegAdvertisedCap
- SysEthernetSetAutoNegMode
- SysEthernetSetMauType
- SysEventCreate
- SysEventDelete
- SysEventSet
- SysEventWait
- SysExceptGenerateException
- SysFileClose
- SysFileCopy
- SysFileDelete
- SysFileDeleteByHandle
- SysFileEOF
- SysFileFlush
- SysFileGetName
- SysFileGetName2
- SysFileGetPath
- SysFileGetPos
- SysFileGetSize
- SysFileGetSizeByHandle
- SysFileGetStatus
- SysFileGetStatus2
- SysFileGetTime
- SysFileloctl
- SysFileOpen
- SysFileRead
- SysFileRename
- SysFileSetPos
- SysFileTruncate
- SysFileWrite
- SysMCBDAlloc
- SysMCBDCount
- SysMCBDFree
- SysMCBDGetFirstID
- SysMCBDGetNextID
- SysMCBDIsSet
- SysMCGetLoad
- SysMCGetNumOfCores
- SysMCGetProcessBinding
- SysMCGetTaskBinding
- SysMemAllocData
- SysMemCmp
- SysMemCpy
- SysMemForceSwap
- SysMemFreeData
- SysMemGetCurrentHeapSize
- SysMemIsValidPointer
- SysMemMove
- SysMemReallocData
- SysMemSet
- SysMemSwap
- SysProcessCreate
- SysProcessCreate2
- SysProcessExecuteCommand
- SysProcessExecuteCommand2
- SysProcessFreeHandle
- SysProcessGetCurrentHandle
- SysProcessGetOSId
- SysProcessGetPriority
- SysProcessGetState
- SysProcessResume
- SysProcessSetPriority
- SysProcessTerminate
- SysSemCreate
- SysSemDelete
- SysSemEnter
- SysSemLeave
- SysSemTry
- SysSharedMemoryClose
- SysSharedMemoryCreate
- SysSharedMemoryDelete
- SysSharedMemoryGetPointer
- SysSharedMemoryOpen2
- SysSharedMemoryRead
- SysSharedMemoryReadByte
- SysSharedMemoryWrite
- SysSharedMemoryWriteByte
- SysSock2Accept
- SysSock2Bind
- SysSock2Close
- SysSock2Connect
- SysSock2Create
- SysSock2FdInit
- SysSock2Fdlsset
- SysSock2FdZero
- SysSock2GetOption
- SysSock2GetPeerName
- SysSock2GetSockName
- SysSock2Htonl
- SysSock2Htons
- SysSock2InetAddr
- SysSock2InetNtoa
- SysSock2loctl
- SysSock2Listen
- SysSock2Ntohl
- SysSock2Ntohs
- SysSock2Recv
- SysSock2RecvFrom
- SysSock2RecvMsg
- SysSock2Select
- SysSock2Send
- SysSock2SendTo
- SysSock2SetOption
- SysSock2Shutdown
- SysSockAccept
- SysSockAsyncFB
- SysSockBind
- SysSockClose
- SysSockCloseUdp
- SysSockConnect
- SysSockCreate
- SysSockCreateUdp
- SysSocket2 Parameter
- SysSocket2 SpecificParameter
- SysSocket2 StdSockets
- SysSocket2 TIsSockets
- SysSocket2_Type
- SysSocketPair
- SysSockFdlnit
- SysSockFdlsset
- SysSockFdZero
- SysSockGetAdapterInfo
- SysSockGetFirstAdapterInfo
- SysSockGetHostByName
- SysSockGetHostName
- SysSockGetNextAdapterInfo
- SysSockGetOption
- SysSockGetOSHandle
- SysSockGetPeerName
- SysSockGetRecvSizeUdp
- SysSockGetSockName
- SysSockGetSubnetMask
- SysSockHtonl
- SysSockHtons
- SysSockInetAddr
- SysSockInetNtoa
- SysSockloctl
- SysSockListen
- SysSockNtohl
- SysSockNtohs
- SysSockPing
- SysSockRecv
- SysSockRecvFrom
- SysSockRecvFromUdp
- SysSockRecvFromUdp2
- SysSockRecvMsg
- SysSockSelect
- SysSockSend
- SysSockSendTo
- SysSockSendToUdp
- SysSockSetDefaultGateway
- SysSockSetIPAddress
- SysSockSetlpAddressAndNetMask
- SysSockSetOption
- SysSockSetSubnetMask
- SysSockShutdown
- SysTargetGetDeviceName
- SysTargetGetld
- SysTargetGetNodeName
- SysTargetGetOperatingSystemId
- SysTargetGetProcessorld
- SysTargetGetSerialNumber
- SysTargetGetType
- SysTargetGetVendorName
- SysTargetGetVersion
- SysTargetOperationNumber
- SysTaskAutoReleaseOnExit
- SysTaskCheckStack
- SysTaskCreate
- SysTaskCreate2
- SysTaskDestroy
- SysTaskEnd
- SysTaskEnter
- SysTaskExit
- SysTaskGenerateException
- SysTaskGetContext
- SysTaskGetCurrent
- SysTaskGetCurrentOSHandle
- SysTaskGetInfo
- SysTaskGetInterval
- SysTaskGetName
- SysTaskGetOSHandle
- SysTaskGetOSPriority
- SysTaskGetPriority
- SysTaskJoin
- SysTaskLeave
- SysTaskResume
- SysTaskSetExit
- SysTaskSetInterval
- SysTaskSetPriority
- SysTaskSuspend
- SysTaskWaitlnterval
- SysTaskWaitSleep
- SysTaskWaitSleepUs
- SYSTEM MEMORY INFORMATION
- SystemParameter
- SYSTEMTIME
- SYSTIME
- SYSTIMEDATE
- SysTimeGetMs
- SysTimeGetNs
- SysTimeGetUs
- SysTimeRtcControl
- SysTimeRtcConvertDateToHighRes
- SysTimeRtcConvertDateToUtc
- SysTimeRtcConvertHighResToDate
- SysTimeRtcConvertHighResToLocal
- SysTimeRtcConvertLocalToHighRes
- SysTimeRtcConvertLocalToUtc
- SysTimeRtcConvertUtcToDate
- SysTimeRtcConvertUtcToLocal
- SysTimeRtcGet
- SysTimeRtcGetTimezone
- SysTimeRtcHighResGet
- SysTimeRtcHighResSet
- SysTimeRtcSet
- SysTimeRtcSetTimezone
- SYSTYPE
- TA5130KNXPB
- TA5131RTC
- TA5141RS232I
- TA5142RS485
- TargetVisuCyclic
- TargetVisuFindByld
- TargetVisuNotify
- Task Desc
- Task_Desc2
- TASK_GROUP
- Task Info2
- TASK NAME
- Task WatchdogHitCount
- TaskLock
- TaskUnlock
- tCmpLogAdd
- TCP_Client
- TCP Connection
- TCP Processor
- TCP Read
- TCP ReadBuffer
- TCP Reader
- TCP Server
- TCP_Write
- TCP_WRITE_STATE
- TCP WriteBuffer
- TCP Writer
- teClass
- teEcatDcControlState
- teEcatDevState
- teEcatSIvDcInfoFlags
- teErrorCodesOB
- teEvent
- teExtSyncInfoFlags
- teHwld
- Test_State
- Testcases
- TEXT
- TextCopyToString
- TextCopyToWString
- TextFree
- TextHelper
- TextListForCombobox_CIPClass
- TICK
- TICK TO UDINT
- TICK TO UINT
- TICK TO ULINT
- TIME TO DURATION
- TIME TO INT64
- TIME_TO_ISO8601
- TIME_TO_REAL8
- TIME ZONE INFO
- TimeElement
- TimeElementFactory
- TimerSwitch
- TIMESTAMP
- Timestamp to DT
- TimeZone
- TimezoneInformation
- TimingControlledBehaviourModelBase
- TimingController
- TL AlarmStatus
- TL AlarmTableColumnTitles
- TL_DateTime
- TL_ElementProperties
- TL RecipeManager
- TL VUM Errors
- TLR PACKET HEADER T
- TLS VERSION
- TLSContext
- TO IDeviceCM579EtherCAT
- TODConcat
- TODSplit
- TOF
- TON
- TP
- TraceAbsoluteAddressInfoWriter
- TraceAbstractAddressInfoWriter
- TraceAddress
- TraceFctGetPropertyValue
- TraceFctGetVariableName
- TraceFctGetVariableNameW
- TraceMgrGetConfigFromFile
- TraceMgrGetConfigFromFileRelease
- TraceMgrPacketCheckTrigger
- TraceMgrPacketClose
- TraceMgrPacketComplete
- TraceMgrPacketCreate
- TraceMgrPacketDelete
- TraceMgrPacketDisable
- TraceMgrPacketDisableTrigger
- TraceMgrPacketEnable
- TraceMgrPacketEnableTrigger
- TraceMgrPacketGetAbsoluteStartTime
- TraceMgrPacketGetChangeTimestamp
- TraceMgrPacketGetConfig
- TraceMgrPacketGetFirst
- TraceMgrPacketGetNext
- TraceMgrPacketGetStartTime
- TraceMgrPacketGetState
- TraceMgrPacketOpen
- TraceMgrPacketReadBegin
- TraceMgrPacketReadEnd
- TraceMgrPacketReadFirst
- TraceMgrPacketReadFirst2
- TraceMgrPacketReadNext
- TraceMgrPacketReadNext2
- TraceMgrPacketResetTrigger
- TraceMgrPacketRestart
- TraceMgrPacketRestore
- TraceMgrPacketStart
- TraceMgrPacketStop
- TraceMgrPacketStore
- TraceMgrRecordAdd
- TraceMgrRecordGetConfig
- TraceMgrRecordGetFirst
- TraceMgrRecordGetNext
- TraceMgrRecordRemove
- TraceMgrRecordUpdate
- TraceMgrRecordUpdate2
- TraceMgrRecordUpdate3
- TraceMgrRecordUpdate4
- TraceMgrRecordUpdate5
- TracePacketConfiguration
- TracePropertyByCallAddressInfoWriter
- TraceRecordConfiguration
- TraceRecordEntry
- TraceState
- TraceTrigger
- TraceVariable
- TraceVariableAddress
- TraceVarinfo
- TRANSITION STATE
- TRANSMISSION
- TransmissionTrigger
- TransmitParameterGroup
- Tree
- TreeBase
- TreeNode
- TreeNodeFactory
- TreeNodeType
- TrendFbDatabaseAccessErrorHandler
- TrendFbTrendStorageWriterReader
- TrendFctCursorSearchFirstRow
- TrendFctGetDBFileSize
- TrendFctGetDBPercLimit
- TrendFctGetDBReOrderingData
- TrendFctGetDBReOrderingForceFailure
- TrendFctGetRecordSizeFactor
- TrendFctGetTimestamp
- TrendFctResetDBReOrderingData
- TrendFctSetComplexElementCallState
- TrendFctSetDBPercLimit
- TrendFctSetDBReOrderingForceFailure
- TrendFctSetRecordSizeFactor
- TrendFctShowLossOfPrecisionWarning
- TrendFctShowUnsupportedFunctionWarning
- TrendStorageCacheltemData
- TrendStorageConvertFromTimestamp
- TrendStorageConvertToTimestamp
- TrendStorageReader
- TrendStorageReaderValueConverter
- TrendStorageVariableDescription
- TriggerState
- TriggerValue
- Truncate
- TruncateF
- TryAdjustData_Int
- TryAdjustData Real
- TryAdjustData UInt
- tsEcmExtSyncInfo
- tsEcmMstrDcInfo
- tsEcmMstrFrameLossCnt
- tsEcmMstrFrameLossCntEntry
- tsEcmMstrInfo
- tsEcmMstrMemInfo
- tsEcmMstrThresholdCnt
- tsEcmMstrThresholdCntEntry
- tsEcmMstrTimingInfo
- tsEcmSlvConnInfo
- tsEcmSIVDcInfo
- tsEcmS/vEmergencies
- tsEcmSIvEmergency
- tsEcmS/vESCVersion
- tsEcmSIvESCVersion 1
- tsEcmSIVInfo
- tsEcmSIvLostLinkCnt
- tsEcmSIvRxErrorCnt
- tsNetxChannel
- tsNetxEcatBusScanDeviceInfo
- tsNetxEcatHandle
- tsParameterStruct
- tSysSockAccept
- tSysSockBind
- tSysSockClose
- tSysSockCloseUdp
- tSysSockConnect
- tSysSockCreate
- tSysSockCreateUdp
- tSysSockGetHostByName
- tSysSockGetHostname
- tSysSockGetOption
- tSysSockGetOsHandle
- tSysSockGetRecvSizeUdp
- tSysSockGetSubnetMask
- tSysSockHtonl
- tSysSockHtons
- tSysSockInetAddr
- tSysSockInetNtoa
- tSysSockloctl
- tSysSockListen
- tSysSockNtohl
- tSysSockNtohs
- tSysSockPing
- tSysSockRecv
- tSysSockRecvFrom
- tSysSockRecvFromUdp
- tSysSockSelect
- tSysSockSend
- tSysSockSendTo
- tSysSockSendToUdp
- tSysSockSetlpAddress
- tSysSockSetOption
- tSysSockSetSubnetMask
- tSysSockShutdown
- tyIEC61850 ASN1 Header
- tyIEC61850 AT AnalogueValue
- tyIEC61850 AT AnalogueValue Struct
- tyIEC61850_AT_APC
- tyIEC61850_AT_APC1
- tylEC61850 AT APC Operate
- tyIEC61850 AT APC Operate SP
- tyIEC61850 AT BOOLEAN
- tyIEC61850 AT BSC Operate
- tylEC61850 AT Check
- tylEC61850 AT CODED ENUM
- tyIEC61850 AT DPC Operate
- tyIEC61850 AT DstAddress
- tyIEC61850 AT EntryTime
- tyIEC61850 AT ENUM CtIModels
- tyIEC61850 AT ENUM MODE
- tyIEC61850 AT ENUM SboClass
- tyIEC61850 AT ENUMERATED
- tyIEC61850_AT_FLOAT32
- tyIEC61850_AT_INC
- tyIEC61850 AT INC1
- tyIEC61850 AT INC Operate
- tyIEC61850 AT INS
- tyIEC61850 AT INT128
- tyIEC61850 AT INT16
- tyIEC61850 AT INT16U
- tyIEC61850 AT INT32
- tyIEC61850 AT INT32U
- tyIEC61850 AT INT8
- tyIEC61850 AT INT8U
- tyIEC61850_AT_ISC_Operate
- tyIEC61850 AT Octet255
- tyIEC61850 AT Octet64
- tylEC61850 AT Origin
- tyIEC61850 AT POINT
- tyIEC61850 AT PulseConfig
- tyIEC61850 AT Quality
- tyIEC61850_AT_RANGECONFIG
- tyIEC61850_AT ScaledValConfig
- tylEC61850 AT SPC
- tyIEC61850 AT SPC Operate
- tylEC61850 AT StatusValue Struct
- tylEC61850 AT TimeStamp
- tyIEC61850 AT UCSTRING255
- tyIEC61850_AT_UINT32
- tyIEC61850_AT_UNIT
- tyIEC61850 AT ValWithTrans
- tyIEC61850 AT VECTOR
- tyIEC61850 AT VisSTRING129
- tyIEC61850 AT VisSTRING255
- tylEC61850 AT VisSTRING32
- tyIEC61850_AT_VisSTRING64
- tyIEC61850_AT VisSTRING65
- tylEC61850 DataPoint
- tyIEC61850 DataSetRef
- tyIEC61850 GOOSE Check
- tyIEC61850 GOOSEMsg
- tyIEC61850 MMS DataExchange
- tyIEC61850 MMS_Initiate
- tylEC61850 SubsDataBlock
- tylEC61850 SubsDataPoint
- tyISO8073 BlockHeader
- tyISO8073 ClientPara
- tyISO8073 PDU
- tyISO8327 BlockHeader
- tyISO8327 ClientData
- tyISO8327_Connect_Acceptltem
- tyISO8327 Connectionldent
- tyISO8650 UserInfoData
- tylSO8823 ContextList
- tyISO8823 ContextName
- tyISO8823 CP Type
- tyISO8823 DataUser
- tyISO8823_NormalModePara
- tyISO_BlockHeader
- tyISO SPDU
- TypeClass
- TypeClass3
- TypedElement
- TypeDesc
- TypeDesc Alias
- TypeDesc AliasWithAttributes
- TypeDesc Array
- TypeDesc Array ByteAddressed
- TypeDesc_Array Remote
- TypeDesc_Enum
- TypeDesc EnumWithAttributes
- TypeDesc Executable
- TypeDesc Executable2
- TypeDesc OpcUaBuiltInType
- TypeDesc Property
- TypeDesc Property Remote
- TypeDesc_Reference
- TypeDesc_Simple
- TypeDesc Simple Bit
- TypeDesc Struct
- TypeDesc Struct2
- TypeDesc Struct2 WithBaseType
- TypeDesc Struct2 WithBaseTypeAndAttributes
- TypeDesc_Struct_Derived_Remote
- TypeDesc_Struct_Remote
- TypeDesc Subrange
- TypeDesc UAProperties
- TypeDesc VarLenArray
- TypeDescArrayAsStruct
- TypeDescAsUnion
- TypeDescSimpleAsStruct
- TypeDescStructAsStruct
- TypeDescUnion
- TypeDescVarArrayAsStruct
- TypedList
- TypedTree
- TypeHasCompleteBlittableLayout
- Typelds
- TZ_NAME
- UDINT IN BYTES
- UDINT IN WORDS
- UDINT TO COUNT
- UDINT TO HEX
- UDINT TO IPARRAY
- UDINT TO IPSTRING
- UDINT-TO_SIZE
- UDINT TO TICK
- UDINT TO UNSIGNED
- UdintElement
- UdintElementFactory
- UDP GetDataSize
- UDP Peer
- UDP_Processor
- UDP_Receive
- UDP ReceiveBuffer
- UDP Receiver
- UDP REPLY
- UDP REPLY2
- UDP Send
- UDP SendBuffer
- UDP Sender
- UDPDriver
- UdpGetReceiveDataSize
- UdpOpenReceiveSocket
- UdpOpenSendSocket
- UdpReceiveData
- UdpSendData
- UINT TO COUNT
- UINT TO HEX
- UINT TO SIZE
- UINT TO TICK
- UINT_TO_UNSIGNED
- UintElement
- UintElementFactory
- ULINT TO COUNT
- ULINT TO SIZE
- ULINT TO TICK
- ULINT TO UNSIGNED
- UlintElement
- UlintElementFactory
- UNPACK
- UnpackArrayOfByte
- UnpackByte
- UnpackDWord
- UnpackWord
- Unregister
- UnregisterCallback
- UnregisterldArea
- UNSIGNED
- UNSIGNED TO UDINT
- UNSIGNED TO UINT
- UNSIGNED TO ULINT
- UpdateByDefaultInfo
- UpdateByDefaultitem
- UpdateDiagnosis Status
- UpdateDiagnosisEntry
- UserConfigurationV1
- UserMgrCacheValue
- UserMgrChangeMyPassword
- UserMgrCredentials
- UserMgrCredentials Password
- UserMgrCredentials Store
- UserMgrCredentials Type
- UserMgrGetSessionUser
- UserMgrGetUserAccessRights
- UserMgrGroupAdd
- UserMgrGroupAddUser
- UserMgrGroupGetFirst
- UserMgrGroupGetFirstUser
- UserMgrGroupGetHandle
- UserMgrGroupGetName
- UserMgrGroupGetNext
- UserMgrGroupGetNextUser
- UserMgrGroupRemove
- UserMgrGroupRemoveUser
- UserMgrHasUserAccessRights
- UserMgrIsActive
- UserMgrLogin
- UserMgrLogout
- UserMgrObjectAdd
- UserMgrObjectAddGroup
- UserMgrObjectClearRights
- UserMgrObjectGetFirstChild
- UserMgrObjectGetFirstGroup
- UserMgrObjectGetGroupRights
- UserMgrObjectGetHandle
- UserMgrObjectGetName
- UserMgrObjectGetNextChild
- UserMgrObjectGetNextGroup
- UserMgrObjectRemove
- UserMgrObjectRemoveGroup
- UserMgrObjectSetGroupDeniedRights
- UserMgrObjectSetGroupRights
- UserMgrObjectSetUsedRights
- UserMgrRelogin
- UserMgrTypeClass
- UserMgrUserAdd
- UserMgrUserAddInfoToUser
- UserMgrUserGetFirst
- UserMgrUserGetInfoOfUser
- UserMgrUserGetName
- UserMgrUserGetNext
- UserMgrUserGetProperty
- UserMgrUserRemove
- UserMgrUserRemoveInfoFromAllUsers
- UserMgrUserRemoveInfoFromUser
- UserMgrUserSetCredentials
- UserMgrUserSetProperty
- UserMgrValue
- UtilAddrEqual
- UtilAreGuidsEqual
- UtilBuildItemErrorCode
- UtilBytesFromHexSubString
- UtilByteToHexString
- UtilFillLanguageGroupNames
- UtilFillNodeAddress
- UtilGetHostName
- UtilGetLocalByteorder
- UtilGetLocalByteorderAtRuntime
- UtilGuidToString
- UtillsGeneralErrorReply
- UtillsGuidEmpty
- UtillsIntelByteorder
- UtillsToSwap
- UtilNumeric VersionToString
- UtilReadAddressFromRouter
- UtilResolveItemToRemoteAddr
- UtilSizeFromAddrInfo
- UtilSwapGuidlf
- UtilsWriteBYTE
- UtilsWriteString
- UtilsWriteUDINT
- UtilsWriteUINT
- UtilsWriteULINT
- UtilTokenizer
- UtilValidateByteOrder
- UtilVersionToString
- UUID
- UUID COMPARE
- UUIDGenerator
- ValueToString
- VarAccUaNamespaceFragment
- VariableInformation
- VariableInformationStruct
- VariableInformationStruct2
- VariableInformationStruct3
- VariableInformationStruct4
- VariableInformationStruct5
- VARIANCE
- Variance
- VECTOR3D
- Visu Assert
- Visu CheckPropertyInfo
- Visu ClientTagData
- Visu ClientType
- Visu_DialogResult
- Visu FbClearEventsAfterStart
- Visu FbDialogLoginProvider
- Visu FbFileListProvider
- Visu FbStringDintMap
- Visu FbWebserver
- Visu FctCheckForLongFormatSpecifier
- Visu FctClosePAADialogIfNecessary
- Visu_FctGetDatasources
- Visu FctGetDevicePixeIRatioFromAdditionaIData
- Visu FctGetElementFromInputInfo
- Visu FctGetElementIndexFromID
- Visu FctGetInitFramePreviewFlags
- Visu FctGetSubElementFromIndex
- Visu FctHandleInputGesture
- Visu FctHandleVisulnputDialogTarget
- Visu FctHandleVisulnputMouseEvent
- Visu FctHandleVisulnputOverlayMeasureString
- Visu FctHandleVisulnputPAA
- Visu FctInitMemSet
- Visu FctlsEventTolgnoreWhileEditboxOpen
- Visu FctlsGestureEvent
- Visu_FctlsModaIDialogOpen
- Visu_FctlsOnStraightLine
- Visu FctlsRelevantGestureEvent
- Visu FctlsSelectionEmpty
- Visu FctLegacyIDStackInfoFill
- Visu FctLegacyIDStackInfoReadFromAdditionalData
- Visu FctRaiseMouseLeave
- Visu FctReleaseNonIECMemClientResources
- Visu FctSetDialogClientData
- Visu FctTransformSelectionIsotropicOverlay
- Visu HelpDumpLibHierarchy
- Visu OnlinechangeNotify
- Visu_ScalarTypesUnion
- Visu ScalarTypesWithPtr
- Visu SetCodegenFeatureSupport
- Visu StructCommandData
- Visu TypeString
- Visu UserPasswordCredentials
- VisuAlarmScrollValueProvider
- VisuAssignRtsBasedUserMgt
- VisuBenchmarkFBStatistics
- VisuBenchmarkNowInUs
- VisuChangePasswordProvider
- VisuClientAnimationData
- VisuClientObject
- VisuClientObjectClientSpecificData
- VisuClientObjectFlags
- VisuClientObjectFlags_HighWord
- VisuClientObjectldStack
- VisuClientObjectldStackOptimized
- VisuClientObjectldStackWithParentSize
- VisuClientObjectInputRectangleMgr
- VisuClientObjectLayerInitFlags
- VisuClientObjectMgr
- VisuClientObjectReservedlds
- VisuClientObjectStateFlags
- VisuClientTag
- VisuClientType
- VisuCreateUserMgmtObjects
- VisuDateTimeFormatPlaceholders
- VisuDialogOpenFlags
- VisuDIg CheckedAppend
- VisuDIg CheckedAppendExp
- VisuDlg CheckedAppendW
- VisuDlg CheckedBack
- VisuDIg CheckedBackW
- VisuDlg CheckedChangeNumberFormat
- VisuDlg CheckedChangeSign
- VisuDlg_CheckedChangeSignExtended
- VisuDlg CheckedClear
- VisuDlg CheckedExpFloatFormatSwitch
- VisuDIg GetPasswordString
- VisuDlg ProcessBack
- VisuDlg ProcessBack2
- VisuDlg ProcessBackPrepare
- VisuDlg_ProcessBackW
- VisuDlg_ProcessChangeNumberFormat
- VisuDlg ProcessChangeSign
- VisuDlg ProcessChangeSignExtended
- VisuDlg ProcessChar
- VisuDlg ProcessCharW
- VisuDlg ProcessCharW WithInit
- VisuDlg ProcessCharWithInit
- VisuDlg ProcessClear
- VisuDlg ProcessExp
- VisuDlg ProcessExpFloatFormatSwitch
- VisuDlg SetErrorColor
- VisuDIg_Utf8PasswordCaretPosition
- VisuDIg Utf8RuneLen
- VisuDIgUtil AdjustCaretAndSelectionAfterNumberFormatChange
- VisuDlgUtil CloseNumpadExtended
- VisuDlgUtil CloseNumpadExtended2
- VisuDlgUtil FormatExpToG
- VisuDIgUtil FormatValueToDec
- VisuDlgUtil_FormatValueToHex
- VisuDlgUtil InitValueFromFlags
- VisuDIgUtil IsDialogOpen
- VisuDIgUtil SetbInitFormatValue
- VisuDlgUtil UpdateOutputValue
- VisuElemLayer
- VisuElemLayerAlignmentFlag
- VisuElemLayerClientSpecificData
- VisuElemLayerData
- VisuElemLayerFlag
- VisuElemMgrClientData
- VisuElemMgrClientSpecificData
- VisuElemMgrClientSpecificDataIndices
- VisuElemSelectionLayer
- VisuEnumActionType
- VisuEnumAfterTransformation
- VisuEnumAlarmDataType
- VisuEnumAnalogClockStyle
- VisuEnumAttachedElementLocation
- VisuEnumBackgroundDrawingState
- VisuEnumBrushStyle
- VisuEnumClientTag
- VisuEnumCreateTemporaryRenderLocationFlags
- VisuEnumCurrentCheckboxState
- VisuEnumFileTransferDirection
- VisuEnumFileTransferError
- VisuEnumFileTransferInputEventType
- VisuEnumHorizontalAlignment
- VisuEnumInputOnElementType
- VisuEnumIsotropicType
- VisuEnumLegendDisplayerLineType
- VisuEnumLineCapStyle
- VisuEnumLineJoinStyle
- VisuEnumOrientation
- VisuEnumPenStyle
- VisuEnumPieType
- VisuEnumPolygonType
- VisuEnumPropertyElement
- VisuEnumRectangleFlags
- VisuEnumRedundancyValueChanged
- VisuEnumTableParts
- VisuEnumTextFlag
- VisuEnumVerticalAlignment
- VisuEnumXYChartActivityType
- VisuEnumXYChartAxisPosition
- VisuEnumXYChartAxType
- VisuEnumXYChartBarType
- VisuEnumXYChartBgType
- VisuEnumXYChartCommands
- VisuEnumXYChartCursorActive
- VisuEnumXYChartCursorType
- VisuEnumXYChartCursorVisible
- VisuEnumXYChartCvChartType
- VisuEnumXYChartCvFillType
- VisuEnumXYChartCvHeapCmd
- VisuEnumXYChartCvOverlapType
- VisuEnumXYChartFocusType
- VisuEnumXYChartGradientType
- VisuEnumXYChartGridType
- VisuEnumXYChartLineType
- VisuEnumXYChartLviLineLbPos
- VisuEnumXYChartPointStyle
- VisuEnumXYChartProgType
- VisuEnumXYChartShadowStyle
- VisuEnumXYChartZeroLineType
- VisuEventOptimization
- VisuEventTarget
- VisuFb3DPathClientData
- VisuFbAlarmBannerDataBlock
- VisuFbAnalyzeDateTimeFormatExtractWithoutWeekdays
- VisuFbAnalyzeDateTimeFormatStringBase
- VisuFbAnalyzeDateTimeFormatStringMinSecOnly
- VisuFbAnalyzeStateVarsTapAware
- VisuFbAnalyzeTextVarsDateTimeOnly
- VisuFbBaseVector
- VisuFbCapturedTransformationProvider
- VisuFbClientLogger
- VisuFbClientManagerListener
- VisuFbClientStartVisuMgr
- VisuFbClientTagDataHelper
- VisuFbCommandVector
- VisuFbDatasourcesResourceEntries MBM
- VisuFbDatasourcesResourceEntries SysMem
- VisuFbDateTimeNamesLocalizer
- VisuFbDialogClientInfo
- VisuFbDialogInfoVector
- VisuFbDialogPos
- VisuFbDialogPosByElementVector
- VisuFbDialogPosVector
- VisuFbDWORDVector
- VisuFbElemTextEditor
- VisuFbExecution
- VisuFbFileTransferManager
- VisuFbFrameRegistrationVector
- VisuFbFrameSwipingHelper
- VisuFbGestureFromEvent
- VisuFbGroupOverlay
- VisuFbInputRectangle
- VisuFbLegacyCapturingTransformationProvider
- VisuFbLibHierarchy
- VisuFbMainClientMgmt
- VisuFbMouseTouchDragUtil
- VisuFbMoveAbsoluteTapAware
- VisuFbMoveAbsoluteTapAwareF
- VisuFbMoveRelativeTapAware
- VisuFbMultiTouchHelper
- VisuFbNamespaceTable
- VisuFbNamespaceTableHelper
- VisuFbNativeControlltemVector
- VisuFbPaintAfterAllDialog
- VisuFbPaintAfterAllElement
- VisuFbPaintRectF
- VisuFbPointF
- VisuFbPrintDateTimeFormatBase
- VisuFbPrintDateTimeFormatCurrentTime
- VisuFbPrintDateTimeFormatVariable
- VisuFbRecordConditionInfo
- VisuFbRecordingRangeRingBuffer
- VisuFbRectangleListManager
- VisuFbRectF
- VisuFbResourcesEntryVector
- VisuFbScalingInfo
- VisuFbSizeF
- VisuFbTabControlOverlayTabs
- VisuFbTableColumnSimpleBase
- VisuFbTemporaryPolygon
- VisuFbTickMarkDrawer2
- VisuFbTransformationCommon
- VisuFbTransformationScrolling
- VisuFbValueChangedListenerForOnValueChanged
- VisuFbVisuVector
- VisuFbWriteDateTimeVariableFormatted
- VisuFbXYChartDataProvider
- VisuFbXYChartDataProviderAxis
- VisuFbXYChartDataProviderCurve
- VisuFbXYChartGenericVariable
- VisuFbXYChartGenericVariableArray
- VisuFct IsBehindOverlayElement
- VisuFctAddChecksumBool
- VisuFctAddChecksumForConverted
- VisuFctAddClientToEventQueue
- VisuFctAdjustElemRectAndSelectRect
- VisuFctAssignValue
- VisuFctCalculateCompleteSurroundingSimpleRectOfElemArray
- VisuFctCalculateMaxTooltipLength
- VisuFctCalculateSurroundingSimpleRectOfElemArray
- VisuFctCheckClientSupportsFramePreview
- VisuFctCheckClientSupportsTouch
- VisuFctCheckSpecificClientType
- VisuFctClearElementEntries
- VisuFctClearEventIdStack
- VisuFctConfigureTextBufferSize
- VisuFctConvertUtcToSystimedate
- VisuFctCreateEventMapIfNeeded
- VisuFctCreateldStack
- VisuFctDatasourcesResourceEntryAllocatorGet
- VisuFctDatasourcesResourceEntryAllocatorGet MBM
- VisuFctDatasourcesResourceEntryAllocatorGet_SysMem
- VisuFctDrawCircle
- VisuFctDrawDot
- VisuFctDrawDot2
- VisuFctDrawlmage
- VisuFctDrawlsoBackground
- VisuFctDrawLine
- VisuFctDrawLineEx
- VisuFctDrawLineExUntransformed
- VisuFctDrawLineUntransformed
- VisuFctDrawPolygon
- VisuFctDrawPolyline
- VisuFctDrawPolyline2
- VisuFctDrawText
- VisuFctEvaluatePanGesture
- VisuFctEventIdStackGetValue
- VisuFctEventIdStackGetValuePtr
- VisuFctEventIdStackGetValuePtrFromEvent2
- VisuFctEventIdStackGetValuePtrFromEventLegacy
- VisuFctEventIdStackHas
- VisuFctEventldStackPopHelp
- VisuFctEventldStackPopld
- VisuFctEventIdStackPopTarget
- VisuFctEventldStackPopVisuVersion
- VisuFctEventldStackPushld
- VisuFctEventIdStackSetValue
- VisuFctExitVisuClientObject
- VisuFctFillPolygon
- VisuFctFillPolygon2
- VisuFctFillPolygon3
- VisuFctFillRectangle
- VisuFctFreeClientTagData
- VisuFctGetAbsolutePosition
- VisuFctGetClientName
- VisuFctGetEffectiveTextProperties
- VisuFctGetElementClientData
- VisuFctGetElementEntry
- VisuFctGetElementState
- VisuFctGetGradient
- VisuFctGetlsNativeElementEventWithResult
- VisuFctGetLineJoinMiterLimit
- VisuFctGetMeasureString2Result
- VisuFctGetMeasureStringApprox
- VisuFctGetMeasureStringResult
- VisuFctGetMultitouchActive
- VisuFctGetMultitouchScrollbarsActive
- VisuFctGetNanosSince1970
- VisuFctGetPaintRectFromSimpleRect
- VisuFctGetRectangleFromPaintRect
- VisuFctGetRectangleFromSimpleRect
- VisuFctGetRectHeight
- VisuFctGetRectWidth
- VisuFctGetShadowColor
- VisuFctGetTargetVisuTouchFlags
- VisuFctGetTextMargins
- VisuFctGetTransparentValue
- VisuFctHandleInputOnElementEvent
- VisuFctHandleInputVisuClientObject
- VisuFctHandleInputWithoutInputHandler
- VisuFctIncreaseCurrentElementIndex
- VisuFctIncreaseSimpleRectIfRotated
- VisuFctInitFlagsVisuClientObject
- VisuFctInitVisuClientObject
- VisuFctlsDegenerateRectangle
- VisuFctlsMultitouchClient
- VisuFct/sRectangleRotated
- VisuFctlsRectRotatedBy0To90Degree
- VisuFctlsRectRotatedBy180Degree
- VisuFctlsRectRotatedBy180To270Degree
- VisuFctIsRectRotatedBy270Degree
- VisuFctlsRectRotatedBy270To360Degree
- VisuFctIsRectRotatedBy90Degree
- VisuFctlsRectRotatedBy90To180Degree
- VisuFctlsToPaintSelection
- VisuFctlsTransparentBackground
- VisuFctLimitSimpleRectangleSize
- VisuFctMainClientsCheck
- VisuFctMainClientsCheck3
- VisuFctMainClientsCheckOld
- VisuFctPaintSelection
- VisuFctPaintVisuClientObject
- VisuFctRectSize
- VisuFctRemoveClientFromEventQueue
- VisuFctSearchElementByld
- VisuFctSelectElement
- VisuFctSetClientDataVisuClientObject
- VisuFctSetMaxElementCountPaintAfterAll
- VisuFctSetNumericValue
- VisuFctSetRectangleUpdateNecessaryOnAllClients
- VisuFctSetSelectionChanged
- VisuFctSetSimpleRect
- VisuFctSimpleRectangleFToSimpleRectangle
- VisuFctSplitColor
- VisuFctTestLReal
- VisuFctTestReal
- VisuFctTextAddOffset
- VisuFctTextEditorGetErrorText
- VisuFctTryAtomicAssignValueBySize
- VisuFctTryAtomicAssignValueByType
- VisuFctUtf8RuneLen
- VisuFctWriteValueIfValid
- VisuGestureInfo
- Visulnput CheckUpdateElementStatePossible DependingOnCurrentInput
- VisulnputType
- VisulsValidClientld
- VisuLoginProvider
- VisuNativeElement
- VisuNativeElementMethod
- VisuNativeElementMethodParameterList
- VisuRegistrationHelpDuringDecl
- VisuScrollValueData
- VisuScrollValueProvider
- VisuSecureTappingKeyEvent
- VisuSizeProvider
- VisuStruct3DControl
- VisuStruct3DPathPoint
- VisuStruct3DTrack
- VisuStructAllDialogInfo
- VisuStructAIIModalDialogInfo
- VisuStructAllNonModalDialogInfo
- VisuStructBackgroundAndStaticElementDrawing
- VisuStructButtonClientSpecificData
- VisuStructClientSpecificInputRectangles
- VisuStructClientTagData
- VisuStructColors
- VisuStructComboboxTouchData
- VisuStructCompleteSurroundingRectInfo
- VisuStructComplexFrameClientSpecificData
- VisuStructDialogPosByElement
- VisuStructElemBaseClientSpecificData
- VisuStructElementClientData
- VisuStructElementClientDataExtended
- VisuStructElementState
- VisuStructFileTransfParams_Union
- VisuStructFindElementEvent
- VisuStructFlickInfo
- VisuStructFont
- VisuStructIECTouchInfo
- VisuStructInputInfo
- VisuStructInputOnElementEvent
- VisuStructLegendDisplayerCheckBoxPos
- VisuStructLegendDisplayerCheckBoxStatus
- VisuStructMethodClientSpecificData
- VisuStructNamespace
- VisuStructNamespaceProject/dent
- VisuStructNativeElementClientData
- VisuStructNativeElementClientSpecificData
- VisuStructPAADialogClientSpecificData
- VisuStructPanInfo
- VisuStructPoint
- VisuStructPointD
- VisuStructPolygonClientSpecificData
- VisuStructRadioButtonClientData
- VisuStructRadius
- VisuStructRecordingRange
- VisuStructRectangularElementUtilBaseClientSpecificData
- VisuStructScaleScrollInfo
- VisuStructSecureTappingKeyEvent
- VisuStructSimpleRectangleD
- VisuStructSimpleRectWithBorder
- VisuStructSingleIECTouchInfo
- VisuStructSpreadPinchInfo
- VisuStructTableColumnSimpleDynamicCellsData
- VisuStructTableColumnSimpleDynamicCellsDataBase
- VisuStructTableColumnTextfieldDynamicCellsData
- VisuStructTemporaryRenderLocationInfo
- VisuStructTextAndId
- VisuStructTextProperties
- VisuStructTopMostDialogInfo
- VisuStructTraceGradientColor
- VisuStructUpdateRectangle
- VisuStructWaitingCubeClientSpecificData
- VisuStructWaitingFlowerClientSpecificData
- VisuStructWinControlClientData
- VisuStructWinControlElementClientData
- VisuStructXYChart
- VisuStructXYChartAxis
- VisuStructXYChartCurve
- VisuStructXYChartGradientColor
- VisuStructXYChartLevelLine
- VisuStyleFct_GetImageAccordingMapping
- VisuTouchState
- VisuUserMgmt
- VisuUserMgmtCyclicCall
- VisuUserMgmtDialogManagerListener
- VisuUserMgmtHasUserRightToChangeUsers
- VisuUserMgmtProvider
- VPNControl
- VPNState
- VUM EditType
- VUM GroupSelection
- VUM ReturnValues
- VUM User
- VUM User2
- VUM User3
- VUM UserAdminAccess
- VUM UserGroup
- VUM UserInternal
- WARNING ID
- WCharToUpper
- WCONCAT
- WDELETE
- WEEK
- WEEKDAY
- WeekOfYear
- WFIND
- WINSERT
- WLEFT
- WLEN
- WMID
- WORD_AS_BIT
- WORD AS STRING
- WORD TO BCD
- WORD TO GRAY
- WORD TO HANDLE
- WORD TO IDENT
- WORD TO PVOID
- WorkerRegister
- WorkerUnregister
- WREPLACE
- WRIGHT
- Write
- WriteBootProject
- WriteMemory
- WriteRequest
- WRREC
- WStringElement
- WStringElementFactory
- WStringsEqual
- X509CertCheckHost
- X509CertCheckIP
- X509CertClose
- X509CertCmsDecrypt
- X509CertCmsVerify
- X509CertCreateCSR
- X509CertCreateSelfSigned
- X509CertGetBinary
- X509CertGetContent
- X509CertGetPrivateKey
- X509CertGetPublicKey
- X509CertGetThumbprint
- X509CertHasExtendedKeyUsage
- X509CertHasKeyUsage
- X509CertInfoExit
- X509CertInfoInit
- X509CertlsAuthority
- X509CertIsDateValid
- X509Cert/sSelfSigned
- X509CertKeyClose
- X509CertStoreAddCert
- X509CertStoreClose
- X509CertStoreGetFirstCert
- X509CertStoreGetNextCert
- X509CertStoreGetRegisteredCert
- X509CertStoreOpen
- X509CertStoreRegister
- X509CertStoreRemoveCert
- X509CertStoreSearchGetFirst
- X509CertStoreSearchGetNext
- X509CertStoreUnregister
- X509CertVerify
- X509CertVerify2
- X509ParseCertificate
- XChgClass
- XChgCreateH
- XChgCreateP
- XChgDelete
- XChgExtendH
- XChgGetSize
- XChglsEmpty
- XChgMsgLeft
- XWORD
- XwordVector
- YEAR
- ZCMC ABB AXIS TYPES
- zCMC_ABB_ERRORINFO
- zCMC_ADDW
- zCMC AXIS INTERNAL
- zCMC createDecelTable
- zCMC Cut40Bit
- zCMC DIFF
- zCMC DIFF MODULO32
- zCMC DIV LONG REST
- zCMC_FBPosiReal
- ZCMC_FBPOSIREAL_ZACTION_INIT
- zCMC FBSpeedReal
- zCMC INTERPOLATION BASE
- zCMC Kernel Base
- zCMC POLYNOM pos
- zCMC POLYNOM time
- zCMC_POSITION_IPO
- zCMC_POSITION_IPO_ORIG
- zCMC PosLoop
- zCMC PT1 LREAL
- zCMC SPLINE pos
- zCMC STOP IPO
- zLOG ENUM STATE


### 6.6 Engineering interfaces and tools

### 6.6.1 Export and import interfaces

### 6.6.1.1 Exporting and importing ECAD data (PBF)

### 6.6.1.1.1 Introduction

Automation Builder provides an ECAD interface for exchanging the PLC configuration data with EPLAN Electric P8 and Zuken E3. This feature removes double data entry between electrical engineering in the ECAD tool and the control logic programming in Automation Builder by synchronizing the PLC hardware including topology and I/O signals between these tools.
Automation Builder - ECAD interface supports various flexible workflows:

- Enables PLC hardware planning and configuration in the ECAD tool and allows importing the exported data from the ECAD tool through the PBF file (process integration bus interchange format) into the Automation Builder project with diff and merge functionality, providing full control on selective import/merge.
- Enables PLC hardware configuration in Automation Builder and allows exporting the configuration to the ECAD tool through a PBF file.
- Supports bi-directional roundtrip engineering with loss less data exchange between Automation Builder and the ECAD tool.

Automation Builder uses the rack information to identify the relations between:

- PLC and devices plugged to I/O bus or extension bus.
- Fieldbus slave and attached IO devices.

It is recommended to assign the PLC, IO devices, communication modules and fieldbus slaves properly to the rack in the ECAD project. If the rack information is missing, devices will be imported to the device pool and must be arranged manually in the Automation Builder project or mapped to already existing devices.

### 6.6.1.1.2 Requirements on EPLAN electric P8

- EPLAN Electric P8 with PLC and Bus Extension. It is recommended to use version 2.3 or later.
- Use of appropriate part data and macros for ABB devices. This can be achieved by getting the part data and macros from the EPLAN data portal.


### 6.6.1.1.3 Importing PLC data from the ECAD tool

You can create a new Automation Builder PLC project from the existing PLC hardware configuration in your ECAD tool, by importing the exported PBF file to Automation Builder.

Import PBF file
to Automation
Builder

1. From the main menu, select "Project $\rightarrow$ Import $\rightarrow$ ECAD (PBF)".
2. From the file system, select the $P B F$ file.

Automation Builder starts importing the devices and its associated signals from the PBF file. After a successful import, the result is displayed in the Project Compare -Differences view. You can now decide and selectively merge the differences.
\& Chapter 6.4.1.21.3.5.21 "Command 'Compare'" on page 2607
3. Select the DevicePool node and click "Accept Block" to accept the complete PLC structure in the ECAD tool.
4. Select the PLC node and click "Accept Block" to accept all child device nodes.


The DevicePool node holds all devices coming from the ECAD tool without any hierarchy information. The missing hierarchy information can be defined after closing the editor.
5. Close the Project Compare - Differences view to accept the changes.
6. Arrange unassigned devices in the DevicePool to the PLC hardware structure by drag-and-drop.

$\Rightarrow$ The I/O signals assigned to I/O devices in the PBF file are imported and allocated to IO devices. IO signals can be viewed in I/O mapping editor of the I/O devices.

### 6.6.1.1.4 Importing third party devices

Prerequisite: To import third party devices from ECAD to Automation Builder, install third party fieldbus devices (for example, GSD, GSDML and EDS files) using "Tools $\rightarrow$ Device repository" in Automation Builder.

1. From the main menu, select "Project $\rightarrow$ Import $\rightarrow$ ECAD (PBF)".
2. From the file system, select the ECAD pbf file which consists of third party devices.
$\Rightarrow$ When the device identifier of the third party device installed in Automation Builder does not match with the device identifier of the device imported from ECAD, an error window is shown with the devices which are failed to import with error identifier 14.
To import third party devices, it is required to assign ECAD identifier (PLC type designation/order number) in Automation Builder in "Tools $\rightarrow$ Device ECAD data". Click the link in the Import window to see the error messages in a text file.
3. Click "Continue" in the Import window to import valid devices to the project that are imported successfully or click "Cancel" to cancel the import process.
4. In Automation Builder, click "Tools $\rightarrow$ Device ECAD data".
5. In the Device ECAD data editor, add the ECAD identifier for the devices shown in the import errors window with error identifier 14, to enable these devices for export and import.

$\Rightarrow$ Also, add the ECAD identifiers for all devices which need to support export/import in ECAD.
6. Reimport the pbf file to import the third party devices.

### 6.6.1.1.5 Exporting PLC data to ECAD tool

1. Open the existing PLC project.
2. In the device tree, right-click "PLC $\rightarrow$ Export $\rightarrow$ ECAD (PBF)".
3. Select the desired location in the file system to save the $P B F$ file.

The ECAD user can import the exported PBF file from Automation Builder and can use the imported PLC data for electrical engineering purpose. If the user modifies imported PLC data in the ECAD project, the data can be imported back to the Automation Builder project which supports the round trip engineering efficiently with loss less synchronization of the data.

### 6.6.1.1.6 Exporting third party devices

1. Right-click on a PLC device, click "Export" and select "ECAD (PBF)".
2. Save the file to the desired location in the file system.

If the third party devices does not contain assigned ECAD identifiers, a message is displayed showing which devices cannot be exported.
$\Rightarrow$ To add ECAD identifiers to the devices, see Importing third party devices $\Leftrightarrow$ Chapter 6.6.1.1.4 "Importing third party devices" on page 4192.

After adding ECAD identifiers to the third party devices, execute "Export" to export the devices including third party devices.

### 6.6.1.1.7 Importing ECAD PLC data to existing $A B$ project

Automation Builder ECAD interface supports concurrent engineering by importing the ECAD data to the existing Automation Builder PLC project.

1. From the main menu, select "Project $\rightarrow$ Import $\rightarrow$ ECAD (PBF)".
2. Select the PBF file which has been created during the export from the ECAD tool.
3. Select the PLC from the list and click "OK".
$\Rightarrow$ A dialog window is displayed if the Automation Builder project provides PLCs of the identical type as defined in the PBF file.

By selecting "None" in the dialog window a new PLC is defined in the ECAD tool.
4. In the Project Compare - Differences view, click 直 to merge device signals.
$\Rightarrow$ The differences between the current PLC hardware configuration in Automation Builder and the ECAD PLC data are displayed.
5. Select the differences as desired and click "Accept Single" to accept the selected difference block.
6. Close the Project Compare - Differences view to accept the changes.

### 6.6.1.1.8 Arrange or map devices imported to the device pool

Devices that are imported to the device pool because of missing hierarchy information (mainly rack information) must be arranged manually in the Automation Builder project or mapped to already existing devices.

Arrange devices Arrange the unassigned devices in the DevicePool to the PLC hardware structure by drag-andimported to the drop.
device pool
Map devices If the devices are already added to the Automation Builder project prior to the import, you have imported to the to map the instances of the same type manually (one instance in the Automation Builder project device pool tree and one instance in the DevicePool).

After mapping the devices, you can selectively merge the device parameter or signal in the difference view.

To map pool devices, proceed as follows:

1. In the device tree, select the Device Pool node, click "Project" and select "Map pool devices".
2. Map the device pool instances of identical types in the project from the drop-down list and click "OK".
$\Rightarrow$ Pool devices which are mapped are removed from the device pool and mapped to the corresponding Automation Builder device. Differences between the signals of the mapped I/O devices are displayed. e.g. AI523_1 device:


### 6.6.1.1.9 Limitations

The following limitations are considered when working with the Automation Builder ECAD interface:

- The scope of a PBF file is limited to one single PLC including all connected devices.
- There is no representation of XC variants of devices in Automation Builder. Therefore, always use the standard variant for export. This might lead to part data mismatch when importing into the ECAD tool.
- In reimport or round trip import cases, if any changes are made in ECAD by adding a new communication module with connecting to one of the PLC slot or replacing existing communication module, then those device changes to the communication modules are not displayed as connected to PLC slots during the import in Automation Builder diff and import, instead those CM modules are added under the device pool. After merging and importing is completed, to work with device pool devices $\stackrel{y}{m}$ Chapter 6.6.1.1.8 "Arrange or map devices imported to the device pool" on page 4194.
- IO mapping data cannot be imported for IO devices plugged to an EtherCAT slave when they are imported individually to the device pool because of missing hierarchy information. After arranging the devices properly in the device tree, the import can be done again to import also the IO mapping data.


### 6.6.1.2 Exporting and importing I/O mapping (CSV)

The I/O module mappings of an Automation Builder project can be exported to CSV for bulk editing in MS Excel or other documentation purposes. I/O mappings can be exported at single I/O module level or at PLC level.

Further, the I/O module mappings can be imported with the option of displaying differences and merging each single changed or import all signals at once by overwriting existing I/O module signals.

### 6.6.1.2.1 Exporting IO mapping data to CSV

To export I/O mappings to a CSV signal list, proceed as follows:

1. In the device tree, right-click "PLC $\rightarrow$ Export $\rightarrow$ IO mapping (CSV)".
2. Save the IO mappings CSV to the desired location in the file system.

If the CSV signal list has been exported successfully, a success message is displayed.
The status of the export is shown in the dialog.
3. In the export dialog, click the link to open the exported IO mapping CSV file in MS Excel.

The template can only be opened if MS Excel is installed and configured to open .csv files.
4. In the IO mapping (CSV) file, change Variable and Description fields to edit I/O mappings.



Do not modify other field's data in IO mapping (CSV) file.

### 6.6.1.2.2 Importing I/O mapping data from CSV

To import an edited I/O mapping (CSV) file, proceed as follows:

1. From the main menu, select "Project $\boldsymbol{\rightarrow}$ Import $\rightarrow$ IO mapping (CSV) $\rightarrow$ Open".
2. A CSV signal list import dialog is displayed.
$\Rightarrow$ With "YES", all I/O mappings will be imported without difference view. With "NO", the difference view is displayed with the I/O mapping differences.
3. In the Project Compare - Differences view, click 直 to merge I/O mappings.
4. Select the signal row for which the difference is to be accepted. Select the Variable field and click "Accept Single" to merge the I/O mappings.
5. Close the Project compare - Differences view to accept the changes and merge the I/O mappings with the Automation Builder project.

### 6.6.1.3 Exporting and importing device list (CSV)

### 6.6.1.3.1 General

The Automation Builder project devices can be exported to CSV for bulk device renaming or adding device tag labels to devices in MS Excel or other documentation purposes. A devices export is only possible at PLC level.
Automation Builder provides importing devices in bulk based on device type, instance and hierarchy information provided in the CSV file.

### 6.6.1.3.2 Exporting device list to CSV

To export a CSV device list, proceed as follows:

1. In the device tree, right-click "PLC $\rightarrow$ Export $\rightarrow$ Device list (CSV)".
2. Select the desired location in the file system to save the Device list (CSV).

If the CSV device list is exported successfully, a success message is displayed.
3. In the Export dialog, click the link to open the exported CSV device list.

The exported CSV device list consists of all devices connected to the PLC that is exported. Each row represents a device with its device type and hierarchy information.


### 6.6.1.3.3 Creating CSV device list

To create the devices in CSV, use the device list template provided in Automation Builder.
$\triangleright \quad$ In the main menu, click "Tools $\rightarrow$ Create CSV Device list". $\Rightarrow$ The device list template is opened in the MS Excel.


In this file, add each device in a separate row with device information like Device Type (Order Num or Device Type Name) and instance details (name, tag) and hierarchy information (parent Device name, parent Device Tag, position). The mandatory information required to import CSV is only Device Type. All other fields are optional. After editing the device list CSV file, save it in the file system and close.

### 6.6.1.3.4 Importing a device list from CSV

To import devices from CSV in bulk, proceed as follows:

1. From the main menu, click "Project $\boldsymbol{\rightarrow}$ Import $\boldsymbol{\rightarrow}$ Device list (CSV)".
2. Select the device list CSV file from the file system and click "Open" in the Import dialog.

All devices that are defined in the CSV are imported. The Project Compare - Differences view displays the current project and the project that has been updated by the import file.
3. Select the desired devices and click "Accept Block" to accept all the devices and its child device nodes or "Accept Single" to accept only a single device.
4. After closing the Project Compare - Differences view, the devices are imported to the Automation Builder project.
$\Rightarrow$ The devices (except PLC) are placed under the device pool if the valid device hierarchy information is not provided in the CSV device list file. By drag-and-drop devices can be assigned to the desired PLC hardware structure ${ }_{c} \stackrel{y}{c}$ Chapter 6.6.1.1.8 "Arrange or map devices imported to the device pool" on page 4194.
If a device tag is provided for a device in CSV, it appears next to each device node in the device tree.

### 6.6.1.3.5 Renaming devices

To rename the devices, proceed as follows:

1. In the device tree, right-click "PLC $\rightarrow$ Export $\rightarrow$ Device list (CSV)".
2. Select the desired location from the file system to save the CSV device list.
3. Rename the device names in the column Device Name:

4. Click "Project $\rightarrow$ Import $\rightarrow$ Device list (CSV)".
5. Select the updated CSV file from the file system.

Open the Project Compare - Differences view. If only the device names have been changed in the CSV file, the difference view does not show the changes.

Device Name changes are not displayed as changes in the difference view.
6. Close the Project Compare - Differences view. The Renamed Devices dialog is displayed with the current name and the new name provided in the CSV file.
7. In the Rename Devices window, select the desired devices and click "OK". The device names are updated in the Automation Builder project.

### 6.6.2 CODESYS Security Agent

### 6.6.2.1 Integration in CODESYS Development System

At this time, you can configure and create certificates of the controller with the CODESYS Security Agent. You can then configure encrypted communication with the controller, as well as encrypt the boot application, download, and online change.

- $\quad$ Chapter 6.6.2.2 "Encrypted Communication with Devices via Controller Certificates" on page 4201
- $乡$ Chapter 6.6.2.3 "Encryption of the Boot Application, Download, and Online Change" on page 4202


### 6.6.2.2 Encrypted Communication with Devices via Controller Certificates

Encrypted and An application can be encrypted and signed in order to protect a running application in an signed applica- AC500 V3 PLC and to protect a configured project. How to set-up the user management, the tions communication and the boot application in order to prevent unauthorized access is explained in the application note AC500 V3 - Encrypt and sign your application.

Requirement: A digital signature for certificate exchange is configured. ③ Chapter 6.4.1.9.18 "Protecting an application" on page 1915

## No certificate use in live system

Self-signed certificates should never be used on production or public websites.
The certificates that are created in the following steps are self signed.

We assume that there is still no certificate on the controller that is intended for encrypted communication. In the following steps, you generate this kind of certificate and encrypt communication:

1. Configure the active path to the controller.
2. Open the "Security Screen" view by double-clicking the symbol in the status bar or by clicking "View $\rightarrow$ Security Screen". Select the "Devices" tab.
3. Click the button to refresh the list of available devices and their certificate stores.
4. Select the corresponding device on the left side.
$\Rightarrow$ On the right side, there is still no license listed for the "Encrypted communication" use case.
5. On the right side, select "Encrypted Communication" and click the ${ }_{4}{ }^{*}$ button to create a new certificate on the device.

Change the default key length to 4096. Otherwise an error occurs that is only visible in the log of the PLC.
$\Rightarrow$ The certificate is generated and listed in the table with its properties. The symbol before "Encrypted communication" now appears as such: $\mathbb{E}_{\Delta}$. The field in the "Valid until" column is highlighted in green because the remaining time is still at least twothirds of the entire validity period.
6. In this step, you activate encrypted communication with the controller.

Open the "Security Screen" view of CODESYS ("Users" tab). In the "Security Level" group, select the "Enforce encrypted communication" option.
$\Rightarrow$ As of this point, communication with all controllers is possible only as long as the certificate is valid on the controller and you have a key for it.
The connecting line between the development system, the gateway, and the controller is displayed in yellow on the "Communication Settings" tab of the device editor of the controller.
As an alternative to the "Enforce encrypted communication" option that was just described and which applies to all controllers, you can also encrypt communication with a specific controller only. To do this, open the "Communication" tab in the device editor of the controller. Click "Encrypted Communication" in the "Device" list box.
7. Now log back in again to the controller.
$\Rightarrow$ A dialog opens with the notification that the certificate of the controller is not signed by a trusted source. In addition, the dialog displays information about the certificate and prompts for you to install it as a trustworthy certificate in the local store in the "Controller Certificates" folder.
8. Confirm the dialog.
$\Rightarrow$ The certificate is installed in the local store and you are logged in to the controller.
In the future, communication with the controller will be encrypted automatically with this control certificate.
Note: When logging in to the controller, the expiration date of the certificate currently in use is checked. You get a warning if the remaining time is just one-third of the entire time or less. Then you can renew the certificate in time in the security screen.

See also

- 宏 Chapter 6.6.2.4.1 "View 'Security Screen' - 'Devices"" on page 4205
- «y Chapter 6.6.2.1 "Integration in CODESYS Development System" on page 4201


### 6.6.2.3 Encryption of the Boot Application, Download, and Online Change

Encrypted and An application can be encrypted and signed in order to protect a running application in an signed applications AC500 V3 PLC and to protect a configured project. How to set-up the user management, the communication and the boot application in order to prevent unauthorized access is explained in the application note $\underline{\text { AC500 V3-Encrypt and sign your application. }}$

Aim: You want to encrypt boot applications, downloads, and online changes with a certificate to make sure that the application on the controller cannot be exchanged at will. To do this, you need to download a corresponding certificate of the type "Encrypted Application" from the controller and install it to the "Windows Certificate Store" of your computer. This certificate is required for all development environments that need to make changes to the application on the controller. For example, if this application has to be downloaded from another computer, then the certificate also has to exist on this computer.
See also

- «y Chapter 6.6.2.1 "Integration in CODESYS Development System" on page 4201
- CODESYS Help: "Security", "Encryption", "Certificate"

Requirement: The active path to the controller is configured.
en the "Properties" dialog of the application.
2. Click the "Encryption" tab. Set "Encryption Technology" to "Encryption with certificates". $\Rightarrow$ The "Encryption Wizard" button is available in the "Certificates" field.
3. Click the "Encryption Wizard" button.
$\Rightarrow$ The "Encryption Wizard" dialog opens. The status is Not connected and under "Details" is Ready.
4. Click the "Start" button.
$\Rightarrow$ The wizard searches for suitable certificates on the controller. If necessary, the controller creates a new certificate which is registered in the Certificate Store of your computer.
NOTE: A certificate obtained this way is automatically accepted as trusted.
If a certificate for application encryption already exists on the controller, then it is used.
If a new certificate has to be created on the controller for your CODESYS, then the "Certificate Settings" dialog opens for configuring the key length for the private key and the validity period.
5. In the "Certificate Settings" dialog, click "OK" to confirm the default or edited values for key length and validity period.
$\Rightarrow$ CODESYS saves the values in the CODESYS options as the default for the next certificate configuration of this kind.

In the "Details" of the wizard, you see a description of the performed actions and the thumbprint of the recently created certificate.
6. When the status reaches "Wizard finished", close the wizard.
$\Rightarrow$ The new certificate is listed in the "Certificates" field of the properties dialog. In the "Certificate Store", it is listed under "Controller Certificates". In the "Security Screen" view, on the "Devices" tab, the certificate is displayed in the right window with the "Encrypted Application" information.
7. Confirm the "Properties" dialog of the application.
8. Open the "Security Screen" view.
$\Rightarrow$ On the "Project" tab, in the "Encryption of boot application, download and online change" group, the certificate is displayed with the "Encrypted Application" information.

Boot application, download, and online change are therefore encrypted and only possible as long as the configured certificate and signature are valid.

See also

- ⓨ Chapter 6.6.2.4.2 "Dialog 'Encryption Wizard"" on page 4207
- CODESYS Help: Dialog "Properties" "Encryption"
- CODESYS Help: "Security", "Encryption", "Certificate"


## Encrypting the boot application, download, and online change without the encryption wizard

Requirement: The active path to the controller is configured. There is still no certificate on the controller that is suitable and valid for encryption.

1. Open the "Security Screen" view by double-clicking the symbol in the status bar or by clicking "View $\rightarrow$ Security Screen". Open the "Devices" tab.
2. Click the $\downarrow$ "Refresh the list of available devices and their certificate stores" button.
3. Select the device listed on the left side.
4. Select "Encrypted Application" on the right side and click the "Create a new certificate on the device" button.
Change the default key length to 4096. Otherwise an error occurs that is only visible in the log of the PLC.
$\Rightarrow$ The certificate is created and listed in the table with the 圈 symbol.
5. Double-click the certificate entry.
$\Rightarrow$ The Windows "Certificate" default dialog opens.
6. Click the "Install certificate" button on the "General" tab.
$\Rightarrow$ The "Certificate Import Wizard" opens.
7. In the "Certificate Store" dialog, select the "Place all certificates in the following store" option and select the "Controller Certificates" folder for "Certificate Store".
$\Rightarrow$ The controller certificate is imported to the "Controller Certificates" directory and it is immediately available for the encryption of downloads, online changes, and boot applications.
8. Open the "Project" tab and double-click the application entry in the "Encryption of boot application, download and online change" group.
$\Rightarrow$ The "Properties" dialog of the application opens.
9. Click the "Encryption" tab and set "Encryption Technology" to "Encryption with certificates". Then click ${ }^{3}$. Note: If the "Enforce encryption of downloads, online changes and boot applications" option is selected in the "Security Screen", then "Encryption with certificates" is already preset.
10. In the "Certificate Selection" dialog, select the corresponding certificate from the "Controller Certificates" folder and click 回.
11. Click "OK" to confirm the dialog.
$\Rightarrow$ The certificate is displayed in the properties dialog.
12. As above when using the wizard, steps 7 and 8 .

Enforcing the encryption of boot applications, downloads, and online changes
$\triangleright \quad$ Open the "Users" tab in the "Security Screen". In the "Security level" group, select the
$\Rightarrow$ Only with a valid certificate is it possible to change the application on the controller.

See also

- CODESYS Help: "Security-Screen"


### 6.6.2.4 Reference, User Interface

6.6.2.4.1 View 'Security Screen' - 'Devices' ..... 4205
6.6.2.4.2 Dialog 'Encryption Wizard' ..... 4207

### 6.6.2.4.1 View 'Security Screen' - 'Devices'

Symbol: (3)
Function: The tab allows for the configuration and the transfer of controller certificates for encrypted communication with the controller.
Call: Menu bar: "View"

The "Devices" tab shows all PLC devices configured in the project and their certificate store. If the communication path to the controller is configured, then you see the certificates that are stored in memory. Here you can create and configure new certificates on the controller. If a certificate currently in use is about to expire, then you get a warning when you log in to the device. From there you can also switch directly to the "Security Screen" to renew the certificate.

| Left side: "Information" | Devices and certificate store <br> Shows the individual devices as expandable nodes, each with the controller- <br> specific certificate store below it. |
| :--- | :--- |
| Toolbar (left side) | ¢: Refresh the display <br> D: Download: Transfer the selected certificate to the PLC |

Right side:
"Information"

If the active path to the controller is set and a device node is selected, then every use case for controller certificates is displayed on the right side.

- "OPC UA Server": Encrypted communication over an OPC UA server
- "Encrypted Communication": Encrypted communication between the development system and the controller
- "Encrypted Communication": Encryption of the boot application
- "Web server": Encrypted communication with the web server

As long as a certificate is not available for one of these use cases, it is displayed with the ${ }_{\#}^{* *}$ symbol as "(not available)".
When a certificate store is selected on the left side, all certificates in it are displayed on the right side with the following information:
"Information": Use case (currently the controller component in question is displayed: for example "CmpSecureChannel".)
"Created for": Name of the computer for which the certificate was created (for example, "MyLocalPC")
"Created by": Name of the computer on which the certificate was created (for example, "MyLocalPC")
"Valid as of": Date (for example, "07/20/2017 15:09:29")
"Valid until": Date (example: "07/20/2022 00:00:00". Depending on the remaining time of the certificate, the highlight color of the field changes: green -> yellow (two-thirds expired) -> orange (nine-tenths expired) -> red (expired). Note: When logging in to the controller, you get a warning when two-thirds or more of the validity period have expired. Then you can renew the certificate here in the "Security Screen".
"Thumbprint": Hash value from specific properties of the certificate for purposes of identification (for example, "279e1a46b86bd636c8e6f19fd51c222469ec49a8")
This thumbprint can be used together with the Mqtt library. Refer to the Mqtt library documentation in the Library Manager.
Double-clicking a certificate entry opens the default Windows "Certificate" dialog. As a result, you can import a controller certificate into the Windows Certificate Store in the "Controller Certificates" folder, so that it is available for the encryption of boot applications, downloads, and online changes.
If multiple certificates are available for one use case, then the system follows the steps below to determine the certificate that is used:

- Certificate that was created directly by the user (currently not supported)
- Filtering of existing certificates by:
- 1. Subject (user of the certificate)
- 2. Key usage
- 3. Extended key usage
- 4. Valid time stamp
- Dividing of detected, valid certificates as "signed" and "self-signed"
- Filtering of signed certificates, and the self-signed certificates by the following criteria:
- 1. Longest validity period
- 2. Strongest key

|  | Drag\&Drop: Moving of the certificate to another certificate store of the same device <br> Double-clicking a certificate entry opens the default Windows dialog for displaying all certificate information. |
| :---: | :---: |
| Toolbar (right side) | Creates a new certificate for a specific use case <br> The "Certificate Settings" dialog opens for configuring the "Validity period" of the certificate and the "Key length" for the private key. Clicking "OK" saves the specified values in the CODESYS options. The values are reset at the next operation. <br> As long as the certificate is being created, ""(computing)"" is shown after the use case. You cannot cancel the creation operation, but you can close and continue working with the "Security Screen". <br> $\mathbf{x}$ : Delete the selected certificate. <br> 7: Upload and save the selected certificate to the local file system. <br> IIAB: Details about the selected certificate: Opens the "Certificate" dialog with the "General" tab, "Details" tab, and "Certification Path" tab. <br> Pind $_{n}$ Renew the selected certificate. Opens the "Certificate Settings" dialog to create an additional new certificate for a certificate that will expire soon, with the same purpose and specified key length. The predefined values in the dialog are adapted, if necessary, depending on the selected certificate. |

### 6.6.2.4.2 Dialog 'Encryption Wizard'

Function: The wizard makes sure that a certificate for the encryption of downloads, online changes, and boot applications is downloaded from the controller. If a valid certificate does not exist on the controller for this purpose, then the wizard makes sure that a certificate is created. Changes to the application on the controller (download, online change, boot application) are possible only when this certificate exists.
Call: "Properties" dialog of an application, "Encryption" tab, "Encryption with certificates" setting, "Encryption Wizard" button
Requirement: "Encryption Technology" is set to "Encryption with certificates".
\(\left.$$
\begin{array}{|l|l|}\hline \text { "Status" } & \begin{array}{l}\text { Statuses while the wizard:is in action: } \\
\text { - } \\
\text { "Not connected": The connection to the controller has not been established } \\
\text { yet or the device cannot be reached. } \\
\text { "Error connecting to the device": The network path to the controller has not } \\
\text { been set correctly. } \\
\text { "Connecting...": A connection to the controller is being established. } \\
\text { - "Processing request..." The wizard is checking for available certificates } \\
\text { and if necessary makes sure that the controller creates a new certificate. } \\
\text { The certificate downloaded from the controller is automatically classified as } \\
\text { "trusted" and registered in the Certificate Store of the computer. } \\
\text { "Wizard finished" }\end{array} \\
\hline \text { "Details" } & \begin{array}{l}\text { Description of the individual actions of the wizard with corresponding notices in } \\
\text { the case of failures }\end{array} \\
\hline \text { "Start" } & \begin{array}{l}\text { If the connection path to the controller is set correctly in the device editor, } \\
\text { then the wizard starts the necessary actions for encrypting downloads, online } \\
\text { changes, and boot applications with a certificate. } \\
\text { If an expired certificate exists, then a corresponding warning is displayed with }\end{array}
$$ <br>
a dialog prompt whether or not a new certificate should be created by the con- <br>
troller. When this is confirmed, the new certificate is created and loaded to the <br>
local Certificate Store. In this case, the existing boot application may not start <br>

anymore and must be created again with the new certificate.\end{array}\right\}\)| The "Certificate Settings" dialog opens when a new certificate is to be created |
| :--- |
| on the device. Here you configure the "Key length (bit)" and the "Validity period |
| (days)" for the certificate. |

See also

- ${ }^{\text {® }}$ Chapter 6.6.2.3 "Encryption of the Boot Application, Download, and Online Change" on page 4202
- CODESYS Help: "Security", "Encryption", "Certificate"


### 6.6.3 CODESYS Static Analysis

### 6.6.3.1 Introduction

Already when programming in CODESYS, CODESYS Static Analysis helps to write more readable code and to detect contradictory or unsupported settings. In particular, potential sources of error can be identified, such as test code or pointers that have not been checked for 0 before dereferencing. With specific checks, you can make sure that the code is portable. Example: The analysis should report the use of language resources for object orientation because the code is to run on platforms that do not support object orientation.
The analysis checks the source code of the CODESYS project and reports any deviations from certain coding rules, naming conventions, or permitted keywords and identifiers. CODESYS Static Analysis is based on the rule set defined in the PLCopen Coding Guidelines and extends it with additional test options.
You can display the detected deviations as errors or warnings in the message view before the project is downloaded to the target system. For errors that are reported by Static Analysis based on precompile information, there is support for an immediate error handling ("Quickfix").
You activate Static Analysis either explicitly by clicking "Build $\rightarrow$ Run Static Analysis", or you let it execute automatically at each code generation. You activate the automatic execution in the "Static Analysis" dialog of the projects settings. In this dialog, you also configure what is to be checked in detail. You can use pragma statements to exclude individual parts of the code from the check.
To evaluate the code quality, you can also display selected metrics that CODESYS Static Analysis detects in your code in a separate view. An example of this is the McCabe metric, which measures the cyclomatic complexity and indicates the number of execution paths that can be processed during code execution.

## NOTICE!

The analysis is performed only for the code of the applications in the current project. Libraries are not taken into consideration.

The CODESYS development system contains a light version of Static Analysis that is extended by CODESYS Static Analysis.

## Application example

The Pro Static Analysis tool can be used to check the source code during programming or for debugging purposes in order to identify possible weaknesses of the code. Usage and benefits of the Pro Static Analysis tool are described in the application example Pro Static Analysis tool.

## - Chapter 6.6.3.3.2.2 "Dialog 'Static Analysis Settings' - 'Settings'" on page 4217 ${ }^{4}$ Chapter 6.6.3.3.1 "Commands" on page 4212 <br> - ${ }^{\circ}$ Chapter 6.6.3.4.1 "Pragmas and Attributes" on page 4228

### 6.6.3.2 Configuring and Running Static Analysis

Using a basic sample project below, you will find the most important steps and options for configuring and running a static analysis.

Requirements: CODESYS Static Analysis is installed.

Sample project If you want to reproduce the example project, create a standard project and insert the POUs below the application in the device tree. Then configure the communication settings for the connection to your local CODESYS Control Win V3.

```
FUNCTION BLOCK fb1
VAR_INPUT
    iVar fblin1 : INT;
    ivar fblin2 : INT;
    rVar_fblin3 : REAL;
END VAR
VAR_OUTPUT
    iVar_fb1out:INT;
END VAR
VAR }\mp@subsup{}{}{-
    P_fSampleProperty : INT;
        rVar : REAL;
            PRO : BOOL;
END_VAR
iVar_fb1out:=iVar_fb1in1 + 1;
FUNCTION_BLOCK fb2
VAR_INPUT
        iVar fb2in:INT;
END_VAR
VAR_OUTPUT
        iVar_fb2out:INT;
END_VAR
VAR
END_VAR
```


## Checking for compliance to rules

```
PROGRAM PLC_PRG
VAR
    fb1_inst: fb1;
    fb2-inst: fb2;
END VAR
fb1-inst(iVar fb1in1 := 99);
fb2-inst(iVar_fb2in := 22);
fb2_inst(iVar_fb2in := 1);
```

1. Click "Build $\rightarrow$ Static Analysis $\rightarrow$ Settings". Switch to the "Rules" tab.
$\Rightarrow$ A list is displayed containing all possible rule checks. They are organized in a tree structure by topical category. The rule number is added in parentheses (for example, "Unused variables (33)" in category "Unused objects").
2. Click the check box of the first line a few times ("Rules" node).
$\Rightarrow$ Clicking toggles the activation status. The check boxes in the entire tree have a red or orange check mark, or no check mark at all.
3. In this way, activate all entries with a red check mark. This means that CODESYS Static Analysis should report any detected rule violations as errors.
4. Click "Build $\rightarrow$ Static Analysis $\rightarrow$ Run Static Analysis".
$\Rightarrow$ Errors are reported in the message view. The message texts are tagged with a § and begin with the error number "SA<rule number>".
5. Double-click the first message SA0033: Unused variables 'iVar_fb2out'.
$\Rightarrow$ The focus moves to the declaration part of function block fb 2 and the relevant variable is selected. The variable is declared, but not used. This is checked in Rule 33 ("Unused variables"). In the code, the relevant locations are underlined with a wavy line.
6. To test the automatic execution of the analysis, click "Build $\rightarrow$ Static Analysis $\rightarrow$ Settings". On the "Settings" tab, select the "Perform static analysis automatically" option. Click "OK" to exit the dialog.
7. Click "Online $\rightarrow$ Login".
$\Rightarrow$ A dialog prompt indicates that compile errors exist. The errors reported by the code analysis are displayed again in the message view.
8. Click "Build $\rightarrow$ Static Analysis $\rightarrow$ Settings". Switch to the "Rules" tab.Now clear all of the rules in the dialog. In the "Unused Objects" category, explicitly activate Rule SA0035 ("Unused input variables (35)") with an orange-colored check mark to report a warning. See the tooltip for the rule text: "This rule corresponds to the following PLCopen rules: CP24"). Click "OK" to exit the dialog.
In the project settings, click "OK".
9. Click "Build $\rightarrow$ Generate Code".
$\Rightarrow$ The analysis is performed automatically. Two errors are reported in the message view: § SA0035: Unused input variable 'iVar_fb1in2 and § SA0035: Unused input variable 'iVar_fb1in3.
10. Double-click the message and comment or remove the declaration. Perform the code analysis again.
$\Rightarrow$ No error messages are displayed.

## See also

- $\boldsymbol{y y}^{2}$ Chapter 6.6.3.3.2.2 "Dialog 'Static Analysis Settings' - 'Settings'" on page 4217
- $\quad$ Chapter 6.6.3.3.2.3 "Dialog 'Static Analysis Settings' - 'Rules'" on page 4218


## Checking for compliance to defined naming conventions

1. Click "Build $\rightarrow$ Static Analysis $\rightarrow$ Settings". Click the "Naming Conventions" tab.
$\Rightarrow$ You see a table in a tree structure that is divided into expandable categories of variables and program blocks.
2. Expand the "Prefixes for Variables" - "Prefixes for Types" category, and in the "Prefix" column, specify I for "INT (14)".

Expand the "Prefixes for POUs" - "Prefixes for POU Type" category: In the "Prefix"column, specify the prog for "PROGRAM (122)" and fb for "FUNCTIONBLOCK (103)".
3. Select the "First character after prefix should be an upper case letter" option. Clear all other options.
4. Click "Build $\rightarrow$ Static Analysis $\rightarrow$ Run Static Analysis".
$\Rightarrow$ Error messages:

- NC0102: Invalid name 'PLC_PRG': Expect prefix 'prog'because PLC_PRG does not have the required prefix
- First character after prefix should be uppercase: 'ivar_fb1in2' because ivar_fb1in2 : INT; in fb1
- NCOO14: Invalid variable name P_fSampleProperty: Expect prefix 'i' because this integer variablē does not have the required prefix

See also

- 号 Chapter 6.6.3.3.2.4 "Dialog 'Static Analysis Settings' - 'Naming Conventions'" on page 4219


## Checking for forbidden symbols

1. Click "Build $\rightarrow$ Static Analysis $\rightarrow$ Settings". Click the "Forbidden Symbols" tab.
$\Rightarrow$ A line editor allows for specifying character strings that should not to be used in the code.
2. As an example, double-click the blank line and type in the invalid character string PRO directly. Double-click the next blank line and click ... to open the input assistance. From "Standard Types", select "REAL". Click "OK" to exit the dialog.
3. Click "Build $\rightarrow$ Static Analysis $\rightarrow$ Run Static Analysis".
$\Rightarrow$ The error messages Forbidden symbol 'REAL' and Forbidden symbol ' PRO' are displayed in the message view. Double-click the message text to jump to the relevant line of code.

See also

- $\quad$ y Chapter 6.6.3.3.2.6 "Dialog 'Static Analysis Settings' - 'Forbidden Symbols'" on page 4227


## Displaying of CODESYS Static Analysis performs selected tests on the code, and you can display the results metrics in a view.

1. Click "Build $\rightarrow$ Static Analysis $\rightarrow$ Settings". Click the "Metrics" tab. $\Rightarrow$ The metrics that CODESYS Static Analysis applies to the code are listed in a table.
2. For this example, activate the "Number of inputs variables" metric and specify the permitted range of values: lower limit 1 and upper limit 2. Activate some more metrics, for example "Code size" and "Number of calls".
3. Click "Build $\rightarrow$ Static Analysis $\rightarrow$ View Standard Metrics".
$\Rightarrow$ The view includes a table with a line for each "Program unit" of the sample program. For each activated metric, there is a column showing the measured values. Values that are outside of the range of values defined in the settings are highlighted in red. In the case of this specific example, this is at least the "PLC_PRG/Inputs" field because the number of input variables in this POU is greater than the defined upper limit of 2 .

See also

- ${ }^{4}$ Chapter 6.6.3.3.2.5 "Dialog 'Static Analysis Settings' - 'Metrics'" on page 4226

See also

- ${ }^{\circ}$ Chapter 6.6.3.3.1 "Commands" on page 4212


### 6.6.3.3 Reference, User Interface

$$
\text { 6.6.3.3.1 Commands........................................................................................ } 4212
$$

6.6.3.3.2 Dialogs............................................................................................. 4217
6.6.3.3.1 Commands

$$
\text { 6.6.3.3.1.1 Command 'Settings'........................................................................ } 4212
$$

6.6.3.3.1.2 Command 'Run Static Analysis'.................................................... 4212
6.6.3.3.1.3 Command 'View Standard-Metrics'.............................................. 4213
6.6.3.3.1.4 Command 'Extract function'.......................................................... 4215
6.6.3.3.1.5 Command 'Detect clones'............................................................. 4216

## Command 'Settings'

Function: The command opens the "Static Analysis Settings" dialog.
Call: Menu bar: "Build $\rightarrow$ Static Analysis"

## Requirement:

- The CODESYS Static Analysis package is installed.
- A project is open.

See also

- ⓨ Chapter 6.6.3.3.2.2 "Dialog 'Static Analysis Settings' - 'Settings'" on page 4217


## Command 'Run Static Analysis'

Symbol: §
Function: The command starts the static analysis for the active application and displays the results in the message view.

## Call: Menu bar: "Build $\rightarrow$ Static Analysis"

During the code analysis, CODESYS generates code just like the "Build $\rightarrow$ Generate Code" command. The results of the analysis are displayed as errors § and warnings § in the message view ("Build" category). The numbers refer to the corresponding rules as they are defined in the project settings. The syntax for the displayed messages is "SA<rule number:<rule text>".

See also

- ⓨ Chapter 6.6.3.4.2 "Rules" on page 4233


## Command 'View Standard-Metrics'

Symbol:
Function: The command starts the static analysis for the active application and displays the metrics for all POUs in a table.

Call: Menu bar: "Build $\rightarrow$ View Standard Metrics"
The metrics (code numbers) to be displayed are activated in the project settings. You can access the configuration by clicking "Configure" in the context menu of the displayed table. If a value is outside of the configured upper and lower limits, then the field in the table is highlighted in red.

See also

- ${ }^{4}$ Chapter 6.6.3.3.2.5 "Dialog 'Static Analysis Settings' - 'Metrics'" on page 4226


## Standard Metrics

| Metric | Description |
| :--- | :--- |
| "Code size" | Number of bytes |
| "Variable size" | Number of bytes |
| "Stack size" | Number of bytes |
| "Calls" | Number of calls |
| "Tasks" | Number of calls from tasks |
| "Global" | Number of different global variables |
| "I/Os" | Number of direct object accesses |
| "Local" | Number of local variables |
| "Inputs" | Number of input variables |
| "Outputs" | Number of output variables |
| "NOS" | Number of statements |
| "Comments" | Percentage of comments |
| "McGabe" | McGabe complexity |
| "Prather" | Prather complexity of nesting |
| "DIT" | Depth of inheritance tree |
| "NOC" | Number of children |
| "RFC" | Response for class |
| "Elshof" | Elshof complexity of reference |
| "CBO" | Coupling between objects |
| "LCOM" | Lack of cohesion in methods |
| "n1 (Halstead)" | Number of different used Halstead (n1) opera- <br> tors |


| Metric | Description |
| :--- | :--- |
| "N1 (Halstead)" | Number of Halstead (N1) operators |
| "n2 (Halstead)" | Number of different used Halstead (n2) oper- <br> ands |
| "N2 (Halstead)" | Number of operands (N2) |
| "HL (Halstead)" | Halstead length (HL) |
| "HV (Halstead)" | Halstead volume (HV) |
| "D (Halstead)" | Halstead difficulty (D) |
| "SFC branches" | Number of SFC branches |
| "SFC steps" | Number of SFC steps |

## Commands in the context menu

The following commands are provided in the context menu of the table:

- "Calculate": The values are refreshed.
- "Copy Table": The table is copied to the clipboard. The separators are tabs.
- "Print Table": The default dialog for setting up a print job opens.
- "Export Table": The table is exported as a CSV file. The separators are semicolons.
- "Kiviat Diagram": Requirement: At least three metrics have defined upper and lower limits. A radar chart is created for the selected POU. This visualizes the quality of POU code with respect to a given standard.
Each metric is depicted as an axis with its origin at the center (value 0 ) which radiates outward into three concentric ring zones. The inner ring zone represents the range of values below the lower limit defined for the metric. The outer ring represents the range of values above the upper limit. The axes of the metrics are distributed uniformly around the circle. The current values of the individual metrics on the axes are connected by a line. In the ideal case, the complete line is located in the middle zone.
- "Configure": The table for selecting the desired metrics opens. This corresponds to the table in the project settings.
- "Open POU": The POU opens in the editor.


See also

- $\quad \Longleftrightarrow$ Chapter 6.6.3.3.2.5 "Dialog 'Static Analysis Settings' - 'Metrics'" on page 4226


## Command 'Extract function'

Function: The command opens the "Extract Function Configuration" dialog.
The command extracts selected code from the ST editor and creates a new method or function containing this code. The affected code in the ST editor is replaced by a correct call. When code is extracted from a function block or the child of a function block, a new method is created from the code. When code is extracted from a program or a function, a new function is created from the code.

## Call: Context menu: "Refactoring"

Requirements: When the selected code consists of one or more statements:

- The selected code does not contain any compile errors.
- The selected code is located in the implementation part of an ST POU.
- The selected code does not contain any exiting jumps

Examples of exiting jumps include the following:

- Using RETURN to exit the enclosing function
- Using CONTINUE or EXIT to exit a loop enclosing the code

[^52]Table 770：Dialog＂Extract Function Configuration＂

\begin{tabular}{|c|c|}
\hline ＂Name＂ \& Name of the recently created function or method The default name can be changed． <br>
\hline ＂Return value＂ \& Determines the return value of a function if there are multiple output and／or input／ output parameters <br>

\hline ＂Parameter＂ \& \begin{tabular}{l}
Display of the available POUs <br>
Configuration whether the parameters are used as input，output，or input／output variables
$\square$ ：Input variables
$\square$
$\square$ Output variables
$\square$

$\square$
$\square$ ：Input／Output variables
\end{tabular} <br>

\hline 6 \& The changes made for＂Name＂，＂Return value＂，or＂Parameter＂are undone． <br>
\hline Upper code window \& Recently created code of the call location <br>
\hline Lower code window \& Recently created code of the function or method <br>
\hline ＂OK＂ \& The displayed code changes are accepted in the ST POUs and the dialog is closed． <br>
\hline ＂Cancel＂ \& The displayed code changes are rejected and the dialog is closed． <br>
\hline
\end{tabular}

## Command＇Detect clones＇

Function：The command scans the program code of the open CODESYS project for copied code，and opens the＂Clone detection results＂view to display the detected cloned code blocks． In the process，only code blocks larger than a specific size are considered to be clones．Very small chunks of code are not displayed as clones．

## Call：

－Menu bar：＂Build $\rightarrow$ Static Analysis＂
－Context menu：＂Static Analysis＂
Requirement：The CODESYS project is open．
Two code positions are considered clones if they have the following properties：
－Same structural composition
－Variables have the same data type．
－Variable names may be different（exception：component access）．However，an identifier that is contained multiple times in the code has to be in the same place in both code positions．
－Literals have the same data type．
－Literals may be different．A literal that occurs multiple times in the code has to occur at the same place in both code positions．

Table 771：View＂Clone detection results＂

| •＂Summary＂ | Tab to display the search results <br> $\bullet \quad$＂Number of found cloned code sequences＂ <br> $\bullet$＂Number of statements compared＂ |
| :--- | :--- |
|  | $\bullet$＂Number of statements in cloned code＂ <br> $\bullet$＂Clone ratio＂：Specified as a percentage：＂Number of statements in cloned <br> code＂／＂Number of statements compared＂ |
|  |  |


| Q "Results" | The tab displays the code clones in a tree view and provides commands and filter options. <br> The first occurrence of a duplicate from the set of duplicates is taken as the root node. The background color of the child nodes indicates whether the code is different or completely identical. Same colors mean the "same code". The contents of the tree view are sorted in descending order by the number of statements of the duplicated code. |
| :---: | :---: |
| Commands and filters on the $q$ "Results" tab |  |
| "Subnodes/Clone" | Number of subnodes (statements) in the code block If the number of subnodes is less than 20, then the code clone is not considered. |
| "Filter on Object" | Input field for an "Object", by which the clone list is filtered |
| "Show selected clones" | Requirement: Two child nodes of the same parent node are selected. <br> Both programming objects are displayed in the upper part of the view for comparison. In the process, the code duplicates are highlighted and differences (for example, different variable names) are highlighted in a different color. |
| List of code clones | Columns <br> - "Description" <br> - "Subnodes/Clone" <br> - "Object" <br> - "Position" <br> Double-clicking a child node opens the corresponding programming object, and the duplicated code block is selected there. |

### 6.6.3.3.2 Dialogs

6.6.3.3.2.1 General.................................................................................. 4217
6.6.3.3.2.2 Dialog 'Static Analysis Settings' - 'Settings'................................ 4217
6.6.3.3.2.3 Dialog 'Static Analysis Settings' - 'Rules'.................................... 4218
6.6.3.3.2.4 Dialog 'Static Analysis Settings' - 'Naming Conventions'.............. 4219
6.6.3.3.2.5 Dialog 'Static Analysis Settings' - 'Metrics'................................. 4226
6.6.3.3.2.6 Dialog 'Static Analysis Settings' - 'Forbidden Symbols'............... 4227

## General

For the dialogs for the configuration of static code analysis, click "Build $\rightarrow$ Static Analysis
$\rightarrow$ Settings". Requirement: A CODESYS project is open.

## Dialog 'Static Analysis Settings' - 'Settings'

Function: In the dialog, you select automatic static analysis, and save or load the project settings for static analysis as a CSA file.

## Call:

- Menu bar: "Project $\boldsymbol{\rightarrow}$ Project Settings", "Static Analysis" category, "Open configuration dialog" link
- Menu bar: "Build $\rightarrow$ Static analysis $\boldsymbol{\rightarrow}$ Settings"


## Requirement:

- The CODESYS Static Analysis package is installed.
- A project is open.

| "Perform static analysis | $\square:$ CODESYS performs the code check automatically at each code generation <br> automatically" <br> (for example, when the "Build $\rightarrow$ Generate Code" command is executed or <br> before a download. <br> $\square:$ The code check is not performed automatically, but it can be performed <br> explicitly by means of the "Build $\rightarrow$ Static Analysis $\rightarrow$ Run Static Analysis" <br> command. |
| :--- | :--- |
| "Load" | Opens the "Load Static Analysis Configuration" dialog for selecting the project <br> settings for the static analysis as a CSA file in the file system. When you click the <br> "Open" button, the selected CSA file is loaded. |
| "Save" | Opens the "Save Static Analysis Configuration" dialog for saving all project set- <br> tings in the "Static Analysis" category as a CSA file in the file system. |

See also

- ⓨ Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209


## Dialog 'Static Analysis Settings' - 'Rules'

Function: In the dialog, you select the rules that are checked during the static analysis of the source code of a project.

## Call:

- Menu bar: "Project $\rightarrow$ Project Settings", "Static Analysis" category, "Open configuration dialog" link
- Menu bar: "Build $\rightarrow$ Static analysis $\rightarrow$ Settings"


## Requirement:

- The CODESYS Static Analysis package is installed.
- A project is open.

This tab shows a tree structure of all rules that can be checked during static analysis. By default, every rule is activated, with the exception of SA0016, SA0024, SA0073, SA0101, SA0105, SA0106, SA0133, SA0134, SA0150, SA0162, and all strict IEC rules.

Each rule has a unique number. When the rule is checked and a violation is detected, the rule number and an error description are shown in the message view in the "Build" category in the following format: SA<rule number>, where SA stands for "Static Analysis" (example: "SA003" for rule 3).


The list of available rules can be extended by specific plug-ins.

Some rules that are activated in the dialog can be deactivated temporarily in the application by applying a pragma.
When you click the check box, the setting toggles between $\square, \square$, and $\square$.
When you activate or deactivate a parent node, all child rules are also activated or deactivated, respectively.
"Filter":

- Input field for the strings to be searched for
- ${ }^{\text {© }}=$ : Rules are grouped by category.
- "Structured by Importance": Sorting by "Importance High", "Importance Medium", and "Importance Low"
- "Default": Default structuring of the rules in CODESYS Static Analysis
- $\bar{\equiv}$ : Rules are displayed as a flat list. By clicking on the corresponding column header, the list can be sorted by rule number, activation/deactivation, rulespecific configuration, or importance.

| Columns |  |
| :---: | :---: |
| "Rules" | List of rules with rule number |
| Rule check | - $\square$ : The rule is not checked. <br> - 回: If the result of the check is positive, then an error ( $\S$ ) for the static analysis is displayed in the message view. <br> - $\square$ : If the result of the check is positive, then a warning ( $\$$ ) for the static analysis is displayed in the message view. |
| "Precompile" | Rules which can be checked during precompile are identified by a check mark $\qquad$ in this column. <br> An immediate bugfix (Quickfix) is possible for these rules. You can execute an automatic, immediate error handling directly at the affected code positions. |
| "Rule specific configuration" | For some rules, you can double-click the field to open a rule-specific dialog to configure the rule. |
| "Importance": | Importance of the rule: <br> - 3 red stars: High <br> - 2 orange stars: Medium <br> - 1 gray star: Low |

See also

- "A Chapter 6.6.3.4.2 "Rules" on page 4233
- « Chapter 6.6.3.4.1 "Pragmas and Attributes" on page 4228
- «4 "Checking for compliance to rules" on page 4210


## Dialog 'Static Analysis Settings' - 'Naming Conventions'

Function: In the dialog, you define the prefixes for the data types and scopes of variables, as well as prefixes for POUs and user-defined data types (DUTs). Static analysis checks compliance with the naming conventions. When a convention is not observed, the static analysis reports an error message in the "Messages" view.

## Call:

- Menu bar: "Project $\boldsymbol{\rightarrow}$ Project Settings", "Static Analysis" category, "Open configuration dialog" link
- Menu bar: "Build $\rightarrow$ Static analysis $\boldsymbol{\rightarrow}$ Settings"


## Requirement:

- The CODESYS Static Analysis package is installed.
- A project is open.

The error messages are displayed in the following format: § NC <prefix convention number> : <message text>. NC stands for "Naming Convention". For example, the error message "§"NC0102: Invalid name...'" means a violation of naming convention 102 for POUs of type PROGRAM.

You can use the pragma 'naming' to deactivate naming conventions for individual identifiers. The identifiers can begin with anything, not necessarily with the prefix.

[^53]| Table with the naming conve |  |
| :---: | :---: |
| "Names" | Nodes and elements for which a prefix can be defined. <br> The number in parentheses after each element (for example, "PROGRAM (102)") is the prefix convention number that is reported in the case of noncompliance with a naming convention. |
| "Prefix" | Input field of the prefix <br> - Multiple prefixes can be specified by means of comma separation. <br> Example: <br> "Prefix for POUs", PROGRAM (102): prog, PRG_ <br> "Prefix for POUs", FUNCTION (103): fun, FUN_ <br> - Regular expressions (RegEx) are also possible for prefixes. To do this, an @ has to be prepended. <br> Example: <br> The name has to begin with x and may contain one character from the scope $a-d A-D: ~ @ x[a-d A-D]$. <br> - For variables of type "Alias" and POUs of type "Property", the prefix can be defined with the placeholder \{datatype\}. |
| "Prefixes for variables" | Organizational node for all variables for which a prefix can be defined dependent on data type or scope. |
| "Prefixes for POUs" | Organizational node for all POU types and method scopes for which a prefix can be defined |
| "Prefixes for DUTs" | Organizational node for the DUT data types (structure, enumeration, alias, or union) for which a prefix can be defined |
| "Prefixes for custom types" | Organizational node for special custom types (particularly those from libraries) You can extend the list with conventions: Click the blank space below it. In the "Input Assistant" dialog, specify the name of a custom type or select a custom type. <br> To delete a convention, select it and press the [Del] key. <br> Note: These conventions have priority over the prefixes which are defined with the attribute \{attribute 'nameprefix' := '<prefix>'\}. |


| Options |  |
| :--- | :--- |
| "First character after prefix |  |
| should be an upper case letter" | variable name after the defined prefix is not an uppercase letter. |


| "Combine scope prefix with data type prefix" | As its namespace, a variable must have the defined prefix followed by the defined prefix for its data type. <br> Example: The following prefixes are defined: g for "VAR_GLOBAL", and r for the data type "REAL". The code analysis reports errors for global REAL variables that do not have the prefix g_r. $\square$ If conventions for the namespace are specified for a variable, then these conventions are taken into account. As a result, any data type conventions are ignored. <br> Example: The following prefixes are defined: g _ for "VAR_GLOBAL", and r for the data type " $R E A L$ ". The code analysis reports exclusively errors for global REAL variables that do not have the prefix $g_{-}$. |
| :---: | :---: |
| "Recursive prefixes for combinable data types" | Variables of combined data types have to have compound prefixes that follow the defined naming conventions. <br> Example: <br> ppiVariable : POINTER TO POINTER TO INT; <br> The prefix $p$ was defined for variables of data type POINTER, and the prefix I was defined for the data type INT. Static analysis reports errors for all variables of type POINTER TO POINTER TO INT which do not have the prefix ppi. <br> refaiVar : REFERENCE TO ARRAY[1..3] OF INT; <br> The prefix ref was defined for the data type REFERENCE TO, the prefix a for an array, and the prefix I for the data type INT. Static analysis reports errors for all variables of type REFERENCE TO ARRAY[1..3] OF INT which do not have the prefix refai. |

Example The following naming convention corresponds for the most part to the recommendations described in the "Identifiers" chapter.

| Names | Prefix |
| :---: | :---: |
| - Prefixes for variables |  |
| -- Prefixes for types |  |
| BOOL (3) | x |
| BIT (4) | bit |
| BYTE (5) | by |
| WORD (6) | w |
| DWORD (7) | dw |
| LWORD (8) | Iw |
| SINT (13) | si |
| - INT (14) | i, n |
| - DINT (15) | di |
| - LINT (16) | li |
| USINT (9) | usi |
| - UINT (10) | ui |
| - UDINT (11) | udi |
| $\cdots$ ULINT (12) | uli |
| REAL (17) | r |
| LREAL (18) | Ir |
| STRING (19) | s |
| WSTRING (20) | ws |
| - TIME (21) | tim |
| - LTIME (22) | Itim |
| DATE (23) | dat |
| DATE_AND_TIME (24) | dt |
| TIME_OF_DAY (25) | tod |
| POINTER (26) | p |
| REFERENCE (27) | ref |
| SUBRANGE (28) | range |
| ARRAY (30) | a |
| Function block instance (31) | fb |
| Interface (36) | iff |
| Structure (32) | struct |
| ENUM (29) | e |
| - Alias (33) | \{datatype |
| Variables with data type UNION (34) |  |
| - _ XWORD (35) | xw |
| -_- UXINT (37) | xui |
| -..._XINT (38) | xi |
| $\pm$ Prefixes for scopes |  |
| $\pm$ Prefixes for POUs |  |
| $\pm$ Prefixes for DUTs |  |



| Names | Prefix |
| :---: | :---: |
| $\pm$ Prefixes for variables |  |
| - Prefixes for POUs |  |
| - Prefixes for POU type |  |
| PROGRAM (102) | PRG_ |
| FUNCTIONBLOCK (103) | FB_ |
| FUNCTION (104) | fun_ |
| METHOD (105) | meth_ |
| ACTION (106) | act_ |
| PROPERTY (107) | prop_\{datatype\} |
| - INTERFACE (108) | ITF_ |
| - Method scope |  |
| PRIVATE (121) |  |
| - PROTECTED (122) |  |
| - INTERNAL (123) |  |
| - PUBLIC (124) |  |
| - Prefixes for DUTs |  |
| Structure (151) | S_ |
| - Enumeration (152) | E_ |
| Variables with data type UNION (153) | U_ |
| .-. Alias (154) | A_ |

## Example The naming convention (1) refers to the standard POU TON. As a result, declarations of the special library POU are checked for the prefix "ton_". Click the blank space (2) to insert more naming conventions.

## Filter

| Names | Prefix |
| :--- | :--- |
| $\pm$ Prefixes for variables |  |
| $\pm+$ Prefixes for POUs |  |
| $\pm$ Prefixes for DUTs |  |
| - Prefixes for custom types |  |
| Standard.TON | 1 |
|  | 2 |
|  |  |

## Combine scope prefix with data type prefix

First character after prefix should be an upper case letter$\square$ Recursive prefixes for combinable data types

See also

- (3) "Checking for compliance to defined naming conventions" on page 4211
- 乡 Chapter 6.6.3.4.1.4 "Attribute 'naming'" on page 4229
- ⓨ Chapter 6.6.3.4.1.5 "Attribute 'nameprefix'" on page 4230
- Identifiers
- Data Type Alias
- PROPERTY

Dialog 'Static Analysis Settings' - 'Metrics'
Function: In the dialog, you select the metrics to be displayed for each POU in the "Standard Metrics" view by means of the "Build $\rightarrow$ Static Analysis $\rightarrow$ View Standard Metrics" command.

## Call:

- "Open configuration dialog" button in the menu "Project $\rightarrow$ Project Settings", "Static Analysis" category
- Menu bar: "Build $\rightarrow$ Static analysis $\rightarrow$ Settings"


## Requirement:

- The CODESYS Static Analysis package is installed.
- A project is open.

The "Code size", "Variable size", "Stack size", and "Calls" metrics are reported only for POUs from libraries which are integrated in the project.

Violations of the upper and lower limits of the activated metrics can be reported as build errors by means of static analysis rule SA0150.

| "Metrics" | All selectable metrics are displayed in the column. |
| :--- | :--- |
| "Active" | 凹: The metric is displayed for each POU in the "Standard Metrics" view following <br> the "Build $\rightarrow$ Static Analysis $\rightarrow$ View Standard Metrics" command. <br> $\square:$ The metric is not displayed in the "Standard Metrics" view following the "Build <br> $\rightarrow$ Static Analysis $\rightarrow$ View Standard Metrics" command. |
| "Lower limit" | Lower value from which the "Metric" is displayed |

See also

- ${ }^{\text {E }}$ "Displaying of metrics" on page 4212
- 学 Chapter 6.6.3.4.2.52 "SA0150: Violations of lower or upper limits or the metrics" on page 4299


## Dialog 'Static Analysis Settings' - 'Forbidden Symbols'

Function: In the dialog, you define the keywords and symbols that must not be used in the project code.

Call:

- "Open configuration dialog" button in the menu "Project $\rightarrow$ Project Settings", "Static Analysis" category
- Menu bar: "Build $\rightarrow$ Static analysis $\rightarrow$ Settings"


## Requirement:

- The CODESYS Static Analysis package is installed.
- A project is open.

| Input line | Double-clicking the line opens the line editor for specifying a keyword or symbol. <br> $\ldots$ <br> $\ldots$ |
| :--- | :--- |

See also

- (3) "Checking for forbidden symbols" on page 4211


### 6.6.3.4 Reference, Programming

6.6.3.4.1 Pragmas and Attributes................................................................. 4228
6.6.3.4.2 Rules.............................................................................................. 4233

### 6.6.3.4.1 Pragmas and Attributes

## General

CODESYS Static Analysis provides pragmas and attributes for activating or deactivating individual rules or naming conventions for static code analysis.
Requirement: The rules or conventions are activated or defined in the project settings.
Attributes are inserted in the declaration part of a POU to deactivate specific rules for an entire programming object.
Pragmas are used in the implementation part of a POU to deactivate specific rules for individual lines of code. One exception is Rule 164, which can also be switched off in the declaration part.


Rules that are deactivated in the project settings cannot be activated by means of pragmas or attributes.

Rule SA0004 cannot be deactivated by means of a pragma or an attribute.

See also

- $\sum^{4}$ Chapter 6.6.3.4 "Reference, Programming" on page 4227


## Pragma 'analysis'

This pragma is used to deactivate the code rules for individual code lines of a POU. You deactivate code rules by specifying the rule numbers with a prepended minus sign ("-"). A prepended plus sign ("+") activates the rule. You can specify any number of rules in the pragma.
Insert location: Deactivation: In the implementation part, with \{analysis - . . \} before the first code line where the code analysis is deactivated. Activation: With \{analysis + . . \} after the last line of the deactivation. For Rule 164, the pragma can also be inserted in the declaration part before a comment.

## Syntax:

```
Deactivation of rules:
{analysis -<rule number> ( , -<additional rule number> )* }
* : optional none, one or more additional rule numbers
Activation of rules:
{analysis +<rule number> ( , +<additional rule number> )* }
* : none, one or more additional rule numbers
```

Example $\quad$ Rule 24 is deactivated for two lines and then reactivated. As a result, rule 24 is not checked in these lines so that nTest:=DINT\#99 is allowed for example.

```
        {analysis -24}
nTest := 99;
iVar := INT#2;
{analysis +24}
```

Deactivating multiple rules:

```
{analysis -10, -24, -18}
```


## See also

- ${ }^{4}$ Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209


## Attribute 'analysis'

The attribute deactivates specific rules for an entire programming object. You deactivate the code rules by specifying the rule numbers with a prepended minus sign ("-"). You can specify any number of rules in the attribute.

Insert location: In the declaration part of a POU, in the first line.

## Syntax:

```
{attribute 'analysis' := '-<rule number> ( , -<additional rule
number> ) * '}
* : none, one or more additional rule numbers
```

Example Rules 33 and 31 are deactivated for the entire structure:

```
{attribute 'analysis' := '-33, -31'}
TYPE My_Structure :
STRUCT
            iLocal : INT;
            uiLocal : UINT;
            udiLocal : UDINT;
END_STRUCT
END_TYPE
```

Rule 100 is deactivated for the array:

```
{attribute 'analysis' := '-100'}
PROGRAM PLC_PRG
VAR
    aBigData: ARRAY[1..10000] OF DWORD;
    aBigDATA_2: ARRAY[1..10000] OF DWORD;
END_VAR
;
```

See also

- ${ }^{\mu}$ Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209


## Attribute 'naming'

The attribute marks the code lines that are excluded from the analysis of naming convention.
An off is assigned to the pragma attribute before the first code line where the code analysis is deactivated. An on is assigned after the last line. When an omit is assigned, only the next code line is ignored.
Insert location: Deactivation: In the declaration part of POUs and DUTs, above the affected lines. Activation: Below the affected lines.

## Syntax:

```
{attribute 'naming' := '<switch state>'}
<switch state> : on | off | omit
on : naming is switched on
off : naming is switched off
omit : only next code line is switched off
```

```
variables: cccVar, aVariable, and bVariable.
```

```
VAR
```

VAR
{attribute 'naming' := 'off'}
{attribute 'naming' := 'off'}
iVarA : INT;
iVarA : INT;
iVarB : INT;
iVarB : INT;
{attribute 'naming' := 'on'}
{attribute 'naming' := 'on'}
iVarC : INT;
iVarC : INT;
END VAR
END VAR
VAR
VAR
{attribute 'naming' := 'omit'}
{attribute 'naming' := 'omit'}
iVarC : INT;
iVarC : INT;
..
..
END_VAR
END_VAR
{attribute 'naming' := 'omit'}
{attribute 'naming' := 'omit'}
PROGRAM PLC_PRG
PROGRAM PLC_PRG
VAR
VAR
END_VAR
END_VAR
{attribute 'naming' := 'off'}
{attribute 'naming' := 'off'}
PROGRAM DoSomethingA
PROGRAM DoSomethingA
VAR
VAR
{attribute 'naming' := 'on'}
{attribute 'naming' := 'on'}
iVarA : INT;
iVarA : INT;
iVarB : INT;
iVarB : INT;
VAR_END

```
VAR_END
```

Example Defined naming conventions: 1) INT variable names must be prepended with "int" as the
identifier prefix, for example "intVar1". 2) Program names must begin with "prog".
For the code presented below, the static analysis issues messages only for the following

See also

- ${ }^{\text {• }}$ Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209


## Attribute 'nameprefix'

The attribute defines a prefix for variables of a structured data type. The prefix must be prepended to the identifier of variables that are declared by this type.

Insert location: In the line before the declaration of a structured data type
Syntax:

```
{attribute 'nameprefix' := '<prefix>'}
```

```
Example In the following example, Static Analysis issues a message for pB because the variable name
does not begin with "point".
```

```
{attribute 'nameprefix' := 'point'}
```

{attribute 'nameprefix' := 'point'}
TYPE DATAPOINT :
TYPE DATAPOINT :
STRUCT
STRUCT
iX: INT;
iX: INT;
iY: INT;
iY: INT;
END STRUCT
END STRUCT
END_TYPE
END_TYPE
PROGRAM PLC PRG
PROGRAM PLC PRG
VAR
VAR
pointA : DATAPOINT;
pointA : DATAPOINT;
pB : DATAPOINT;
pB : DATAPOINT;
END_VAR
END_VAR
pointA.iX := 1;
pointA.iX := 1;
pointA.iY := 10;
pointA.iY := 10;
pB.iX := 2;
pB.iX := 2;
pB.iY := 20;
pB.iY := 20;
Error message after static analysis: "Invalid variable name ' $p B$ '. Expect prefix 'point'"

```

See also
- \(\Rightarrow\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{Attribute 'analysis:report-multiple-instance-calls'}

The attribute marks a function block for checking for rule 105: Only function blocks with this attribute are checked whether the function block instances are called more than one time. If rule 105 is deactivated in the project settings, then the attribute does not have any effect.

Insert location: Top line in the declaration part of a function block.

\section*{Syntax:}
```

{attribute 'analysis:report-multiple-instance-calls'}

```

\section*{Example}
```

// {attribute 'analysis:report-multiple-instance-calls'} Deactivated
FUNCTION_BLOCK FB_DOA
VAR_INPUT
END - VAR
VAR_OUTPUT
END_VAR
VAR
iA : INT;
END_VAR
iA \overline{:= iA + 1;}
{attribute 'analysis:report-multiple-instance-calls'}
FUNCTION_BLOCK FB_DOB
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
iB : INT;
END VAR
iB := iB +1;
PROGRAM PLC_PRG
VAR
fbA : FB_DoA;
fbB : FB_DoB;
ND_VAR
fbA();
fbB(); // SA0105
fbA();
fbB(); // SA0105
--> SA0105: Instance 'fbB' called more than once

```

See also
- \(\quad\) Chapter 6.6.3.4.2.46 "SA0105: Multiple instance calls" on page 4285
- \(\Leftrightarrow\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209
6.6.3.4.2.1 SA0001: Unreachable code. ..... 4234
6.6.3.4.2.2 SA0002: Empty objects. ..... 4234
6.6.3.4.2.3 SA0003: Empty statements. ..... 4235
6.6.3.4.2.4 SA0004: Multiple write access on output. ..... 4235
6.6.3.4.2.5 SA0006: Write access from several tasks ..... 4236
6.6.3.4.2.6 SA0007: Address operator on constants. ..... 4237
6.6.3.4.2.7 SA0008: Check subrange types ..... 4237
6.6.3.4.2.8 SA0009: Unused return values. ..... 4238
6.6.3.4.2.9 SA0010: Arrays with only one component. ..... 4239
6.6.3.4.2.10 SA0011: Useless declarations ..... 4239
6.6.3.4.2.11 SA0012: Variable which could be declared as constants ..... 4240
6.6.3.4.2.12 SA0013: Declarations with the same variable name. ..... 4240
6.6.3.4.2.13 SA0014: Assignment of instances ..... 4241
6.6.3.4.2.14 SA0015: Access to global data via FB_Init. ..... 4242
6.6.3.4.2.15 SA0016: Gaps in structures ..... 4242
6.6.3.4.2.16 SA0017: Non-regular assignments ..... 4243
6.6.3.4.2.17 SA0018: Unusual bit access. ..... 4243
6.6.3.4.2.18 SA0020: Possibly assignment of truncated value to REAL vari- able. ..... 4244
6.6.3.4.2.19 SA0021: Transporting the address of a temporary variable. ..... 4245
6.6.3.4.2.20 SA0022: (Possibly) unassigned return value. ..... 4245
6.6.3.4.2.21 SA0023: Complex return values. ..... 4246
6.6.3.4.2.22 SA0024: Untyped literals / constants ..... 4246
6.6.3.4.2.23 SA0025: Unqualified enumeration constants ..... 4247
6.6.3.4.2.24 SA0026: Possible truncated strings ..... 4247
6.6.3.4.2.25 SA0027: Multiple uses of identifiers ..... 4248
6.6.3.4.2.26 SA0028: Overlapping memory areas ..... 4248
6.6.3.4.2.27 SA0029: Notation in code different to declaration. ..... 4249
6.6.3.4.2.28 Unused Objects. ..... 4249
6.6.3.4.2.29 SA0034: Enumerations with incorrect assignment. ..... 4252
6.6.3.4.2.30 SA0037: Write access to input variable. ..... 4252
6.6.3.4.2.31 SA0038: Read access to output variable. ..... 4253
6.6.3.4.2.32 SA0040: Possible division by zero. ..... 4254
6.6.3.4.2.33 SA0041: Detect possible loop invariant code ..... 4255
6.6.3.4.2.34 SA0042: Usage of different access paths. ..... 4256
6.6.3.4.2.35 SA0043: Use of a global variable in only one POU ..... 4256
6.6.3.4.2.36 SA0044: Declarations with reference to interface. ..... 4257
6.6.3.4.2.37 Conversions ..... 4258
6.6.3.4.2.38 Use of Direct Addresses ..... 4263
6.6.3.4.2.39 Rules for Operators ..... 4265
6.6.3.4.2.40 Rules for Statements ..... 4275
6.6.3.4.2.41 SA0095: Assignments in conditions. ..... 4281
6.6.3.4.2.42 SA0100: Variables greater than <n> bytes ..... 4282
6.6.3.4.2.43 SA0101: Names with invalid length. ..... 4283
6.6.3.4.2.44 SA0102: Access to program/fb variables from the outside. ..... 4283
6.6.3.4.2.45 SA0103: Concurrent access on not atomic data ..... 4284
6.6.3.4.2.46 SA0105: Multiple instance calls. ..... 4285
6.6.3.4.2.47 SA0106: Virtual method calls in FB_INIT ..... 4286
6.6.3.4.2.48 SA0107: Missing formal parameters ..... 4287
6.6.3.4.2.49 Checking Strict IEC Rules ..... 4288
6.6.3.4.2.50 SA0140: Statements commented out. ..... 4296
6.6.3.4.2.51 Possible Use of Uninitialized Variables. ..... 4296
6.6.3.4.2.52 SA0150: Violations of lower or upper limits or the metrics ..... 4299
6.6.3.4.2.53 SA0160: Recursive calls. ..... 4299
6.6.3.4.2.54 SA0161: Unpacked structure in packed structure. ..... 4300
6.6.3.4.2.55 SA0162: Missing comments. ..... 4301
6.6.3.4.2.56 SA0163: Nested comments. ..... 4302
6.6.3.4.2.57 SA0164: Multiline comments ..... 4303
6.6.3.4.2.58 SA0165: Tasks calling other POUs than programs ..... 4303
6.6.3.4.2.59 SA0166: Max. number of input/output/in-out variables. ..... 4304
6.6.3.4.2.60 SA0167: Temporary function block instances. ..... 4304
6.6.3.4.2.61 SA0168: Unnecessary Assignments ..... 4305
6.6.3.4.2.62 SA0169: Ignored outputs. ..... 4306

\section*{SA0001: Unreachable code}

Detects lines of code that are not executed, for example due to a RETURN or CONTINUE statement

Justification: Unreachable code should always be avoided. The test often indicates that test code still exists which should be removed.

Importance: High

\section*{PLCopen rule: CP2}

\section*{Example}
```

PROGRAM PLC_PRG
VAR
xReturn_Before_End: BOOL;
xContinue_In_LOop_FUN: BOOL;
iCounter: INT;
END_VAR
xContinue In Loop FUN := FALSE;
FOR iCounter := INT\#O TO INT\#5 BY INT\#1 DO
CONTINUE;
xContinue_In_Loop_FUN := FALSE;
END_FOR
--> SA0001: Unreachable code detected in 'PLC_PRG'

```

\section*{See also}
- «Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0002: Empty objects}

Detects POUs, GVLs, data type declarations, or interfaces that do not contain any code
Justification: Empty objects should be avoided. They are often a sign that an object has not been implemented completely. Exception: In some cases, no code is specified in the body of a function block when it should be used by interfaces only. In other cases, a method is created only because it is required by an interface without a sensible implementation being possible for the method. No matter the case, this kind of situation should be commented.

Importance: Medium
See also
- \({ }^{4}\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0003: Empty statements}

Detects lines of code that have a semicolon (; ) but not a statement Justification: An empty statement can be a sign for missing code.
Note: There are good reasons for using empty statements. For example, in a CASE statement it can make sense to explicitly program out all cases, even those where there is nothing to do. When this kind of empty CASE statement contains a comment, Static Analysis does not generate an error message.

Importance: Low

\section*{Examples}
```

CASE value OF
1:DoSomething();
2:;
3:DoSomethingElse();
END_CASE
--> SA0003: Empty statements
CASE value OF
1:DoSomething();
2:; //nothing to do
3:DoSomethingElse();
END_CASE
--> No SA error

```

\section*{See also}
- \({ }^{4}\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0004: Multiple write access on output}

Detects outputs that are written to more than one location
Justification: The maintainability is degraded when an output is written in different locations in the code. Then it is uncertain which write access is the one that actually has an effect in the process. Good practice is to calculate the output variables in auxiliary variables and assign the calculated value at one location at the end of the cycle.
Importance: High
PLCopen rule: CP12


An error is not issued when an output variable (VAR_IN_OUT) is written in different branches of IF and CASE statements.

A pragma cannot deactivate this rule.
```

Example

```
```

VAR_GLOBAL

```
VAR_GLOBAL
    g_xVar AT %QXO.0 : BOOL ;
    g_xVar AT %QXO.0 : BOOL ;
    g_iTest AT %QWO : INT ;
    g_iTest AT %QWO : INT ;
END_VAR
END_VAR
PROGRAM PLC_PRG
PROGRAM PLC_PRG
IF g_iCondition < INT#O THEN
IF g_iCondition < INT#O THEN
    g_xVar := TRUE;
    g_xVar := TRUE;
    g_iTest := INT#12;
    g_iTest := INT#12;
END_IF
END_IF
CASE g_iCondition OF
CASE g_iCondition OF
    INT#1:
    INT#1:
        g xVar := FALSE;
        g xVar := FALSE;
    INT#2:
    INT#2:
            g_iTest := INT#11;
            g_iTest := INT#11;
    ELSE
    ELSE
        g_xVar := TRUE;
        g_xVar := TRUE;
        g_iTest := INT#9;
        g_iTest := INT#9;
END_CASE
END_CASE
--> SAO004: Multiple write access on output '%QXO.O'
--> SAO004: Multiple write access on output '%QXO.O'
--> SAOOO4: Multiple write access on output '%QWO'
```

--> SAOOO4: Multiple write access on output '%QWO'

```

\section*{See also}
- \({ }^{4}\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0006: Write access from several tasks}

Detects variables that are written by more than one task
Justification: A variable that is written in multiple tasks may change its value unexpectedly. This can lead to confusing situations. String variables (and on some 32 -bit systems also 64-bit integer variables) can even reach an inconsistent state if the variable is written to two tasks simultaneously.
Exception: In specific cases, it may be necessary for several tasks to write a variable. For example, use semaphores to make sure that access does not lead to an inconsistent state.
Importance: High
PLCopen rule: CP10

\section*{Example}
```

VAR_GLOBAL
g_iTemp1: INT;
END_VA}
PROGRAM PLC_PRG // Controlled by MainTask
g_iTemp1 := g_iTemp1 + INT\#2;
PROGRAM PLC_PRG_1 //Controlled by SubTask
g_iTemp1 := - g_iTemp1 - INT\#3;
--> SA0006: Concurrent write access to 'g_iTempl' in Tasks
MainTask, SubTask

```

\section*{See also}
- y Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0007: Address operator on constants}

Detects lines of code where the operator ADR is applied for a constant
Justification: Using a pointer to a constant variables overrides the CONSTANT property of the variable. The variable can be changed by means of the pointer without any notification from the compiler.

Exception: In rare cases, it might be useful to pass a pointer to a constant to a function. However, you have to make sure that this function does not change the transferred value. Whenever possible, use VAR_IN_OUT CONSTANT.

Importance: High

When the "replace constants" option is selected in the "Compiler options" of the project settings, the address operator is not permitted for scalar constants (integer, BOOL, REAL) and a complie error is issued. (Constant strings, structures, and arrays always have an address.)

\section*{Example}
```

PROGRAM PLC_PRG
VAR CONSTANT
c iValue : INT := INT\#15;
END_VA}
VAR
poiValue : POINTER TO INT;
END_VAR
poiV
--> SA0007: Address to constant variable 'c_iValue'

```

\section*{See also}
- \(y^{*}\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0008: Check subrange types}

Detects out-of-range violations of subrange types. Assigned literals are already checked by the compiler. When constants are assigned, then the values must be within the defined range. When variables are assigned, then the data types must be identical.

Justification: If subrange types are used, then make sure that this subrange is not exited. The compiler checks for these kinds of subrange violations only for assignments of constants.

Importance: Low

The check is not performed for CFC objects because the code structure does not allow for it.

\section*{Example}
```

VAR_GLOBAL
iVarGlob:INT;
END_VAR
PROGRAM PLC_PRG
VAR
iSubr1: INT (INT\#1..INT\#10);
iSubr2: INT (INT\#1..INT\#1000);
iCount: INT;
by_SubType : BYTE (BYTE\#O..BYTE\#11);
iVar : INT (-4095..4095);
END_VAR
iSubr1 := nCount; // SA0008
iSubr1 := subr2; // SA0008
iSubr1 := gvl.iVarGlob; // SA0008
//byBYTE_SubType := BYTE\#123; //already detected by compiler,
error "Cannot convert type..."
--> SA0008: Subrange variable 'iSubrl' maybe out of allowed range

```

\section*{See also}
- «y Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0009: Unused return values}

Detects function, method and property calls in which the return value is not used Justification: When a function or method returns a return value, it should also be evaluated. The return value often indicates whether or not the function was executed successfully. If not, then you will not be able to identify later whether the return value was forgotten or if it is actually not needed.

Exception: If a return value is irrelevant to the call, then you can document this and omit the assignment. Error returns should never be ignored.
Importance: Medium
PLCopen rule: CP7 / CP17
Example
```

FUNCTION Return_BOOL : BOOL
VAR_INPUT
END_VAR
VAR
xTest : BOOL;
END_VAR
xTes̄t := FALSE;
Return_BOOL := xTest;
PROGRAM PLC_PRG
Return_BOOL (); // SA0009
--> SA0009: Ignoring return value of 'Return_Bool'

```

\section*{See also}
- \({ }^{\mu}\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0010: Arrays with only one component}

Detects arrays with only one element
Justification: An array with one element can be replaced by a base-type variable. Access to this variable is considerably faster than access by index to the variable.
Exception: The length of an array is often determined by a constant and is a parameter for a program. Then the program can work with arrays of different lengths and does not have to be changed if the length is only 1 . This kind of situation should be documented accordingly.
Importance: Low
Example
```

PROGRAM PLC_PRG
VAR
aoiEmpty : ARRAY [22..22] OF INT;
aorEmpty : ARRAY [1..1] OF REAL;
END_VAR
aoiEmpty;
aorEmpty;
--> SA0010: Vacuous array element in variable 'aoiEmpty'
--> SA0010: Vacuous array element in variable 'aorEmpty'

```

\section*{See also}
- «Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0011: Useless declarations}

Detects structures or enumerations with only one component
Justification: This kind of declaration can be confusing for the reader. A structure with only one element can be replaced by an alias type. An enumeration with only one element can be replaced by an constant.
PLCopen rule: CP22 / CP24
Importance: Low

\section*{Example}
```

TYPE SingleStruct :
STRUCT
iPart : INT;
END_STRUCT
END_TYPE
TYPE myUnion :
UNION
lrValue : LREAL;
END_UNION
END_TYPE
TYPE SingleEnum :
(
OnlyOne := 1
);
END_TYPE
--> Useless declaration 'SingleStruct'
--> Useless declaration 'myUnion'
--> Useless declaration 'SingleEnum'

```

See also
- \({ }^{*}\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

SA0012: Variable which could be declared as constants
Detects variables that are not accessed with write permission and therefore could be declared as constants

Justification: If a variable is written only at the declaration point and is otherwise used only for reading, then the static analysis assumes that the variable should also not to be changed. Firstly, a declaration as a constant results in checking that the variable is not changed when the program is changed. Secondly, the declaration as a constant may result in faster code.
```

NOTICE!
If multiple applications exist in one project, then only the objects below the currently active application are affected. If there is only one application, then the objects in the common POU pool are also affected.

```

Importance: Low
Example
```

PROGRAM PLC_PRG
VAR
iVar : INT := INT\#17;
iTest : INT;
END_VAR
iTest := iTest + iVar; // SA0012: iVar could be declared as
constant
--> SA0012: Variable 'iVar' could be declared as constant

```

\section*{See also}
- \(\Rightarrow\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0013: Declarations with the same variable name}

Detects variables with names that are already used by other variables (for example, global and local variables with the same name). Also detects variables with names of functions, actions, methods, or properties which are used in the same access scope. Variables are also detected that are declared in a GVL in the "Devices" view or in the POUs pool. For this, however, the GVL of the "POUs" view have to be used in the application program.

Justification: The same names can be confusing when reading the code, and they can cause errors if the wrong object is accessed unintentionally. We recommend that you use naming conventions to avoid these situations.

PLCopen rule: N5 / N9
Importance: Medium
```

Example

```
```

VAR_GLOBAL

```
VAR_GLOBAL
    xVar1 : BOOL;
    xVar1 : BOOL;
    iVar3 : INT;
    iVar3 : INT;
END_VAR
END_VAR
PROGRAM PLC_PRG
PROGRAM PLC_PRG
VAR
VAR
    xVar1 : BOOL; // SA0013
    xVar1 : BOOL; // SA0013
    iVar3 : INT; // SA0013
    iVar3 : INT; // SA0013
END_VAR
END_VAR
xVar1 := NOT GVL.xVar1;
xVar1 := NOT GVL.xVar1;
iVar3 := iVar3 + INT#2;
iVar3 := iVar3 + INT#2;
iVar3 := GVL.iVar3;
iVar3 := GVL.iVar3;
--> SA0013: Declaration of 'iVar1' hides symbol 'GVL.iVar1
--> SA0013: Declaration of 'iVar1' hides symbol 'GVL.iVar1
--> SA0013: Declaration of 'xVar3' hides symbol 'GVL.xVar3
```

--> SA0013: Declaration of 'xVar3' hides symbol 'GVL.xVar3

```

Example The function block POU has the action ACT and the method METH.
```

FUNCTION_BLOCK POU
VAR
ACT : UINT; // SA0013
METH : BYTE; // SA0013
END_VAR
--> SA0013: Declaration of 'ACT' hides symbol 'POT.ACT'
--> SAOO13: Declaration of 'METH' hides symbol 'POT.METH'

```

\section*{See also}
- \({ }^{\text {n }}\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0014: Assignment of instances}

Detects assignments to function block instances. In the case of instances with pointer or reference variables, these assignments are potentially risky.
Justification: This is a performance warning. When an instance is assigned to another instance, all elements and subelements are copied from the one instance to the other instance. Pointers to data are also copied, but not their referenced data, so that the target instance and the source instance contain the same data after the assignment. Depending on the size of the instances, this kind of assignment could last a long time. For example, if an instance should be passed to a function for processing, then it is much more efficient to pass a pointer to the instance. If you want to selectively copy values from one instance to another, then a copy method is useful: inst_First.Copy_From(inst_Second).
Importance: Medium

\section*{Example}
```

PROGRAM PLC_PRG
VAR
inst_First : My_FB;
inst_Second : My_FB;
END VAR
inst_First();
inst_Second := inst_First; // SA0014
--> SA0014: Assignment of instances

```

\section*{See also}
- « Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0015: Access to global data via FB_Init}

Detects the access of a function block to global variables by means of the method FB_Init. The value of this variable depends on the order of initializations.
Justification: Depending on the declaration location of the POU instance, an uninitialized variable could be accessed if the rule is violated.
Importance: High

\section*{Example}
```

VAR_GLOBAL
g_xTest1 : BOOL;
g_iTest3 : INT;
END_VAR
METHOD PUBLIC fb_init : BOOL
VAR INPUT
bInitRetains : BOOL; // If TRUE, the retain variables are
initialized (warm start / cold start)
bInCopyCode : BOOL; // If TRUE, the instance afterwards gets
moved into the copy code (online change)
END_VAR
g_x\overline{Test1 := NOT g_xTest1; // SA0015}
g_iTest3 := g_iTest3 + INT\#1; // SA0015
--> SA0015: FB_Init method of function block 'POU' accesses global
data

```

\section*{See also}
- «. Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0016: Gaps in structures}

Detects gaps in structures or function blocks that are caused by the alignment requirements of the currently set target system. If possible, you should remove the gaps by resorting the structure elements or filling them with a dummy element. If this is not possible, then you can deactivate the rule for the affected structures by means of the analysis pragma.
Justification: Due to different alignment requirements on different platforms, there may be a different layout in the memory for these kinds of structures. Then the code can perform differently, depending on the platform.
Importance: Low
```

Example

```
```

PROGRAM PLC_PRG

```
PROGRAM PLC_PRG
VAR
VAR
    myStruct : Unpadded_Structure;
    myStruct : Unpadded_Structure;
END VAR
END VAR
myS\overline{truct.iTest := 0;}
myS\overline{truct.iTest := 0;}
TYPE Unpadded_Structure :
TYPE Unpadded_Structure :
STRUCT
STRUCT
    xTest : BOOL;
    xTest : BOOL;
    iTest : INT; // SA0016
    iTest : INT; // SA0016
    byTest : BYTE;
    byTest : BYTE;
    wTest : WORD;
    wTest : WORD;
END_STRUCT
END_STRUCT
END_TYPE
END_TYPE
--> SA0016: Structure 'Unpadded_Structure' must be padded (pack-
--> SA0016: Structure 'Unpadded_Structure' must be padded (pack-
mode=8)
```

mode=8)

```

\section*{See also}
- \& Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0017: Non-regular assignments}

Detects assignments to pointers that are neither addresses (ADR operator, pointer variables) nor constants 0

Justification: If a pointer contains a value that is not a valid address, then an access violation exception occurs when dereferencing the pointer.
Importance: High
Example
```

PROGRAM PLC_PRG
VAR
pInt : POINTER TO INT;
dwAddress : DWORD;
END VAR
dwA\overline{d}dress := dwAddress + DWORD\#1;
pInt := dwAddress; // SA0017
--> SA0017: Non-regular assignment

```

\section*{See also}
- «Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0018: Unusual bit access}

Detects bit access to signed variables. However, the IEC 61131-3 standard permits only bit access and bit shift operations on bitfields. See also the strict rules SA0147 and SA0148.
Justification: Signed data types should not be used as bitfields and the other way around. The IEC 61131-3 standard does not provide for this kind of access, and therefore you should comply with this rule when you write portable code.
Importance: Medium

> Exception for flag enumerations: When an enumeration is declared as a flag by means of the \{attribute 'flags'\} pragma attribute, the SA0018 error is not issued for bit access with the OR, AND or NOT operators.

Example
```

PROGRAM PLC_PRG
VAR
iTemp1 : INT;
diTemp3 : DINT;
uliTemp4 : ULINT;
siTemp5 : SINT;
usiTemp6 : USINT;
byTemp2 : BYTE;
END_VAR
iTemp1.3 := TRUE; // SA0018
diTemp3.4 := TRUE; // SA0018
uliTemp4.18 := FALSE; // no error because this is an unsigned data
type
siTemp5.2 := FALSE; // SA0018
usiTemp6.3 := TRUE; // no error because this is an unsigned data
type
byTemp2.5 := FALSE; // no error because the byte is a bitfield
--> SAOO18: Unusual bit access

```

See also
- \({ }^{4}\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209
- \(\Leftrightarrow\) Chapter 6.6.3.4.2.49.12 "SA0147: Unusual shift operation - strict" on page 4293
- \(\quad\) Chapter 6.6.3.4.2.49.13 "SA0148: Unusual bit access - strict" on page 4294

\section*{SA0020: Possibly assignment of truncated value to REAL variable}

Detects operations on integer variables for which a truncated value could be assigned to a REAL data type variable

Justification: Static analysis issues an error when the result of an integer calculation is assigned to a REAL or LREAL variable. The programmer should be alerted to a possible incorrect interpretation of this kind of assignment: lrealvar := dintvar1 * dintvar2. Because the range of values of LREAL is greater than that of DINT, one could assume that the result of the calculation could always be represented in LREAL. But that is not the case. The processor calculates the result of the multiplication as an integer and then casts the result to LREAL. An overflow in the integer calculation would be lost. To work around the problem, the calculation has to be done as a REAL operation: lreal_var := TO_LREAL (dintvar1) * TO_LREAL (dintvar2).
Importance: High
Example
```

PROGRAM PLC_PRG
VAR
rx : LREAL;
di : DINT;
END VAR
rx \overline{:= di * di // SA0020}
rx := TO_LREAL(di) * TO_LREAL(di) // No message
--> SA0020: Possibly assignment of truncated value to REAL variable

```

See also
- «Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0021: Transporting the address of a temporary variable}

Detects address assignments of temporary variables (on the stack) to non-temporary variables
Justification: Local variables of a function or method are created on the stack and they exist only while the function or method is being processed. If a pointer points to this kind of variable after processing the method or function, then you can use this pointer to access undefined memory, or to access an incorrect variable in another function. This situation should be avoided at all costs.

Importance: High

\section*{Example}
```

FUNCTION TempVarInFUNC : DWORD
VAR
uiTemp : UINT;
END_VAR
Temp\overline{p}VarInFUNC := ADR(uiTemp); // SA0021
PROGRAM PLC PRG
VAR
dwTest : DWORD;
END VAR
dwTest := TempVarInFUNC();
--> SA0021: Transporting address of temporary variable to outer
scope symbol

```

\section*{See also}
- \& Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0022: (Possibly) unassigned return value}

Detects all functions and methods that include an execution thread without an assignment to the return value
Justification: An unassigned return value in a function or method is an indication of missing code. Even if the return value always has a default value, it is always useful to assign it again explicitly to avoid confusion.

Importance: Medium

\section*{Example}
```

FUNCTION FUN : DINT
VAR_INPUT
bTest : BOOL;
END_VAR
IF bTest THEN
RETURN;
END IF
FUN-
--> SAOO22: (Possibly) unassigned return value

```

See also
- \({ }^{*}\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0023: Complex return values}

Determines complex return values that cannot be returned with a simple registry copy of the processor. This includes structures, arrays, and return values of type STRING (regardless of the size of the used memory).
Justification: This is a performance warning. If large values are returned as the result of a function, method, or property, then the processor copies them multiple times when executing the code. This can lead to runtime problems and should be avoided whenever possible. Performance can be improved by passing a structured value as VAR_IN_OUT to a function or method and filling it in the function or method.
Importance: Medium

\section*{Example}
```

TYPE LargeStructure :
STRUCT
a : LINT;
b : BOOL;
END_STRUCT
END_TYPE
FUNCTION Large_Return_Value_FUNC : LargeStructure // SA0023
--> SA0023: Complex return values

```

\section*{See also}
- 4 Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0024: Untyped literals / constants}

Detects untyped literals and constants
Justification: CODESYS assigns types for literals depending on their use. In some cases, this can cause unexpected problems, which should be resolved better with a typed literal. For example: dw := ROL(DWORD\#1, i)
Importance: Low

\section*{Example}
```

PROGRAM PLC_PRG
VAR
iTemp1 : INT = 10; // SA0024
diTemp2 : DINT;
liTemp3 : LINT;
rTemp4 : REAL;
lrTemp5 : LREAL;
END_VAR
iTemp1 := iTemp1 + INT\#34;
diTemp2 := diTemp2 + 23; // SA0024
liTemp3 := liTemp3 + 124; // SA0024
rTemp4 := rTemp4 + 1.1; // SA0024
lrTemp5 := lrTemp5 + 3.4; // SA0024
--> SA0024: Untyped literal found

```

See also
- \(\boldsymbol{y y}^{\text {• }}\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0025: Unqualified enumeration constants}

Detects enumeration constants for which a qualified name does not prepend the enumeration Justification: Qualified access makes the code more readable and easier to maintain. Without forcing qualified variable names, an additional enumeration could be inserted when the program is extended. This enumeration contains a constant with the same name as an existing enumeration (see the example below: "red"). This would result in ambiguous access to this piece of code. We recommend to always use only enumerations with the \{attribute 'qualified-only'\}.
Importance: Medium

\section*{Example}
```

TYPE COLOR
(red,green,blue);
END_TYPE
PROGRAM PLC_PRG
enumVar : COLOR;
enumVar := COLOR.red; // SA0025
enumVar := red; // SA0025
--> SA0025: Enumeration constant 'red' not qualified

```

\section*{See also}
- \({ }^{4}\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0026: Possible truncated strings}

Detects string assignments and string initializations that do not use sufficient string length Justification: When strings of different lengths are assigned, a string could be truncated. This can have unexpected results.

Importance: Medium

\section*{Example}
```

PROGRAM PLC_PRG
VAR
strVar1 : STRING[10];
strVar2 : STRING[6];
strVar3 : STRING[6] := 'abcdefghi'; // SA0026
END_VAR
strVar2 := strVar1; // SA0026
--> SA0026: Truncation of string 'abcdefghi'
--> SA0026: Possible truncation of string 'strVarl'

```

\section*{See also}
- \({ }^{4}\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0027: Multiple uses of identifiers}

Detects multiple uses of a name/identifier for a variable or an object (POU) within the scope of a project
Justification: Same names can be confusing when reading the code. They can cause errors if the wrong object is accessed accidentally. Define and follow naming conventions to avoid any situation like this.

The following cases are detected:
- The name of an enumeration is identical to the name of another enumeration in the application or in an integrated library.
- The name of a variable is identical to the name of another object in the application or in an integrated library.
- The name of a variable is identical to the name of an enumeration constant in an enumeration in the application or in an integrated library.
- The name of an object is identical to the name of another object in the application or in an integrated library.
Importance: Medium
```

Example The Standard library is integrated in the project and provides the TON function.

```
```

PROGRAM PLC_PRG

```
PROGRAM PLC_PRG
VAR
VAR
ton : INT;
ton : INT;
END_VAR
END_VAR
--> Variable name 'ton' in 'PLC_PRG' already used for an object in
--> Variable name 'ton' in 'PLC_PRG' already used for an object in
library 'standard, ...'
```

library 'standard, ...'

```

\section*{See also}
- \& Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0028: Overlapping memory areas}

Detects the lines of code where two or more variables reserve the same memory Justification: When two variables reserve the same memory, the code may behave with unexpected results. This situation should be avoided at all costs. If you cannot avoid using a value in different interpretations (for example, one time as DINT and another time as REAL), then you should define a UNION. You can also use a pointer to access a value with a different type without the value being converted.
Importance: High

\section*{Example}
```

PROGRAM PLC_PRG
VAR
iVvar1 AT %QB21: INT;
dwVar2 AT %QD5: DWORD;
END_VAR
--> The following variables access the same memory:
--> SA0028: iVar1 AT %QB21
--> SA0028: dwVar2 AT %QD5

```

\section*{See also}
- «Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0029: Notation in code different to declaration}

Detects the code locations where the notation of an identifier is different from the notation in its declaration

Justification: The IEC 61131-3 standard defines identifiers as not case-sensitive. This means that a variable declared as "varx" can also be used as "VaRx" in the code. However, this is confusing and misleading and should be avoided.
Importance: Medium
```

Example A POU PLC_PRG and a POU fnc (function) exist in the device tree.
PROGRAM PLC_PRG
VAR
iVar: INT;
123test_var_: INT;
END_V/VAR
ivar := iVar + 1; // SA0029
_123TEST_var_ := _123test_var_; // SA0029
Fnc(); - - // SA0029
--> SAO029: Notation in code (ivar) must equal declaration (iVar)
--> SA0029: Notation in code (_123TEST_var_) must equal declaration
(_123test_var_)
--> SA0029: Notation in code (Fnc) must equal declaration (fnc)

```

See also
- „ Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{Unused Objects}
6.6.3.4.2.28.1 SA0031: Unused signatures ..... 4249
6.6.3.4.2.28.2 SA0032: Unused enumeration constants ..... 4250
6.6.3.4.2.28.3 SA0033: Unused variables ..... 4250
6.6.3.4.2.28.4 SA0035: Unused input variables. ..... 4251
6.6.3.4.2.28.5 SA0036: Unused output variables ..... 4251

\section*{SA0031: Unused signatures}

Detects programs, function blocks, functions, data types, interfaces, methods, properties, and actions that are not called within the compiled program code
Justification: Unused objects unnecessarily increase the size of the project and can be confusing when reading the code.
Importance: Low
PLCopen rule: CP2

If multiple applications exist in a project, then only the objects below the currently active applications are affected. If there is only one application, then the objects in the POU pool are also affected.

\section*{See also}
- \({ }^{4}\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0032: Unused enumeration constants}

Detects enumeration constants that are not used in the compiled program code
Justification: Unused enumeration constants unnecessarily increase the size of the enumeration definition and can be confusing when reading the program.

PLCopen rule: CP24
Importance: Low

If multiple applications exist in a project, then only the objects below the currently active applications are affected. If there is only one application, then the objects in the common POU pool are also affected.

Example
```

TYPE My_Enum :
(
one := 1, two := 2
);
END_TYPEE
--> SA0032: Unused enumeration constant 'one'
--> SA0032: Unused enumeration constant 'two'

```

\section*{See also}
- \& Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0033: Unused variables}

Detects variables that are declared but not used within the compiled program code
Justification: Unused variables make a program less readable and maintainable. Unused variables unnecessarily fill memory and unnecessarily waste runtime during initialization.

Importance: Medium
PLCopen rule: CP22 / CP24


For GVL variables:If multiple applications exist in a project, then only the objects below the currently active applications are affected. If there is only one application, then the objects in the common POU pool are also affected.

\section*{Example}
```

PROGRAM PLC_PRG
VAR
iCounter1 : INT;
iCounter2 : INT; // SA0035
END_VAR
ICounter1 := 100;
--> SA0035: Unused Variable 'iCounter2'

```

\section*{See also}
- \(\Rightarrow\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0035: Unused input variables}

Detects input variables that are not used by any function block instance
Justification: Unused variables make a program less readable and maintainable. Unused variables unnecessarily fill memory and unnecessarily waste runtime during initialization.
Importance: Medium
PLCopen rule: CP24

\section*{Example}
```

FUNCTION_BLOCK AFB
VAR_INPUT
iIn1: INT;
iIn2: INT;
END_VAR
VAR_OUTPUT
iOut1: INT;
END_VAR
PROGRAM PLC_PRG
VAR
Fb1: AFB;
END_VAR
Fb1(iIn1 := 99)
--> SA0035: Unused input 'iIn2'

```

See also
- «A Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0036: Unused output variables}

Detects output variables that are not used by any function block instance
Justification: Unused variables make a program less readable and maintainable. Unused variables unnecessarily fill memory and unnecessarily waste runtime during initialization.
Importance: Medium
PLCopen rule: CP24
Example
```

FUNCTION_BLOCK AFB
VAR_INPUT
iIn1: INT;
iIn2: INT;
END_VAR
VAR_OUTPUT
iOut1: INT;
END_VAR
PROGRAM PLC_PRG
VAR
Fb1: AFB;
END_VAR
Fb1(iIn1 := 99)
--> SA0036: Unused output 'iOut1'

```

See also
- \({ }^{*}\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0034: Enumerations with incorrect assignment}

Detects values that are assigned to an enumeration variable. Only defined enumeration constants of an enumeration variable are permitted to be assigned.

Justification: A variable of the enumeration type should have only the intended values, otherwise the code that uses this variable may not work correctly. We recommend to always use enumerations with the \{attribute 'strict'\}. Then the compiler already checks the correct use of the enumeration components.
Importance: High

\section*{Example}
```

TYPE COLOR :
(
Red := 0,
Green,
Yellow
);
END_TYPE
PROGRAM PLC_PRG
VAR
eColor1: COLOR;
END_VAR
eColor1 := COLOR.Red;
eColor1 := 1; // SA0034
--> SA0034: Use enumeration value instead of 'INT\#1'

```

\section*{See also}
- \({ }^{4}\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0037: Write access to input variable}

Detects input variables (VAR_INPUT) that are accessed with write permission within the POU Justification: According to the IEC 61131-3 standard, an input variable must not be changed within a POU. This kind of access is also a cause for errors and makes the code poorly maintainable. This is an indication that a variable is used as both an input variable and an auxiliary variable. This kind of dual use should be avoided.
Importance: Medium

\section*{Example}
```

VAR_GLOBAL
g_xGlob AT %QXO.0 : BOOL;
END_VAR
PROGRAM PLC PRG
VAR_INPUT
xVarIn1:BOOL;
xVarIn2:BOOL;
END_VAR
VAR
iCondition : INT;
END_VAR
iCondition := iCondition + INT\#1;
CASE iCondition OF
INT\#1:
g_xGlob := xVarIn1;
INTT\#2:
g_xGlob := xVarIn2;
ELSE
g_xGlob := FALSE;
xVarIn1 := FALSE; // SA0037
END_CASE
--> SA0037: Write access to input variable 'xVarIn1'

```

\section*{See also}
- \& Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0038: Read access to output variable}

Detects output variables (VAR_OUTPUT) that are accessed with read permission within the POU

Justification: According to the IEC 61131-3 standard, it is prohibited to read an output within a POU. This is an indication that the output is not only used as an output but also as a temporary variable for intermediate results. This kind of dual use should be avoided.
Importance: Low

\section*{Example}
```

VAR_GLOBAL
g_xGlob AT %QXO.0 : BOOL ;
g_iGlob AT %QW1 : INT ;
END_VAR
PROGRAM PLC_PRG
VAR_OUTPUT
xVarOut1:BOOL;
xVarOut2:INT;
xVarOut3:INT;
END VAR
VAR
iCondition : INT;
END VAR
iCondition := iCondition + INT\#1;
CASE iCondition OF
INT\#1:
xVarOut1 := g_xGlob;
xVarOut2 := g_iGlob;
INT\#2:
xVarOut3 := xVarOut2; // SA0038
ELSE
xVarOut1 := FALSE;
g_xGlob := xVarOut1; // SA0038
xVarOut2 := INT\#0;
xVarOut3 := INT\#-1;
END_CASE
--> SA0038: Read access to output variable 'xVarOUT2'
--> SAO038: Read access to output variable 'xVarOUT1'

```

\section*{See also}
- « Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0040: Possible division by zero}

Detects code locations where there is possible division by zero
Justification: Division by zero should never occur, and a variable denominator should always be checked for 0 first.
Importance: High
```

Example

```
```

VAR_GLOBAL

```
VAR_GLOBAL
    g_iVar AT %QW1 : INT ;
    g_iVar AT %QW1 : INT ;
END_VĀR
END_VĀR
PROGRAM PLC_PRG
PROGRAM PLC_PRG
VAR
VAR
    iCounter : INT;
    iCounter : INT;
    iSumme:INT;
    iSumme:INT;
    iMid:INT;
    iMid:INT;
    iVal1:INT := INT#2;
    iVal1:INT := INT#2;
    iVal2:INT;
    iVal2:INT;
    iVal3:INT := INT#3;
    iVal3:INT := INT#3;
    iVal4:INT := INT#4;
    iVal4:INT := INT#4;
    iVal5:INT;
    iVal5:INT;
END_VAR
END_VAR
IF iVal2 <> 0 THEN
IF iVal2 <> 0 THEN
iVal1 := iVal1/iVal2; // no error
iVal1 := iVal1/iVal2; // no error
END_IF;
END_IF;
iMi\overline{d}}:= iSumme / iCounter; // SA004
iMi\overline{d}}:= iSumme / iCounter; // SA004
iCounter := iCounter + INT#1;
iCounter := iCounter + INT#1;
iSumme := g_iVar + iSumme;
iSumme := g_iVar + iSumme;
IF iMid < INTT#100 THEN
IF iMid < INTT#100 THEN
    iVal1 := iVal1 / iVal2; // SA0040
    iVal1 := iVal1 / iVal2; // SA0040
END_IF
END_IF
--> SAO040: Possible division by zero
--> SAO040: Possible division by zero
--> SAOO40: Possible division by zero
```

--> SAOO40: Possible division by zero

```

See also
- \& Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0041: Detect possible loop invariant code}

Detects assignments in loops that calculate the same value for each loop cycle. These lines of code could possibly be inserted outside of the loop.
Justification: This is a performance warning. Code that is executed in a loop, but does the same thing in each loop cycle, can be executed outside of the loop.
Importance: Medium
Example
```

PROGRAM PLC_PRG
VAR
iCounter, iVar1, iVar2: INT;
END_VAR
FOR iCounter := 0 TO 10 DO
iVar1 := 100; // SA0041
iVar2 := iVar2 + iVar1;
END_FOR
--> SANO041: Possible loop invariant code 'iVar1 := 100'

```

See also
- \& Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0042: Usage of different access paths}

Detects the usage of different access paths for the same variable
Justification: Different access to the same element decreases the readability and maintainability of a program. We recommend the consistent usage of \{attribute 'qualified-only'\} for libraries, global variable lists, and enumerations. This forces a fully qualified access.
Importance: Low

\section*{Example}
```

VAR_GLOBAL
iTemp:INT;
instPOU:POU;
END_VAR
FUNCTION_BLOCK POU
VAR
a:INT;
END_VAR
a := INT\#1;
PROGRAM SAO042
VAR
ptiTemp:POINTER TO INT;
sTemp:STRING;
END_VAR
ptiTemp := ADR(iTemp);
ptiTemp^:= INT\#1;
iTemp:= INT\#2; // SA0042 - direct access
on variable
GVL.iTemp := INT\#3; // SA0042 - access on
variable via GVL
sTemp := CONCAT( 'ab', 'cd'); // SA0042 - direct access on
function
sTemp := Standard.CONCAT( 'ab', 'cd'); // SA0042 - access on
function via Standard
instPOU(); // SA0042 - direct access
on POU instance
GVL.instPOU(); // SA0042 - access via GVL
--> SA0042: Different access paths for 'CONCAT'
--> SA0042: Different access paths for 'Standard.CONCAT'
--> SA0042: Different access paths for 'instPOU'
--> SA0042: Different access paths for 'GVL.instPOU'
--> SA0042: Different access paths for 'iTemp'
--> SA0042: Different access paths for 'GVL.iTemp'

```

\section*{See also}
- \& Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0043: Use of a global variable in only one POU}

Detects the use of a global variable in only a single POU
Justification: A global variable that is used in only one location should also only be declared at this location.
Importance: Medium

PLCopen rule: CP26

\section*{Example}
```

VAR_GLOBAL
g_\overline{x}Var AT %QX0.0 : BOOL ;
g_iTest AT %QW1 : INT ;
g_wTest AT %QW2 : WORD;
END_VA}
PROGRAM prog1
VAR
iCondition : INT;
bTemp : BOOL;
END_VAR
iCondition := iCondition + INT\#1;
IF iCondition < INT\#O THEN
bTemp := g_xVar; // SA0043 - g_xVar only read in this POU
ELSIF iCondition = INT\#O THEN
bTemp := g_xVar; // SAO043 - g_xVar only read in this POU
ELSE
bTemp := g_xVar; // SA0043 - g_xVar only read in this POU
g_wTest := WORD\#4; // g_WTest used also in prog2 -> OK
END_IF
PROGRAM prog2
VAR
iCondition : INT;
END VAR
iCoñdition := iCondition + INT\#1;
CASE iCondition OF
INT\#1:
g_iTest := WORD_TO_INT(g_wTest); // SA0043 - g_iTest only
written in this POU
INT\#2:
g_iTest := INT\#2; // SA0043 - g_iTest only
written i\overline{n}\mathrm{ this POU}
ELSE
g_iTest := INT\#3; // SA0043 - g_iTest only
written i\overline{n}\mathrm{ this POU}
END_CASE
--> SA0043: Global variable 'g_xVar' only used in 'prog1'
--> SA0043: Global variable 'g_iTest' only used in 'prog2'

```

\section*{See also}
- «Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0044: Declarations with reference to interface}

Detects declarations with REFERENCE TO interfaces and declarations of VAR_IN_OUT variables with interfaces (implicitly implemented by means of REFERENCE TO)
Justification: An interface type is always implicitly a reference to an instance of a function block that implements this interface. A reference to an interface is therefore a reference to a reference and can result in unwanted behavior.
Importance: High
```

Example ITF is an interface that is defined in the project.

```
```

PROGRAM PLC_PRG

```
PROGRAM PLC_PRG
VAR
VAR
            inst:POU;
            inst:POU;
            itf_inst1 : ITF;
            itf_inst1 : ITF;
            itf}\mp@subsup{}{-}{-
            itf}\mp@subsup{}{-}{-
END_VAR FUNCTION_BLOCK POU
END_VAR FUNCTION_BLOCK POU
VAR_INPUT
VAR_INPUT
            inst itf2 : ITF;
            inst itf2 : ITF;
END_VAR
END_VAR
VAR_OUTPUT
VAR_OUTPUT
            inst_itf3 : ITF;
            inst_itf3 : ITF;
END_VAR
END_VAR
VAR_IN_OUT
VAR_IN_OUT
            ins̄t_itf4 : ITF; // SA0044
            ins̄t_itf4 : ITF; // SA0044
END_VAR
END_VAR
--> SA0044: Reference to interface 'itf ref'
--> SA0044: Reference to interface 'itf ref'
--> SA0044: Reference to interface 'itf4_ref'
```

--> SA0044: Reference to interface 'itf4_ref'

```

See also
- \({ }^{*}\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{Conversions}
6.6.3.4.2.37.1 SA0019: Implicit pointer conversions. ..... 4258
6.6.3.4.2.37.2 SA0130: Implicit expanding conversions ..... 4259
6.6.3.4.2.37.3 SA0131: Implicit narrowing conversions. ..... 4260
6.6.3.4.2.37.4 SA0132: Implicit signed/unsigned conversions ..... 4261
6.6.3.4.2.37.5 SA0133: Explicit narrowing conversions ..... 4261
6.6.3.4.2.37.6 SA0134: Explicit signed/unsigned conversions ..... 4262

\section*{SA0019: Implicit pointer conversions}

Detects implicitly generated pointer conversions
Justification: In CODESYS, pointers are not strictly typed and they can be assigned to each other in any way. This is often used and therefore not reported by the compiler. However, it can also accidentally cause unexpected access. If you assign a POINTER TO BYTE to a POINTER TO DWORD, then you can unintentionally overwrite memory using the latter pointer. Therefore, always check this rule and block the message for cases in which you intentionally want to access a value with a different type.
Implicit data type conversions are reported with a different message.
Importance: High
PLCopen rule: CP25
Exception: BOOL <-> BIT

\section*{Example}
```

PROGRAM PLC_PRG
VAR
pINT : POINTER TO INT;
byteVar : BYTE;
END_VAR
pINT := ADR(byteVar);
--> SA0019: Implicit conversion from pointer to 'POINTER TO BYTE'
to pointer to 'POINTER TO INT'

```

\section*{See also}
- \& Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0130: Implicit expanding conversions}

Detects implicit conversions from smaller data types to larger data types
Justification: The compiler permits any assignments of different types when the value range of the source type is completely contained within the value range of the target type. However, the compiler will build a conversion into the code as late as possible. For an assignment of type lint := dint * dint, the compiler performs the implicit conversion only after multiplication: lint := TO_LINT (dint * dint). An overflow is therefore truncated. To prevent this, you can already convert the elements: lint := TO_LINT (dint) * TO_LINT (dint). Therefore, it may be useful to report locations where the compiler implements implicit conversions in order to check whether these are exactly what is intended. Furthermore, explicit conversions can be used to improve portability to other systems when those systems have more restrictive type checks.
Importance: Low
```

Example

```
```

PROGRAM PLC_PRG

```
PROGRAM PLC_PRG
VAR
VAR
    byTemp : BYTE;
    byTemp : BYTE;
    usiTemp : USINT;
    usiTemp : USINT;
        uiTemp: UINT;
        uiTemp: UINT;
        iTemp : INT;
        iTemp : INT;
        udiTemp: UDINT;
        udiTemp: UDINT;
    diTemp : DINT;
    diTemp : DINT;
uliTemp : ULINT;
uliTemp : ULINT;
        liTemp : LINT;
        liTemp : LINT;
        lwTemp : LWORD;
        lwTemp : LWORD;
        lrTemp : LREAL;
        lrTemp : LREAL;
END_VAR
END_VAR
liTemp := iTemp; // SA0130
liTemp := iTemp; // SA0130
uliTemp := usiTemp; // SA0130
uliTemp := usiTemp; // SA0130
lwTemp := udiTemp; // SA0130
lwTemp := udiTemp; // SA0130
lrTemp := byTemp; // SA0130
lrTemp := byTemp; // SA0130
diTemp := uiTemp; // SA0130
diTemp := uiTemp; // SA0130
byTemp.5 := FALSE; // OK (BIT_BOOL conversion)
byTemp.5 := FALSE; // OK (BIT_BOOL conversion)
--> SA0130: Implicit widening conversion from type 'INT' to type
--> SA0130: Implicit widening conversion from type 'INT' to type
'LINT'
'LINT'
--> SA0130: Implicit widening conversion from type 'USINT' to type
--> SA0130: Implicit widening conversion from type 'USINT' to type
'ULINT'
'ULINT'
--> SA0130: Implicit widening conversion from type 'UDINT' to type
--> SA0130: Implicit widening conversion from type 'UDINT' to type
'LWORD'
'LWORD'
--> SA0130: Implicit widening conversion from type 'BYTE' to type
--> SA0130: Implicit widening conversion from type 'BYTE' to type
'LREAL'
'LREAL'
--> SA0130: Implicit widening conversion from type 'UINT' to type
--> SA0130: Implicit widening conversion from type 'UINT' to type
'DINT'
```

'DINT'

```

\section*{See also}
- \# Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0131: Implicit narrowing conversions}

Detects implicit conversions from larger data types to smaller data types
Justification: This message is obsolete now because it is already reported as a warning by the compiler.
Importance: Low

\section*{Example}
```

PROGRAM PLC_PRG
VAR
rTemp : REAL;
lrTemp : LREAL;
END_VAR
rTemp := lrTemp; // SA0131
--> SA0131: Implicit narrowing conversion from type 'LREAL' to type
'REAL'

```

\section*{See also}
- \& Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

SA0132: Implicit signed/unsigned conversions
Detects implicit conversions from signed data types to unsigned data types or the other way around.

This message is obsolete now because it is already reported as a warning by the compiler.

Importance: Low

\section*{Example}
```

PROGRAM PLC_PRG
VAR
byTest :BYTE;
udiTest: UDINT;
ulktest: ULINT;
wTest : WORD;
lwTest : LWORD;
siTest : SINT;
iTest : INT;
diTest : DINT;
liTest :LINT;
END VAR
liTēst := ulktest; // SA0132
udiTest:= diTest; // SA0132
siTest := byTest; // SA0132
wTest := iTest; // SA0132
lwTest := siTest; // SA0132
--> SA0132: Implicit signed/unsigned conversion from type 'ULINT'
to type 'LINT'
--> SAO132: Implicit signed/unsigned conversion from type 'DINT' to
type 'UDINT'
--> SA0132: Implicit signed/unsigned conversion from type 'BYTE' to
type 'SINT'
--> SA0132: Implicit signed/unsigned conversion from type 'INT' to
type 'WORD'
--> SA0132: Implicit signed/unsigned conversion from type 'SINT' to
type 'LWORD'

```

\section*{See also}
- \({ }^{\wedge}\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0133: Explicit narrowing conversions}

Detects explicit conversions from a larger data type to a smaller data type
Justification: A large number of type conversions may indicate that you have chosen the wrong data types for variables. For this reason, there are programming guidelines that require an explicit justification for data type conversions.
Importance: Low
```

Example

```
```

PROGRAM SA0133

```
PROGRAM SA0133
VAR
VAR
    siVar:SINT;
    siVar:SINT;
    diVar:DINT;
    diVar:DINT;
        liVar:LINT;
        liVar:LINT;
        byVar:BYTE;
        byVar:BYTE;
        uiVar:UINT;
        uiVar:UINT;
        dwVar:DWORD;
        dwVar:DWORD;
        lwVar:LWORD;
        lwVar:LWORD;
        rVar:REAL;
        rVar:REAL;
        lrVar:LREAL;
        lrVar:LREAL;
END_VAR
END_VAR
siVar := LINT TO SINT(liVar); // SA0133
siVar := LINT TO SINT(liVar); // SA0133
byVar := DINT TO-BYTE(diVar); // SA0133
byVar := DINT TO-BYTE(diVar); // SA0133
siVar := DWORD_TO_SINT(dwVar); // SA0133
siVar := DWORD_TO_SINT(dwVar); // SA0133
uiVar := LREAL TO UINT(lrVar); // SA0133
uiVar := LREAL TO UINT(lrVar); // SA0133
rVar := LWORD TO \overline{REAL(lwVar); // SA0133}
rVar := LWORD TO \overline{REAL(lwVar); // SA0133}
--> SA0133: Explicit narrowing conversion from type 'LINT' to type
--> SA0133: Explicit narrowing conversion from type 'LINT' to type
'SINT'
'SINT'
--> SA0133: Explicit narrowing conversion from type 'DINT' to type
--> SA0133: Explicit narrowing conversion from type 'DINT' to type
'BYTE'
'BYTE'
--> SA0133: Explicit narrowing conversion from type 'DWORD' to type
--> SA0133: Explicit narrowing conversion from type 'DWORD' to type
'SINT'
'SINT'
--> SA0133: Explicit narrowing conversion from type 'LREAL' to type
--> SA0133: Explicit narrowing conversion from type 'LREAL' to type
'UINT'
'UINT'
--> SA0133: Explicit narrowing conversion from type 'LWORD' to type
--> SA0133: Explicit narrowing conversion from type 'LWORD' to type
'REAL'
```

'REAL'

```

\section*{See also}
- \& Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0134: Explicit signed/unsigned conversions}

Detects explicit conversions from signed data types to unsigned data types and the other way around

Justification: Excessive use of type conversions may indicate that you have chosen the wrong data types for variables. For this reason, there are programming guidelines that require an explicit justification for data type conversions.
Importance: Low
```

Example

```
```

PROGRAM PLC_PRG

```
PROGRAM PLC_PRG
VAR
VAR
    byVar :BYTE;
    byVar :BYTE;
    udiVar : UDINT;
    udiVar : UDINT;
    uliVar : ULINT;
    uliVar : ULINT;
    lwVar : LWORD;
    lwVar : LWORD;
    wVar : WORD;
    wVar : WORD;
    siVar : SINT;
    siVar : SINT;
    iVar : INT;
    iVar : INT;
    diVar : DINT;
    diVar : DINT;
    liVar : LINT;
    liVar : LINT;
END_VAR
END_VAR
liVar := ULINT TO LINT(uliVar);
liVar := ULINT TO LINT(uliVar);
udiVar := DINT TO- UDINT(diVar);
udiVar := DINT TO- UDINT(diVar);
siVar := BYTE TO_\overline{SINT(byVar);}
siVar := BYTE TO_\overline{SINT(byVar);}
wVar := INT TO WORD(iVar);
wVar := INT TO WORD(iVar);
lwVar := SINTT_\overline{TO_LWORD(siVar);}
lwVar := SINTT_\overline{TO_LWORD(siVar);}
--> SA0134: Explicit signed/unsigned conversion from type 'ULINT'
--> SA0134: Explicit signed/unsigned conversion from type 'ULINT'
to type 'LINT'
to type 'LINT'
--> SA0134: Explicit signed/unsigned conversion from type 'DINT' to
--> SA0134: Explicit signed/unsigned conversion from type 'DINT' to
type 'UDINT'
type 'UDINT'
--> SA0134: Explicit signed/unsigned conversion from type 'BYTE' to
--> SA0134: Explicit signed/unsigned conversion from type 'BYTE' to
type 'SINT'
type 'SINT'
--> SA0134: Explicit signed/unsigned conversion from type 'INT' to
--> SA0134: Explicit signed/unsigned conversion from type 'INT' to
type 'WORD'
type 'WORD'
--> SA0134: Explicit signed/unsigned conversion from type 'SINT' to
--> SA0134: Explicit signed/unsigned conversion from type 'SINT' to
type 'LWORD'
```

type 'LWORD'

```

\section*{See also}
- \& Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{Use of Direct Addresses}
\[
\text { 6.6.3.4.2.38.1 SA0005: Invalid addresses and data types.............................. } 4263
\]
6.6.3.4.2.38.2 SA0047: Accesses to direct address ..... 4264
6.6.3.4.2.38.3 SA0048: AT-declarations on direct addresses. ..... 4264

\section*{SA0005: Invalid addresses and data types}

Detects invalid addresses and data type specifications. Valid size prefixes in addresses: X for BOOL B for 1-byte data types, W for 2-byte data types, and D for 4-byte data types.

Justification: Variables located on direct addresses should preferably be associated with an address that corresponds to their data type width. It can be confusing for the reader of the code, for example, if a DWORD is assigned to a BYTE address.

Importance: Low

\section*{Example}
```

PROGRAM Check_Address_Type_PRG
VAR
iVar AT %QBO : INT ; // OK e. g.: %QWO
xTest AT %QW1 : BOOL ; // OK e. g.: %QX1.0
END_VAR
iVar := iVar + INT\#1;
xTest := NOT xTest;
--> SA0005: Invalid address for data type 'iVar'

```

\section*{See also}
- \& Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0047: Accesses to direct address}

Detects direct address access in the implementation code
Justification: Symbolic programming is always preferable. A variable has a name that can also have a meaning. An address cannot indicate what it is used for.
Importance: High

\section*{PLCopen rule: N1 / CP1}

\section*{Example}
```

PROGRAM PLC_PRG
VAR
xVar : BOOL;
byVar : BYTE;
END_VAR
xVar := %IX0.0;
%QXO.O := xVar;
%MX0.1 := xVar;
%MB1 := byVar;
--> Access to direct address '%IX0.0'
--> Access to direct address '%QX0.0'
--> Access to direct address '%MX0.1'
--> Access to direct address '%MB1'

```

\section*{See also}
- \& Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0048: AT-declarations on direct addresses}

\section*{Detects AT declarations on direct addresses}

Justification: The use of direct addresses in the code is problematic because the address then appears in multiple locations: first in the controller configuration where the assignment of a physical object to an address is defined, and second in the program where variables are assigned to these addresses. If the addresses are relocated because the configuration is changed, then you have to reassign variables to addresses at a completely different location in the program. This is a cause of error and results in poorer readability and maintainability of the code. Therefore, it is best to perform all assignments in the I/O mapping of the device editor.
Importance: High

PLCopen rule: N1 / CP1
Note: We recommend that you use direct addresses ONLY in the "//O Mapping" tab of the device editor.

\section*{Example}
```

PROGRAM PLC_PRG
VAR
xVar1 AT %IX0.0 : BOOL;
byVar1 AT %IB1 : BYTE;
xVar2 AT %QX0.0 : BOOL;
END_VAR
--> SA0048: Declaration uses direct address '%IXO.0'
--> SA0048: Declaration uses direct address '%IB1
--> SA0048: Declaration uses direct address '%QXO.0'

```

\section*{See also}
- \({ }^{4}\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{Rules for Operators}
6.6.3.4.2.39.1 SA0051: Comparison operations on BOOL variables ..... 4265
6.6.3.4.2.39.2 SA0052: Unusual shift operation ..... 4266
6.6.3.4.2.39.3 SA0053: Too big bitwise shift ..... 4266
6.6.3.4.2.39.4 SA0054: Comparisons of REAL/LREAL for equality / inequality ..... 4267
6.6.3.4.2.39.5 SA0055: Unnecessary comparisons of unsigned operands ..... 4268
6.6.3.4.2.39.6 SA0056: Constant out of valid range ..... 4268
6.6.3.4.2.39.7 SA0057: Possible loss of decimal places ..... 4269
6.6.3.4.2.39.8 SA0058: Operations on enumeration variables ..... 4269
6.6.3.4.2.39.9 SA0059: Comparison operations always returning TRUE or FALSE ..... 4271
6.6.3.4.2.39.10 SA0060: Zero used as invalid operand. ..... 4271
6.6.3.4.2.39.11 SA0061: Unusual operation on pointer. ..... 4271
6.6.3.4.2.39.12 SA0062: Uses of TRUE or FALSE in expressions. ..... 4272
6.6.3.4.2.39.13 SA0063: Possibly not 16-bit-compatible operations. ..... 4272
6.6.3.4.2.39.14 SA0064: Addition of pointer. ..... 4273
6.6.3.4.2.39.15 SA0065: Incorrect pointer addition to base size ..... 4273
6.6.3.4.2.39.16 SA0066: Uses of temporary results ..... 4274

\section*{SA0051: Comparison operations on BOOL variables}

Detects comparison operations on variables of type BOOL
Justification: CODESYS permits these kinds of comparison, but they are very unusual and can be confusing. The IEC 61131-3 standard does not provide for these comparisons. By avoiding them, you increase the portability of the code to other development systems.

Importance: Medium

\section*{Example}
```

PROGRAM PLC_PRG
VAR
xBool1, xBool2 : BOOL;
xResult : BOOL;
END VAR
xRe\overline{sult := xBool1 > xBool2; // SA0051}
xBool1 := NOT xBool1; // OK!
xBool2 := xBool2 XOR xBool1; // OK!
--> SA0051: Comparison operations on BOOL varables

```

\section*{See also}
- \& Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0052: Unusual shift operation}

Detects shift operations (bit shift) on signed variables. In the case of shift operations on bitfield data types (Byte, DWORD, LWORD, WORD), an error is not reported.

Justification: CODESYS permits shift operations on signed data types. However, these operations are unusual and can be confusing. The IEC 61131-3 standard does not provide for these kinds of operations. Therefore, they should be avoided in order to increase the portability of the code to other development systems.
Importance: Medium

\section*{Example}
```

PROGRAM PLC_PRG
VAR
iTemp : INT;
dwTemp1 : DWORD;
byTemp2 : BYTE;
diTemp3 : DINT;
siTemp4 : SINT;
liTemp5 : LINT;
END_VAR
//the following lines each will cause an SA0052:
iTemp := SHL(iTemp, BYTE\#2);
diTemp3 := SHR(diTemp3, BYTE\#4);
siTemp4 := ROL(siTemp4, BYTE\#2);
liTemp5 := ROR(liTemp5, BYTE\#2);
//no error SAO052 because DWORD and BYTE are bit field data types:
dwTemp1 := SHL(dwTemp1, BYTE\#3);
byTemp2 := SHR(byTemp2, BYTE\#1);
---> SAOO52: Unusual shift operation

```

\section*{See also}
- \& Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0053: Too big bitwise shift}

Detects whether or not the data type width of the operand has been exceeded in the case of a bitwise shift (bit shift) of operands

Justification: If a shift operation exceeds the data type width, then a constant 0 is generated. If a rotation shift exceeds the data type width, then it is difficult to read. Therefore, the rotation value should be shortened.
Importance: High

\section*{Example}
```

PROGRAM PLC_PRG
VAR
byTemp1 : BYTE;
wTemp2 : WORD;
dwTemp3 : DWORD;
lwTemp4 : LWORD;
END VAR
byTemp1 := SHR(byTemp1, BYTE\#25);
wTemp2 := SHL (wTemp2, BYTE\#45);
dwTemp3 := ROR(dwTemp3, BYTE\#78);
lwTemp4 := ROL(lwTemp4, BYTE\#111);
--> SA0053: Too big bitwise shift

```

\section*{See also}
- \({ }^{4}\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0054: Comparisons of REAL/LREAL for equality / inequality}

Detects whether or not the comparison operators \(=\) (equality) and \(<>\) (inequality) compare the operands of type REAL or LREAL

Justification: REAL/LREAL values are implemented as floating-point numbers according to the IEEE 754 standard. This standard implies that specific, apparently simple decimal numbers cannot be represented with precision. As a result, there may be different representations as LREAL for the same decimal number.

Consider the following lines of code:
```

lr11 := 1.1;
lr33 := 3.3;
lrVar1 := lr11 + lr11;
lrVar2 := lr33 - lr11;
botest := lrVar1 = lrVar2;

```

In this case, botest returns FALSE, even if the variables \(\operatorname{lrVar1}\) and \(\operatorname{lrVar} 2\) both return the monitoring value of 2.2. This is not an error of the compiler, but a property of the floating point units of all conventional processors. You can avoid this by specifying a minimum value by which the values may differ: botest \(:=\) ABS (lrVar1 - lrVar2) < 0.1 ;

Exception: A comparison with 0.0 is not reported by this analysis. For the 0 , there is an exact representation in the IEEE 754 standard, and therefore the comparison functions normally as expected. Therefore, for better performance, it makes sense to permit a direct comparison here.

Importance: High
PLCopen rule: CP54
```

Example
PROGRAM PLC_PRG
VAR
rTest1 : REAL;
rTest2 : REAL;
lrTest3 : LREAL;
lrTest4 : LREAL;
xResult : BOOL;
END_VAR
//the following lines each will cause an SA0054:
xResult := rTest1 = rTest1;
xResult := rTest1 = rTest2;
xResult := rTest1 <> rTest2;
xResult := lrTest3 = lrTest3;
xResult := lrTest3 = lrTest4;
xResult := lrTest3 <> lrTest4;
//the following lines each will not cause an SA0054:
xResult := rTest1 > rTest2;
xResult := lrTest3 < lrTest4;
--> SA0054: Comparisons of REAL/LREAL for equality / inequality

```

See also
- . Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0055: Unnecessary comparisons of unsigned operands}

Detects unnecessary comparisons with unsigned operands. An unsigned data type is never less than zero. This can be used as a sign check.

Justification: A comparison detected with this check yields a constant result and is an indication of an error in the code.
Importance: High
Example
```

PROGRAM PLC_PRG
VAR
byTest: BYTE;
END_VAR
WHILE byTest >= 0 DO
byTest := byTest - 1;
END_WHILE;
--> SA0055: Unnecessary comparisons of unsigned operands

```

\section*{See also}
- \({ }^{4}\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0056: Constant out of valid range}

Detects literals (constants) outside of the valid range of the operator
Justification: The message is issued in cases when a value is compared with a constant that is outside of the range of this value. Then the comparison constantly returns TRUE or FALSE. This is an indication of a programming error.
```

Importance: High

```

\section*{Example}
```

PROGRAM PLC_PRG
VAR
byTestVar: BYTE;
END_VAR
WHILE byTestVar >= 260 DO
byTestVar := byTestVar + 1;
END_WHILE
--> SA0056: Constant out of valid range

```

\section*{See also}
- \({ }^{4}\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0057: Possible loss of decimal places}

Detects statements with possible loss of decimal places
Justification: A piece of code of the following type (diTemp2 := 1 rTemp1 := TO_REAL (diTemp2 / DINT\#2)) can cause a misinterpretation. The author or reader of this line of code can assume that the division would be performed as a REAL operation, and in this case the result would be REAL\#0.5. However, this is not true. It is an integer operation. The result is cast to REAL and rTemp1 gets the value REAL\# 0 . To avoid this, use a cast to make sure that the operation is performed as a REAL operation: rTemp1 := TO_REAL (diTemp2) / REAL\#2.

Importance: Medium

\section*{Example}
```

PROGRAM PLC_PRG
VAR
rTemp1 : REAL;
diTemp2 : DINT;
liTemp3 : LINT;
END_VAR
diTemp2 := diTemp2 + DINT\#11;
rTemp1 := DINT_TO_REAL(diTemp2 / DINT\#3); // SA0057
rTemp1 := DINT_TO_REAL(diTemp2) / REAL\#3.0;
liTemp3 := liTemp\overline{3 + LINT\#13;}
rTemp1 := LINT_TO_REAL(liTemp3 / LINT\#7); // SA0057
rTemp1 := LINT_TO_REAL(liTemp3) / REAL\#7.0;
--> SA0057: Possible loss of decimal places

```

See also
- \& Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0058: Operations on enumeration variables}

Detects operations on variables of the enumeration data type Assignments are permitted.
Justification: Enumerations should not be used as ordinary integer values. You can also define an alias data type or use a subrange type.

Importance: Medium
Exception: If an enumeration is tagged with the pragma \{attribute 'strict'\}, then the compiler already reports this kind of operation.
If an enumeration is declared as a flag by the pragma \{attribute 'flags'\}, then an error is not issued for AND, OR, NOT, or oder XOR operations.

\section*{Example}
```

TYPE My_Enum :
(
red := 1, blue := 2, green := 3, black := 4
);
END_TYPE
PROGRAM PLC_PRG
VAR
iTemp1 : INT;
abc : My_Enum;
END_VAR
iTemp1 := iTemp1 + INT\#1;
abc := My_Enum.red; // OK
iTemp1 := My_Enum.black / My_Enum.blue; // SA0058
iTemp1 := My_Enum.green / My_Enum.red; // SA0058
--> SA0058: Operations on enumeration variables

```
```

Example with a
pragma
{attribute
'flags'}

```
```

{attribute 'flags'} // declaring the enumeration as a "flag"

```
{attribute 'flags'} // declaring the enumeration as a "flag"
TYPE Flags :
TYPE Flags :
(
(
        Unknown := 16#00000001,
        Unknown := 16#00000001,
        Stopped := 16#00000002,
        Stopped := 16#00000002,
        Running := 16#00000004
        Running := 16#00000004
) DWORD;
) DWORD;
END_TYPE
END_TYPE
PROGRAM PLC_PRG
PROGRAM PLC_PRG
VAR
VAR
    iTemp1 : INT;
    iTemp1 : INT;
    abc : Flags;
    abc : Flags;
        batate : BYTE;
        batate : BYTE;
        dwFlags : DWORD;
        dwFlags : DWORD;
        dwState : DWORD;
        dwState : DWORD;
END_VAR
END_VAR
// OK for the following
// OK for the following
IF (dwFlags AND Flags.Unknown) <> DWORD#O THEN
IF (dwFlags AND Flags.Unknown) <> DWORD#O THEN
    dwState := dwState AND Flags.Unknown;
    dwState := dwState AND Flags.Unknown;
ELSIF (dwFlags OR Flags.Stopped) <>DWORD#O THEN
ELSIF (dwFlags OR Flags.Stopped) <>DWORD#O THEN
    dwState := dwState OR Flags.Running;
    dwState := dwState OR Flags.Running;
END_IF
```

END_IF

```

\section*{See also}
- \(\Rightarrow\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0059: Comparison operations always returning TRUE or FALSE}

Detects comparisons with literals that always have the result TRUE or FALSE, and can already be processed during at the compile.
Justification: An operation that consistently yields TRUE or FALSE is an indication of a programming error.

Importance: High

\section*{Example}
```

PROGRAM PLC_PRG
VAR
byTemp1 : BYTE;
END_VAR
WHILE byTemp1 <= 255 DO
byTemp1 := byTemp1 + 1;
END_WHILE;
--> SAO059: Relational operator '<=' always evaluates 'TRUE'

```

\section*{See also}
- \(\Rightarrow\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0060: Zero used as invalid operand}

Detects operations where an operand with the value " 0 " causes an invalid or a nonsense operation

Justification: This kind of expression could be an indication of a programming error. In any case, it unnecessarily wastes runtime.

Importance: Medium
```

Example

```
```

PROGRAM PLC_PRG

```
PROGRAM PLC_PRG
VAR
VAR
    byTemp1 : BYTE;
    byTemp1 : BYTE;
    wTemp2 : WORD;
    wTemp2 : WORD;
            dwTemp3 : DWORD;
            dwTemp3 : DWORD;
END_VAR
END_VAR
byTemp1 := byTemp1 + 0;
byTemp1 := byTemp1 + 0;
wTemp2 := wTemp2 - WORD#0;
wTemp2 := wTemp2 - WORD#0;
dwTemp3 := dwTemp3 * DWORD#0;
dwTemp3 := dwTemp3 * DWORD#0;
--> SA0060: Zero used as invalid operand
```

--> SA0060: Zero used as invalid operand

```

See also
- \({ }^{4}\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0061: Unusual operation on pointer}

Detects operations one variables of type POINTER TO which are not = (equality), <> (inequality), + (addition), or ADR.

In CODESYS, pointer arithmetic is generally permitted and can also be used appropriately. Therefore, the addition of a pointer with an integer value is considered a common operation on pointers. This makes it possible to use a pointer to process an array of variable length. All other (unusual) operations with pointers are reported with SA0061.

Importance: High
PLCopen rule: E2 / E3

\section*{Example}
```

PROGRAM PLC_PRG
VAR
piTemp : POINTER TO INT;
iTemp : INT;
END_VAR
iTemp := iTemp + INT\#1;
piTemp := ADR(iTemp);
piTemp := piTemp * DWORD\#5; // SA0061
piTemp := piTemp / DWORD\#2; // SA0061
piTemp := piTemp MOD DWORD\#3; // SA0061
piTemp := piTemp + DWORD\#1;
piTemp := piTemp - DWORD\#1; // SA0061
--> SA0061: Unusual operation on pointer

```

\section*{See also}
- \(y^{*}\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0062: Uses of TRUE or FALSE in expressions}

Detects the use of the literals TRUE or FALSE in expressions
Justification: This kind of expression is obviously unnecessary and may indicate an error. In any case, the expression unnecessarily affects the runtime.
Importance: Medium

\section*{Example}
```

PROGRAM PLC_PRG
VAR
xTemp1, xTemp2 : BOOL;
END_VAR
xTemp1 := xTemp1 AND NOT TRUE;
xTemp2 := xTemp1 OR TRUE;
xTemp2 := xTemp1 OR NOT FALSE;
xTemp2 := xTemp1 AND FALSE;
--> Uses of TRUE or FALSE in expressions

```

See also
- \& Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0063: Possibly not 16-bit-compatible operations}

Detects 16-bit operations with temporary results. Background: On 16-bit systems, 32-bit temporary results can be truncated. Example: (int+10) can exceed 16 bits.
Justification: In the very rare case that you have to write code which should run on a 16-bit processor as well as on a 32-bit processor, this message should help to prevent any problems.

Importance: Low

\section*{Example}
```

PROGRAM PLC_PRG
VAR
iVar : INT;
END VAR
iVar := (iVar + 10) / 2;
--> SA0063: Compatibility for 16 Bit - Possible truncated
intermediate result

```

See also
- \({ }^{\text {® }}\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0064: Addition of pointer}

Detects the addition of pointers
Justification: In CODESYS, pointer arithmetic is generally permitted and can also be used appropriately. However, it is also a source of errors. Therefore, programming rules exist that generally prohibit pointer arithmetic. This test can check such a requirement.
Importance: Medium

\section*{Example}
```

PROGRAM PLC_PRG
VAR
iTest:INT;
ariTest:ARRAY[0..10] OF INT;
{attribute 'analysis':='-111'}
piTest:POINTER TO INT;
i:INT;
END_VAR
piTest := ADR(ariTest[0]); // OK
piTest^:= 0;
piTest := ADR(ariTest) + SIZEOF(INT); // SA0064
piTest^:= 1;
piTest := ADR(ariTest) + 6; // SA0064
piTest^:= 3;
piTest := ADR(ariTest[10]);
FOR i:=0 TO 10 DO
piTest^ := i;
piTest := piTest + 2; // SA0064
END_FOR
--> SA0064: Addition of pointer

```

\section*{See also}
- \({ }^{4}\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0065: Incorrect pointer addition to base size}

Detects pointer additions for which the value to be added does not match the base size of the pointer. Only literals of the base size can be added. Also multiplication products of the base size cannot be added.

Justification: In CODESYS (in contrast to C and \(\mathrm{C}++\) ), when adding a pointer with an integer value, only this integer value is added as the number of bytes, and not the integer value multiplied by the base size. Example in ST:
```

pINT := ADR(array_of_int[0])
pINT := pINT + 2 ; // In CODESYS, pINT then points to
array_of_int[1]

```

This code would function differently in C :
```

short* pShort
pShort = \&(array of short[0])
pShort = pShort + 2; // In C, pShort then points to array_of_short[2]

```

Therefore, in CODESYS, you should always add a multiple of the base size of the pointer to a pointer. Otherwise, the pointer may point to non-aligned memory which (depending on the processor) can lead to an alignment exception when accessing it.
Importance: High

\section*{Example}
```

VAR
pudiTest:POINTER TO UDINT;
udiTest:UDINT;
prTest:POINTER TO REAL;
rTest:REAL;
END_VAR
pudiTest := ADR(udiTest) + 4; // OK
pudiTest := ADR(udiTest) + ( 2 + 2 ); // OK
pudiTest := ADR(udiTest) + SIZEOF(UDINT); // OK
pudiTest := ADR(udiTest) + 3; // SA0065
pudiTest := ADR(udiTest) + 2*SIZEOF (UDINT); // SA0065
pudiTest := ADR(udiTest) + ( 3 + 2 ); // SA0065
prTest := ADR(rTest);
prTest := prTest + 4; // OK
prTest := prTest + ( 2 + 2 ); // OK
prTest := prTest + SIZEOF(REAL); // OK
prTest := prTest + 1; // SA0065
prTest := prTest + 2; // SA0065
prTest := prTest + 3; // SA0065
prTest := prTest + ( SIZEOF(REAL) - 1 ); // SA0065
prTest := prTest + ( 1 + 4 ); // SA0065
--> SA0065: Incorrect pointer addition to base size

```

See also
- \({ }^{4}\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0066: Uses of temporary results}

Detects the use of temporary results in statements with a data type that is less than the registry size. The implicit cast in this case may lead to unwanted results.

Justification: For performance reasons, CODESYS performs operations on the register width of the processor. Intermediate results are not truncated. This can lead to misinterpretations as in the following case: usintTest \(:=0\); xError \(:=\) usintTest - \(1<>255\); In CODESYS, xError is TRUE in this case because the operation usintTest - 1 is typically executed as a 32-bit operation and the result is not cast to the byte size. Then the value \(16 \# f f f f f f f f\) (not equal to 255 ) is located in the registry. To avoid this, you have to cast the intermediate result explicitly: xError := TO_USINT (usintTest - 1) <> 255;

\section*{NOTICE!}

If this message is activated, then many less problematic locations in the code will be reported. Although a problem can only occur when the operation produces an overflow or underflow in the data type, the static analysis cannot differentiate between the individual locations.

If you include an explicit typecast in all reported locations, then the code will be much slower and less readable.

Importance: Low

\section*{Example}
```

PROGRAM PLC_PRG
VAR
byTest:BYTE;
liTest:LINT;
xError:BOOL;
END_VAR
//type size smaller than register size;
byTest := 0;
IF (byTest - 1) <> 255 THEN //use of temporary result + implicit
casting -> SA0066
xError := TRUE;
ELSE
xError := FALSE;
END_IF
//type size equal to or bigger than register size;
liTest := 0;
IF (liTest - 1) <> -1 THEN // use of temporary result and no
implicit casting -> OK
xError := TRUE;
ELSE
xError := FALSE;
END_IF
--> SA0066: Use of temporary result: (byTest - USINT \#1)

```

See also
- \({ }^{4}\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{Rules for Statements}
6.6.3.4.2.40.1 SA0072: Invalid uses of counter variable ..... 4276
6.6.3.4.2.40.2 SA0073: Uses of inadequate counter variable. ..... 4276
6.6.3.4.2.40.3 SA0080: Loop index variable for array index exeeds array range ..... 4276
6.6.3.4.2.40.4 SA0081: Upper border is not a constant. ..... 4277
6.6.3.4.2.40.5 SA0075: Missing ELSE. ..... 4278
6.6.3.4.2.40.6 SA0076: Missing enumeration constant ..... 4279
6.6.3.4.2.40.7 SA0077: Type mismatches with CASE expression. ..... 4280
6.6.3.4.2.40.8 SA0078: Missing CASE branches. ..... 4280
6.6.3.4.2.40.9 SA0090: Return statement before end of function. ..... 4281

\section*{SA0072: Invalid uses of counter variable}

Detects the use of a counter variable in a FOR loop
Justification: Manipulation of the counter variable in a FOR loop can easily result in an infinite loop. To prevent the execution of the loop for specific values of the counter variable, use CONTINUE or simply an IF.
Importance: High
PLCopen rule: L12

\section*{Example}
```

PROGRAM PLC_PRG
VAR_TEMP
iIndex : INT;
END VAR
FOR iIndex := INT\#O TO INT\#2O BY INT\#1 DO
iIndex := iIndex - INT\#1;
END_FOR
--> SA0072: Invalid use of counter variable 'iIndex'

```

See also
- «Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0073: Uses of inadequate counter variable}

Detects the use of non-temporary variables in FOR loops.
Justification: This is a performance warning. A counter variable is always initialized each time a POU is called. You can create this variable as a temporary variable (VAR_TEMP). Access to it may be faster and the variable does not take up any permanent memory.
Importance: Medium
PLCopen rule: CP21 / L13

\section*{Example}
```

PROGRAM PLC_PRG
VAR
nIndex : INT;
iVar : INT;
END VAR
FOR nIndex := INT\#0 TO INT\#20 BY INT\#1 DO
iVar := iVar + nIndex;
END_FOR
--> SA0073: Inadequate counter variable

```

\section*{See also}
- \& Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0080: Loop index variable for array index exeeds array range}

Detects the FOR statements where the index variable is used to access an array index and exceeds the range of the array index

Justification: Arrays are typically processed in FOR loops. The start and end value of the counter variable should typically match (or at least not exceed) the upper and lower bounds of the array. A typical cause of error is detected here when array bounds are changed and constants are not carefully used; or when a different value is used accidentally in the FOR loop than in the array declaration.

Importance: High
```

Example

```
```

PROGRAM PLC_PRG

```
PROGRAM PLC_PRG
VAR
VAR
    iIndex1,iIndex2,iIndex3 : INT;
    iIndex1,iIndex2,iIndex3 : INT;
    arWord : ARRAY[1..100] OF WORD;
    arWord : ARRAY[1..100] OF WORD;
    arararINT : ARRAY[1..9,1..9,1..9] OF INT;
    arararINT : ARRAY[1..9,1..9,1..9] OF INT;
    arUSINT : ARRAY[0..99] OF USINT;
    arUSINT : ARRAY[0..99] OF USINT;
END_VAR
END_VAR
//1 violation of the rule(lower range is exeeded): SA0080
//1 violation of the rule(lower range is exeeded): SA0080
FOR iIndex1 := INT#O TO INT#100 DO
FOR iIndex1 := INT#O TO INT#100 DO
            arWord[iIndex1] := INT_TO_WORD(iIndex1);
            arWord[iIndex1] := INT_TO_WORD(iIndex1);
END_FOR
END_FOR
//6 violations (lower and upper range is exceeded for each array
//6 violations (lower and upper range is exceeded for each array
dimension): 3SA0080
dimension): 3SA0080
FOR iIndex2 := INT#0 TO INT#10 DO
FOR iIndex2 := INT#0 TO INT#10 DO
        arararINT[iIndex2, iIndex2, iIndex2] := iIndex2;
        arararINT[iIndex2, iIndex2, iIndex2] := iIndex2;
END_FOR
END_FOR
//l violation (upper range is exceeded by the end result of the
//l violation (upper range is exceeded by the end result of the
index), previous expressions on index are not evaluated -> OK
index), previous expressions on index are not evaluated -> OK
FOR iIndex3 := INT#0 TO INT#50 DO
FOR iIndex3 := INT#0 TO INT#50 DO
        arUSINT[iIndex3 * INT#2] := INT_TO_USINT(iIndex3);
        arUSINT[iIndex3 * INT#2] := INT_TO_USINT(iIndex3);
END_FOR
END_FOR
--> SA0080: Loop index range of 'Index1' exceeds array range
--> SA0080: Loop index range of 'Index1' exceeds array range
--> SA0080: Loop index range of 'Index2' exceeds array range
```

--> SA0080: Loop index range of 'Index2' exceeds array range

```

\section*{See also}
- \(\Leftrightarrow\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0081: Upper border is not a constant}

Detects the FOR statements where the upper bound is not defined with a constant value Justification: If the upper bound of a loop is a variable value, then it is no longer possible to see how often a loop is executed. This can result in serious problems at runtime. The worst case is an infinite loop.
Importance: High

\section*{Example}
```

PROGRAM PLC_PRG
VAR
i:INT;
iBorder1: INT := 10;
iBorder2: INT := 10;
iCounter: INT;
END VAR
VAR ' CONSTANT
ciBorder:INT := 10;
END_VAR
FOR i:=0 TO 10 DO //OK
iCounter := i;
END_FOR
FOR i:=O TO ciBorder DO // OK
iCounter := i;
END_FOR
FOR i:=0 TO iBorder1 DO // SA0081
iCounter := i;
END_FOR
FOR i:=0 TO iBorder2 DO // SA0081
iCounter := i;
IF iCounter = 10 THEN
iBorder2 := 50;
END_IF
END_FOR
--> SA0081: Upper border of a for loop must be a constant value

```

\section*{See also}
- « Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0075: Missing ELSE}

Detects CASE statements without an ELSE branch
Justification: Defensive programming requires the inclusion of an ELSE branch in every CASE statement. If there is nothing to do in the ELSE branch, then include a comment to indicate this. It is then clear to the reader of the code that the case was not simply forgotten.
Importance: Low
PLCopen rule: L17
```

Example
PROGRAM PLC_PRG
VAR
iVar : INT;
xTemp : BOOL;
END_VAR
iVar := iVar + INT\#1;
CASE iVar OF
INT\#1:
xTemp := FALSE;
INT\#2:
xTemp := TRUE;
END_CASE
--> SA0075: Missing ELSE in CASE statement

```

\section*{See also}
- «Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0076: Missing enumeration constant}

Detects whether or not an enumeration variable is used as a condition and not all enumeration values are treated as CASE branches

Justification: Defensive programming requires the processing of all possible values of an enumeration. If an action is not required for a particular enumeration value, then you should add a comment to indicate this explicitly. It is then clear to the reader of the code that the value was not simply forgotten.
Importance: Low

\section*{Example}
```

TYPE My_Enum :
(
red := 1, blue := 2, green := 3, black := 4
);
END_TYPE
PROGRAM PLC_PRG
VAR
iVar : My_Enum;
xTemp : BOOOL;
END_VAR
iVa\overline{r := My_Enum.black;}
CASE iVar OF
My_Enum.red:
xTemp := FALSE;
My_Enum.blue, My_Enum.green:
xTemp := TRUE;
ELSE
xTemp := NOT xTemp;
END_CASE
--> SA0076: Missing enumeration constant 'black' in CASE statement

```

See also
- \#Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0077: Type mismatches with CASE expression}

Detects code locations where the data type of a condition does not match that of the CASE branch

Justification: If the data types between the CASE variable and the CASE itself do not match, then this could indicate an error.
Importance: Low

\section*{Example}
```

TYPE My_Enum :
(
eins := 1, zwei := 2, drei := 3, vier := 4
);
END_TYPE
PROGRAM PLC_PRG
VAR
diVar : DINT;
xTemp : BOOL;
END VAR
diVar := diVar + DINT\#1;
CASE diVar OF
DINT\#1:
xTemp := FALSE;
My_Enum.zwei, DINT\#3: //SA0077
xTemp := TRUE;
ELSE
xTemp := NOT xTemp;
END_CASE
--> SAOO77: Type mismatches with CASE expression

```

\section*{See also}
- \#Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0078: Missing CASE branches}

Detects CASE statements without CASE branches and only one ELSE statement Justification: A CASE statement without cases wastes execution time and it is difficult to read. Importance: Medium
```

Example

```
```

PROGRAM PLC_PRG

```
PROGRAM PLC_PRG
VAR
VAR
    iVar : INT;
    iVar : INT;
    xTemp : BOOL;
    xTemp : BOOL;
END_VAR
END_VAR
iVar := iVar + INT#1;
iVar := iVar + INT#1;
//in the following the case descriptions are missing:
//in the following the case descriptions are missing:
CASE iVar OF
CASE iVar OF
    ELSE
    ELSE
            xTemp := NOT xTemp;
            xTemp := NOT xTemp;
END_CASE
END_CASE
--> SA0078: CASE-Missing CASE branches
```

--> SA0078: CASE-Missing CASE branches

```

See also
- \(\Rightarrow\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0090: Return statement before end of function}

Detects whether or not the RETURN statement is not the last statement in a function, method, property, or program.

Justification: A RETURN in the code results in worse maintainability, testability, and readability of the code. A RETURN in the code is easily overlooked. Before each RETURN, it is often forgotten to insert code that should always be executed when exiting a function.

Importance: Medium
PLCopen rule: CP14

\section*{Example}
```

FUNCTION FUN : DINT
VAR INPUT
bTest : BOOL;
END_VAR
IF bTest THEN
RETURN;
END IF
FUN := 99;
--> SA0090: Return statement before end of function

```

\section*{See also}
- \({ }^{4}\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0095: Assignments in conditions}

Detects assignments in conditions of IF, CASE, or REPEAT constructs
Justification: An assignment (:=) and a comparison (=) can easily be mistaken. As a result, an assignment in a condition can easily be unintentional, and it is therefore reported. This can also confuse the reader of the code.

Importance: High
```

Example

```
```

PROGRAM PLC_PRG

```
PROGRAM PLC_PRG
VAR
VAR
    iCond1:INT := INT#1;
    iCond1:INT := INT#1;
    iCond2:INT := INT#2;
    iCond2:INT := INT#2;
    xCond:BOOL := FALSE;
    xCond:BOOL := FALSE;
    iVar : INT;
    iVar : INT;
END_VAR
END_VAR
IF INT_TO_BOOL(iCond1 := iCond2) THEN // SA0095
IF INT_TO_BOOL(iCond1 := iCond2) THEN // SA0095
    iCond\overline{1}}:= INT#1
    iCond\overline{1}}:= INT#1
    iCond2 := INT#2;
    iCond2 := INT#2;
ELSIF (iCond1 := 11) = 11 THEN // SA0095
ELSIF (iCond1 := 11) = 11 THEN // SA0095
    iCond1 := INT#1;
    iCond1 := INT#1;
    iCond2 := INT#2;
    iCond2 := INT#2;
END_IF
END_IF
IF xCond := TRUE THEN // SA0095
IF xCond := TRUE THEN // SA0095
    xCond := FALSE;
    xCond := FALSE;
END_IF
END_IF
IF (xCond := FALSE) OR (iCond1 := iCond2) = 12 THEN // SA0095
IF (xCond := FALSE) OR (iCond1 := iCond2) = 12 THEN // SA0095
    xCond := FALSE;
    xCond := FALSE;
    iCond1 := INT#1;
    iCond1 := INT#1;
    iCond2 := INT#2;
    iCond2 := INT#2;
END_IF
END_IF
IF (iVar := iVar + 1) = 120 THEN
IF (iVar := iVar + 1) = 120 THEN
/ /
/ /
SA0095 (can be valid, but is not reparable very well
SA0095 (can be valid, but is not reparable very well
iVar := 0;
iVar := 0;
END_IF
END_IF
WHILE (xCond = TRUE) OR (iCond1 := iCond2) = 12 DO // SA0095
WHILE (xCond = TRUE) OR (iCond1 := iCond2) = 12 DO // SA0095
    xCond := FALSE;
    xCond := FALSE;
END_WHILE
END_WHILE
//Error: assignment in repeat loop
//Error: assignment in repeat loop
REPEAT
REPEAT
    xCond := FALSE;
    xCond := FALSE;
UNTIL
UNTIL
    (xCond = TRUE) OR (iCond1 := iCond2) = 12 //
    (xCond = TRUE) OR (iCond1 := iCond2) = 12 //
SA0095
SA0095
END_REPEAT
END_REPEAT
--> SA0095: Assignment in condition: '...'
```

--> SA0095: Assignment in condition: '...'

```

\section*{See also}
- \#Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

SA0100: Variables greater than <n> bytes...
Detects variables that use more than n bytes, where n is defined by the current configuration. Default value: 1024 bytes. The value can be changed by double-clicking the line.
Justification: Some programming guidelines specify a maximum size for a single variable. This can be checked with this.
Importance: Low

\section*{Example}
```

PROGRAM PLC_PRG
VAR
aobyTest : ARRAY [0..1024] OF BYTE;
END_VAR
aobyTest[INT\#0] := aobyTest[INT\#0] + BYTE\#1;
--> SA0100: Variable 'aobyTest' greater 1024 bytes

```

\section*{See also}
- \& Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0101: Names with invalid length}

Detects names with invalid lengths. In the "Project Settings", double-click the rule entry to open a dialog where you can define the length of the name and define any exception.
Justification: Some programming guidelines specify a minimum length for variable names. This analysis can be used to check compliance.
Importance: Low
PLCopen rule: N6

\section*{Example}
```

PROGRAM PLC1
VAR
iVar1: INT; // SA0101
END_VAR
--> SA0101: Incorrect length of name 'PLC1'

```

\section*{See also}
- \& Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0102: Access to program/fb variables from the outside}

Detects external access to local variables of programs or function blocks
Justification: CODESYS permits external read access to local variables of programs or function blocks. This contradicts the principle of data encapsulation (hiding data) and does not comply with the IEC 61131-3 standard.

Importance: Medium
```

Example
PROGRAM PLC_PRG
VAR
iCounter : INT;
afb_Instance : AFB;
bfb_Instance : BFB;
END_VAR
iCounter := A PRG.iLocal; // SA0102
iCounter := b\overline{fb_Instance.iLocal; // SA0102}
A_PRG();
FUNCTION BLOCK AFB
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
iLocal: INT;
END_VAR
MET\overline{HOD METH : INT}
VAR_INPUT
END_VAR
iLocal := iLocal + 1;
FUNCTION_BLOCK BFB EXTENDS AFB
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
END_VAR
METHOD METH : INT
VAR_INPUT
END_VAR
iLocal := iLocal + 1;
PROGRAM A_PRG
VAR
iLocal: INT;
END_VAR
iLocal := iLocal + 1;
--> SA0102: Access to program/fb variable 'iLocal' from the outside

```

\section*{See also}
- « Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0103: Concurrent access on not atomic data}

Detects whether or not non-atomic variables (for example, with data type STRING, WSTRING, ARRAY, STRUCT, FB instances, 64-bit data types) are used in more than one task
Justification: When there is no synchronization during access, inconsistent values can be read when reading in one task and writing in another task at the same time.
Importance: Medium

For some data types, especially 64-bit integers, it depends on the platform whether or not access is atomic. Static analysis reports a problem only when the controller does not support atomic access to 64-bit integer data types.

This rule does not apply in the following cases:
- If the target system has a floating point unit (FPU), then access of multiple tasks to LREAL variables is not detected
- If the target system is a 64-bit processor or the corresponding target setting is set for the target device, then the rule does not apply to 64-bit data types
```

Example The project contains both programs, PRG1 and PRG2: The program PRG1 is called by the task
MainTask_1. The program PRG2 is called by the task MainTask_2.
GVL
VAR_GLOBAL
lrTest : LREAL; // Since the target system has an FPU, SA0103
does apply.
lint1 : LINT;
sTest : STRING; // SA0103
wsTest : WSTRING; // SA0103
END_VAR
PROGRAM PRG1
GVL.lrTest := 5.0;
GVL.sTest := 'welt';
GVL.wsTest := "welt";
GVL.lint1 := 99;
PROGRAM PRG2
GVL.lrTest := 5.0;
GVL.sTest := 'hallo';
GVL.wsTest := "hallo";
GVL.lint1 := 88;
--> SA0103: Concurrent access on not atomic data 'sTest'
--> SA0103: Concurrent access on not atomic data 'wsTest'

```

\section*{See also}
- \({ }^{*}\) Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0105: Multiple instance calls}

Detects the instances of function blocks that are called multiple times. To do this, the function blocks haves to be marked with the pragma \{attribute 'analysis:report-multiple-instance-calls'\}.
Justification: Some function blocks are designed in such as way that they can be called only one time in the cycle. This test checks whether or not a call is made in multiple locations.
Importance: Low
PLCopen rule: CP16 / CP20

\section*{Example}
```

// {attribute 'analysis:report-multiple-instance-calls'} Deactivated
FUNCTION_BLOCK FB_DOA
VAR_INPUT
END - VAR
VAR_OUTPUT
END_VAR
VAR
iA : INT;
END_VAR
iA \overline{:= iA + 1;}
{attribute 'analysis:report-multiple-instance-calls'}
FUNCTION_BLOCK FB_DoB
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
iB : INT;
END_VAR
iB := iB +1;
PROGRAM PLC_PRG
VAR
fbA : FB_DoA;
fbB : FB_DoB;
ND_VAR
fbA();
fbB(); // SA0105
fbA();
fbB(); // SA0105
--> SA0105: Instance 'fbB' called more than once

```

See also
- Chapter 6.6.3.4.1.6 "Attribute 'analysis:report-multiple-instance-calls"' on page 4231

\section*{SA0106: Virtual method calls in FB_INIT}

Detects method calls in the FB_Init method of a base function block, which are overwritten by a function block derived from a base function block
Justification: In these cases, it could be that the variables in the overwritten methods are not initialized in the base FB.

Importance: High
```

Example The function block FB_A includes the methods FB_Init and Meth_MyInit. FB_Init calls
Meth_MyInit for initialization. The function block FB_B is derived from FB_A. PLC_PRG
calls FB_B and therefore uses its mbMyDintB variable before it has been initialized.
FB_B.Meth_MyInit overwrites FB_A.Meth_MyInit.
FUNCTION_BLOCK FB_A
VAR
mbMyDintA : DINT;
END_VAR FUNCTION_BLOCK FB_B EXTENDS FB_A
VAR
mbMyDintB : DINT;
END_VAR METHOD FB_Init : BOOL
VAR_INPUT
bInitRetains:BOOL;
bInCopyCode:BOOL;
END_VAR
VAR
diDummy:DINT; // SA0106
END_VAR
mbMȳDintA := 123;
diDummy := Meth_MyInit(); METHOD Meth_MyInit : DINT
VAR_INPUT
END_VAR
mbMȳDintB := 123; // access to member of FB_B PROGRAM PLC_PRG
VAR
g_BInst : FB_B;
xVar-
END_VAR
xVa\overline{r := g_BInst.fb_init(TRUE, TRUE);}
//this instruction causes the following order of initializations:
//FB_A.fb_init
//FB_B.Me\overline{th_MyInit // SA0106}
//FB_B.fb_init
//FB_B.Meth_MyInit
--> SA0106: Virtual method call 'Meth_MyInit' in FB_INIT

```

See also
- \({ }^{2}\) Chapter 6.6.3.4.1.6 "Attribute 'analysis:report-multiple-instance-calls"' on page 4231

\section*{SA0107: Missing formal parameters}

Detects whether or not formal parameters are missing Justification: Code becomes more readable when formal parameters are specified in the call. Importance: Low
```

Example

```
```

FUNCTION FUNA : BOOL

```
FUNCTION FUNA : BOOL
VAR_INPUT
VAR_INPUT
    bDO: BOOL;
    bDO: BOOL;
    bInit: BOOL;
    bInit: BOOL;
    bManual : BOOL;
    bManual : BOOL;
END_VAR
END_VAR
VAR
VAR
    iInit: INT;
    iInit: INT;
    iLocal: INT;
    iLocal: INT;
    iManual: INT;
    iManual: INT;
END_VAR
END_VAR
IF bInit = TRUE THEN
IF bInit = TRUE THEN
    iInit := iInit + 1;
    iInit := iInit + 1;
END_IF
END_IF
IF \overline{bDo = TRUE THEN}
IF \overline{bDo = TRUE THEN}
    iLocal := iLocal + 1;
    iLocal := iLocal + 1;
END_IF
END_IF
IF \overline{bManual = TRUE THEN}
IF \overline{bManual = TRUE THEN}
    iManual:= iManual + 1;
    iManual:= iManual + 1;
END IF
END IF
FUNA := TRUE;
FUNA := TRUE;
PROGRAM PLC_PRG
PROGRAM PLC_PRG
VAR
VAR
END_VAR
END_VAR
FUNA(bInit := TRUE, bDo := TRUE, bManual := FALSE); // OK
FUNA(bInit := TRUE, bDo := TRUE, bManual := FALSE); // OK
FUNA(TRUE, TRUE, bManual:= FALSE);
FUNA(TRUE, TRUE, bManual:= FALSE);
--> SA0107: Missing formal parameter for input 'TRUE'
```

--> SA0107: Missing formal parameter for input 'TRUE'

```

\section*{See also}
- \({ }^{*}>\) Chapter 6.6.3.4.1.6 "Attribute 'analysis:report-multiple-instance-calls"" on page 4231

\section*{Checking Strict IEC Rules}
6.6.3.4.2.49.1 SA0111: Pointer variables ..... 4289
6.6.3.4.2.49.2 SA0112: Reference variables. ..... 4289
6.6.3.4.2.49.3 SA0113: Variables with data type WSTRING. ..... 4289
6.6.3.4.2.49.4 SA0114: Variables with data type LTIME ..... 4290
6.6.3.4.2.49.5 SA0115: Variables with data type UNION. ..... 4290
6.6.3.4.2.49.6 SA0117: Variables with data type BIT ..... 4290
6.6.3.4.2.49.7 SA0119:Object-oriented features. ..... 4291
6.6.3.4.2.49.8 SA0120: Program calls ..... 4291
6.6.3.4.2.49.9 SA0121: Missing VAR_EXTERNAL declarations ..... 4292
6.6.3.4.2.49.10 SA0122: Array index defined as expression ..... 4293
6.6.3.4.2.49.11 SA0123: Usages of INI, ADR or BITADR ..... 4293
6.6.3.4.2.49.12 SA0147: Unusual shift operation - strict. ..... 4293
6.6.3.4.2.49.13 SA0148: Unusual bit access - strict. ..... 4294
6.6.3.4.2.49.14 SA0118: Initialisations not using constants. ..... 4295
6.6.3.4.2.49.15 SA0124: Pointer dereferences in declarations. ..... 4295
6.6.3.4.2.49.16 SA0125: References in initializations. ..... 4295

\section*{SA0111: Pointer variables}

Detects variables of type POINTER TO
Justification: The IEC 61131-3 standard does not permit pointers.
Importance: Low

\section*{Example}
```

VAR
piTemp : POINTER TO INT;
pbyTemp : POINTER TO BYTE;
END_VAR
--> SA0111: Data type POINTER not allowed

```

See also
- ② Chapter 6.6.3.4.1.6 "Attribute 'analysis:report-multiple-instance-calls"" on page 4231

\section*{SA0112: Reference variables}

Detects variables of type REFERENCE TO
Justification: The IEC 61131-3 standard does not permit references.
Importance: Low

\section*{Example}
```

VAR
ref int : REFERENCE TO INT;
ref_dw : REFERENCE TO DWORD;
END_\overline{VAR}
--> Data type REFERENCE not allowed

```

\section*{See also}
- «2 Chapter 6.6.3.4.1.6 "Attribute 'analysis:report-multiple-instance-calls"' on page 4231

\section*{SA0113: Variables with data type WSTRING}

Detects variables of type WSTRING
Justification: Not all systems support WSTRING. The code is more easily portable without WSTRING.

\section*{Example}
```

VAR
wstrTemp : WSTRING;
END_VAR
--> SA0113: Data type WSTRING not allowed

```

\section*{See also}
- «2 Chapter 6.6.3.4.1.6 "Attribute 'analysis:report-multiple-instance-calls"' on page 4231

\section*{SA0114: Variables with data type LTIME}

Detects variables of type LTIME.
Justification: Not all systems support LTIME The code is more easily portable without LTIME.
Importance: Low

\section*{Example}
```

VAR
ltVar : LTIME; // SA0114
END_VAR
--> SA0114: Data type LTIME not allowed

```

See also
- \({ }^{\Perp}\) Chapter 6.6.3.4.1.6 "Attribute 'analysis:report-multiple-instance-calls"" on page 4231

\section*{SA0115: Variables with data type UNION}

Detects declarations of a UNION data type and variable declarations of the UNION type Justification: The IEC 61131-3 standard does not include unions. The code is more easily portable without unions.
Importance: Low
Example
```

TYPE u1: UNION
lrTemp : LREAL;
liTemp : LINT;
END UNION
END_TYPE
PROGRAM PLC_PRG
VAR
uVar: ul;
END_VAR
--> SA0115: Unions not allowed

```

See also
- \({ }^{*}>\) Chapter 6.6.3.4.1.6 "Attribute 'analysis:report-multiple-instance-calls'" on page 4231

\section*{SA0117: Variables with data type BIT}

Detects variable declarations of data type BIT (possible within structure definitions) Justification: The IEC 61131-3 standard does not include the data type BIT. The code is more easily portable without BIT.

Importance: Low

\section*{Example}
```

TYPE Struct1 :
STRUCT
bitVar : BIT;
iVar : INT;
bVar : BOOL;
END_STRUCT
END_TYPE
--> SA0117: Variables with data type BIT

```

\section*{See also}
- \({ }^{2}\) Chapter 6.6.3.4.1.6 "Attribute 'analysis:report-multiple-instance-calls"' on page 4231

\section*{SA0119:Object-oriented features}

Detects the use of object-oriented features, such as function block declarations with EXTENDS and IMPLEMENTS, or property and interface declarations. This rule is useful when you write code that is intended to be ported to other IEC 61131-3-compliant systems.

Justification: Not all systems support object-oriented programming. The code is more easily portable without object-orientation.

Importance: Low

\section*{Example}
```

//Function block extended by another and implementing an interface:
FUNCTION_BLOCK POU EXTENDS CTD IMPLEMENTS ITF //SA0119
// Declaration parts of property methods assigned to a function
block:
POU.Prop.Get //SA0119
POU.Prop.Set //SA0119
--> SA0119: Object-oriented features not allowed

```

\section*{See also}
- ② Chapter 6.6.3.4.1.6 "Attribute 'analysis:report-multiple-instance-calls"" on page 4231

\section*{SA0120: Program calls}

Detects program calls
Justification: According to the IEC 61131-3 standard, programs can be called in the task configuration only. The code is more easily portable when you do not call programs from other locations.

Importance: Low
```

Example

```
```

PROGRAM prog_control

```
PROGRAM prog_control
VAR
VAR
END_VAR
END_VAR
PROGRAM PLC PRG
PROGRAM PLC PRG
VAR
VAR
END_VAR
END_VAR
prog_control();
prog_control();
--> SA0120: Program call to 'prg_control' not allowed
```

--> SA0120: Program call to 'prg_control' not allowed

```

\section*{See also}
- ② Chapter 6.6.3.4.1.6 "Attribute 'analysis:report-multiple-instance-calls"' on page 4231

\section*{SA0121: Missing VAR_EXTERNAL declarations}

Detects the use of a global variable in function blocks without them being declared there as "VAR_EXTERNAL"
Justification: According to the IEC 61131-3 standard, access to global variables is permitted only by an explicit import by means of a VAR_EXTERNAL declaration.

Importance: Low
PLCopen rule: CP18

\section*{Example}
```

VAR_GLOBAL
iGlob1:INT;
END_VAR
PROGRAM PLC_PRG
VAR
ivar:INT;
END_VAR
ivar:=iGlob1; // SA0121
--> SAO121: EXTERNAL declaration required for variable ''iGlob1'

```

\section*{Example: \\ Avoid error}
```

VAR_GLOBAL
iGlob1:INT;
END_VAR
PROGRAM PLC_PRG
VAR
ivar:INT;
END_VAR
VAR EXTERNAL
iGlob1:INT;
END_VAR
ivar:=iGlob1; // OK

```

See also
- \({ }^{\boldsymbol{y}} \boldsymbol{y}\) Chapter 6.6.3.4.1.6 "Attribute 'analysis:report-multiple-instance-calls"' on page 4231

\section*{SA0122: Array index defined as expression}

Detects the use of expressions in the declaration of array indexes Justification: Not all systems permit expressions as array limits.

Importance: Low

\section*{Example}
```

PROGRAM PLC_PRG
VAR CONSTANT
c_iValue : INT := INT\#15;
EN\overline{D VAR}
VAR
arr: ARRAY[0..c_iValue + 1] OF INT;
END_VAR
--> SA0122: Only constants allowed for array definition 'arr'

```

\section*{See also}
- ② Chapter 6.6.3.4.1.6 "Attribute 'analysis:report-multiple-instance-calls"" on page 4231

SA0123: Usages of INI, ADR or BITADR
Detects the use of the CODESYS-specific operators INI, ADR, and BITADR Justification: CODESYS-specific operators prevent the portability of code. Importance: Low

\section*{Example}
```

PROGRAM PLC_PRG
VAR
uiTemp: UINT;
TempVarInFUNC: DWORD;
END_VAR
TempVarInFUNC := ADR(uiTemp); //SA0123
--> SA0123: Operator 'ADR' not allowed

```

\section*{See also}
- \({ }^{4}\), Chapter 6.6.3.4.1.6 "Attribute 'analysis:report-multiple-instance-calls"" on page 4231

\section*{SA0147: Unusual shift operation - strict}

Detects bit shift operations that are not made to bitfield data types (BYTE, WORD, DWORD, LWORD)
Justification: The IEC 61131-3 standard permits bit access only to bitfield data types. However, the CODESYS compiler also permits bit shift operations with unsigned data types.
Importance: Low

\section*{See also the strict rule SA0018.}

\section*{Example}
```

PROGRAM PLC_PRG
VAR
in_byte : BYTE := 16\#45; // 2\#01000101
in_word : WORD := 16\#0045; // 2\#0000000001000101
in_uint : UINT;
in_dint : DINT;
erg_byte : BYTE;
erg_word : WORD;
erg_uint : UINT;
erg_dint : DINT;
n: \overline{BYTE := 2;}
END_VAR
erg_byte := SHL(in_byte,n); // no error because BYTE is a bit field
erg_word := SHL(in_word,n); // no error because WORD is a bit field
erg_uint := SHL(in_uint,n); // SA0147
erg_dint := SHL(in_dint,n); // SA0147
--> SA0147: Unusual shift operation - strict

```

\section*{See also}
- \({ }^{\star} \Rightarrow\) Chapter 6.6.3.4.1.6 "Attribute 'analysis:report-multiple-instance-calls"" on page 4231
- 乡 Chapter 6.6.3.4.2.17 "SA0018: Unusual bit access" on page 4243

SA0148: Unusual bit access - strict
Detects bit access that is not made to bitfield data types (BYTE, WORD, DWORD, and LWORD). The IEC 61131-3 standard permits only bit access to bitfield data types. However, the CODESYS compiler also permits bit access to unsigned data types.

Example
```

PROGRAM PLC_PRG
VAR
iTemp1 : INT;
diTemp3 : DINT;
uliTemp4 : ULINT;
siTemp5 : SINT;
usiTemp6 : USINT;
byTemp2 : BYTE;
END_VAR
iTemp1.3 := TRUE; // SA0148
diTemp3.4 := TRUE; // SA0148
uliTemp4.18 := FALSE; // SA0148
siTemp5.2 := FALSE; // SA0148
usiTemp6.3 := TRUE; // SA0148
byTemp2.5 := FALSE; // no error because BYTE is a bit field
--> SA0148: Unusual bit access - strict

```

See also
- ② Chapter 6.6.3.4.1.6 "Attribute 'analysis:report-multiple-instance-calls"' on page 4231

\section*{SA0118: Initialisations not using constants}

Detects initializations that do not assign constants
Justification: Initializations should be constant if possible and should not refer to other variables. In particular, you should avoid function calls during initialization because this can allow access to uninitialized data.

Importance: Medium

\section*{Example}
```

PROGRAM PLC_PRG
VAR
dwTemp : DWORD := 22;
dwTest : DWORD := dwTemp; // SA0118
dwVar : DWORD := TempVarInFUNC(); // SA0118
END VAR
--> SAO118: Initialisations not using constants

```

\section*{See also}
- \(\quad>\) Chapter 6.6.3.4.1.6 "Attribute 'analysis:report-multiple-instance-calls'" on page 4231

\section*{SA0124: Pointer dereferences in declarations}

Detects pointer dereferences that are used for initialization in the declaration part
Justification: Pointers and references should not be used for initializations because this can lead to access violations if the pointer has not been initialized.
Importance: Medium
Example
```

FUNCTION_BLOCK FB_Test
VAR_INPUT
refStruct: REFERENCE TO ST_Test;
END VAR
VAR
xPointer : BOOL := refStruct.a; // SA0124
iCount : INT;
END_VAR
--> SA0124: Dereference access in initialisation

```

\section*{See also}
- \({ }^{4}\) ) Chapter 6.6.3.4.1.6 "Attribute 'analysis:report-multiple-instance-calls"" on page 4231

\section*{SA0125: References in initializations}

Detects reference variables that are used for initialization in the declaration part
Justification: Pointers and references should not be used for initializations because this can lead to access violations if the pointer has not been initialized.

Importance: Medium

\section*{Example}
```

PROGRAM PLC_PRG
VAR
xRef: REFERENCE TO INT;
iCount: INT := xRef;
END_VAR
--> SAO125: Reference used in initializations

```

\section*{See also}
- \({ }^{\Perp}\) Chapter 6.6.3.4.1.6 "Attribute 'analysis:report-multiple-instance-calls"' on page 4231

\section*{SA0140: Statements commented out}

Detects commented-out statements
Justification: Code is often commented out for debugging purposes. When this kind of comment is released, it is not always clear at a later time whether the code should be deleted, or whether it has been commented out for debugging purposes and unintentionally not uncommented.

Importance: High
PLCopen rule: C4
Example
```

PROGRAM PLC_PRG
VAR
iValuel: INT;
iValue2: INT;
END_VAR
iValue1 := 100;
iValue2 := 200;
// iValue2 := 300;
--> SA0140: Statement commented out:: iValue2 := 300

```

\section*{See also}
- \({ }^{\star}\), Chapter 6.6.3.4.1.6 "Attribute 'analysis:report-multiple-instance-calls"' on page 4231

\section*{Possible Use of Uninitialized Variables}
6.6.3.4.2.51.1 SA0039: Possible null-pointer deferences.............................. 4296
6.6.3.4.2.51.2 SA0046: Possible use of not initialised interface...................... 4297
6.6.3.4.2.51.3 SA0145: Possible use of not initialised reference.................... 4298

SA0039: Possible null-pointer deferences
Detects code locations where a null pointer is possibly dereferenced
Justification: A pointer should be checked before each dereferencing to make sure it is not equal to zero. Otherwise an access violation may occur at runtime.

Importance: High

\section*{Example}
```

PROGRAM PLC_PRG
VAR
ptiVar1:POINTER TO INT;
ptiVar2:POINTER TO INT;
ptiVar3:POINTER TO INT;
iVar:INT;
iCount :INT;
iCondition: INT;
END_VAR
iCount := iCount + INT\#1;
ptiVar1 := ADR(iVar);
ptiVar1^ := iCondition; // OK - valid reference
ptiVar2^ := iCondition; // SA0039 - null pointer dereferenciation
iVar := ptiVar3^; // SA0039 - null pointer dereferenciation
--> SA0039: Possible null pointer dereference 'ptiVar2^'
--> SA0039: Possible null pointer dereference 'ptiVar3^'

```

\section*{See also}
- \& Chapter 6.6.3.4.1.6 "Attribute 'analysis:report-multiple-instance-calls"' on page 4231

SA0046: Possible use of not initialised interface
Detects the use of interfaces that were not initialized before being used
Justification: An interface reference should be checked for <> 0 before it is used. Otherwise an access violation may occur during access.
Importance: High
```

Example
//declaration of INTERFACE ITF and assigned METH2:
METHOD METH2 : BOOL
VAR_INPUT
iInput2:INT;
END_VAR
//declaration of INTERFACE Master_ITF1 and assigned METH:
METHOD METH : BOOL
VAR INPUT
iInput:INT;
END_VAR
PROGRAM PLC_PRG
VAR
instPOU:POU;
instITF:ITF;
instMasterITF1:Master_ITF1;
instMasterITF2:Master__ITF2;
iDummy:INT;
xDummy:BOOL;
instNoInitITF:ITF;
instNoInitITF2:ITF;
instNoInitMasterITF1:Master_ITF1;
instNoInitMasterITF2:Master_ITF2;
END_VAR
instITF := instPOU;
xDummy := instITF.METH(iInput := iDummy); // OK
instMasterITF1 := instPOU;
xDummy := instMasterITF1.METH(iInput := iDummy); // OK
xDummy := instNoInitITF.METH(iInput := INT\#1); // SA0046
xDummy := instNoInitITF.METH2(iInput2 := INT\#2); // SA0046
xDummy := instNoInitMasterITF1.METH(iInput := INT\#3); // SA0046
iDummy := instNoInitMasterITF2.Prop; // SA0046
IF instNoInitITF <> O THEN
instNoInitITF.Prop; // OK, weil das Interface nicht 0
sein kann
END_IF
--> SAOO46: Possible use of not initialised interface
'instNoInitITF'
--> SA0046: Possible use of not initialised interface
'instNoInitITF'
--> SAOO46: Possible use of not initialised interface
'instNoInitMasterITF1'
--> SAOO46: Possible use of not initialised interface
'instNoInitMasterITF2

```

\section*{See also}
- \({ }^{4}\) Chapter 6.6.3.4.1.6 "Attribute 'analysis:report-multiple-instance-calls"' on page 4231

SA0145: Possible use of not initialised reference
Detects any reference variables that may not be initialized before use and are not checked by the operator ISVALIDREF. This rule is applied in the implementation part of POUs. Rule SA0124 applies to the declaration.

Justification: A reference should be checked for its validity before access because an access violation may occur during access.
Importance: High
Example
```

PROGRAM PLC_PRG
VAR_INPUT
ref_iTest : REFERENCE TO INT;
END_VAR
ref_iTest := 99; // SA0145
IF __ISVALIDREF(ref_iTest) THEN
ref_iTest := 88;
END_IF
--> SA0145: Possible use of not initialised reference 'ref_iTest'

```

\section*{See also}
- ② Chapter 6.6.3.4.1.6 "Attribute 'analysis:report-multiple-instance-calls"' on page 4231
- Hy Chapter 6.6.3.4.2.49.15 "SA0124: Pointer dereferences in declarations" on page 4295

\section*{SA0150: Violations of lower or upper limits or the metrics}

Detects the POUs that violate the activated metrics at the lower or upper limits
Justification: Code that complies with certain metrics is easier to read, easier to maintain, and easier to test.
Importance: High
PLCopen rule: CP9
Example Initial situation: The "Number of calls" metric is selected in "Project Settings \(\rightarrow\) Static Analysis \(\rightarrow\) Metrics". Lower limit: 0 ; upper limit: 3. Prog_1 is called five times.
When running the static analysis, the "SA0150: Metric violation for Prog_1. Report for metric calls (5) > 2" error is issued in the message view, in the "Build" category.

See also
- \({ }^{2}\) Chapter 6.6.3.4.1.6 "Attribute 'analysis:report-multiple-instance-calls"' on page 4231

\section*{SA0160: Recursive calls}

Detects recursive calls in actions, methods, and properties of function blocks. Also detects possible recursions from virtual function calls and interface calls.

Justification: Recursions lead to non-deterministic behavior and are therefore a source of errors. Importance: Medium
PLCopen rule: CP13

\section*{Example The following method Call is assigned to the function block FB_Test:}
```

FUNCTION_BLOCK FB_Test
VAR
bParameter: BOOL;
END_VAR
METHOD Call : BOOL
VAR_INPUT
END-VAR
Cal\overline{l}}:= THIS^.Call(); //SA0160
The program PLC_PRG calls FB_Test:
PROGRAM PLC_PRG
VAR
fbTest : FB_Test;
bValue : BOO}L
END VAR
bValue := fbTest.bParameter;
fbTest.Call();
--> SA0160: Recursive call detected: 'PLC_PRG -> FB_Test.Call ->
FB_Test.Call

```

See also
- \({ }^{2}\) Chapter 6.6.3.4.1.6 "Attribute 'analysis:report-multiple-instance-calls"' on page 4231

\section*{SA0161: Unpacked structure in packed structure}

Detects unpacked structures that are used in packed structures
Justification: The compiler typically sets an unpacked structure to an address that allows aligned access to all elements within the structure. If you create this structure in a packed structure, then aligned access is no longer possible. Furthermore, access to an element in the unpacked structure can lead to a misalignment exception.
Importance: High
```

Example The structure structSingleDataRecord is packed, but it contains the unpacked structures
struct4Byte and struct9Byte.
{attribute 'pack_mode' := '1'}
TYPE structSingle\overline{DataRecord :}
STRUCT
str9ByteData: struct9Byte; (* 9 BYTE *)
str4ByteData: struct4Byte; (* 4 BYTE *)
udil: UDINT;
udi2: UDINT;
udi3: UDINT;
usi4: USINT;
END_STRUCT
END_TYPE (* 9 BYTE *)
TYP\overline{E struct9Byte :}
STRUCT
usiRotorSlots: USINT; (* 1 BYTE *)
uiMaxCurrent: UINT;
usiVelocity: USINT;
uiAcceleration: UINT; (* 2 BYTE *)
uiDeceleration: UINT; (* 2 BYTE *)
usiDirectionChange: USINT; (* 1 BYTE *)
END_STRUCT
END_TYPE TYPE struct4Byte :
STRŪCT
//udiDummy : UDINT;
rRealDummy : REAL;
END_STRUCT
END_TYPE
--> SA0161: Declaration of an unpacked struct 'struct9ByteData'
inside a packed struct 'structSingleDataRecord'
--> SA0161: Declaration of an unpacked struct 'struct4ByteData'
inside a packed struct 'structSingleDataRecord'

```

\section*{See also}
- \({ }^{\star}\) Chapter 6.6.3.4.1.6 "Attribute 'analysis:report-multiple-instance-calls"' on page 4231

\section*{SA0162: Missing comments}

Detects uncommented locations in the program
Justification: Complete commenting is required by many programming guidelines, and it increases the readability and maintainability of the code.
Importance: Low
PLCopen rule: C2
Comments are required in the following cases:
- Declaration of variables (Comments are located either above the declaration or to the right of the declaration.)
- Declaration of programs, function blocks, or methods (Comments are located above the declaration in the first line.)

\section*{Example}
```

PROGRAM PLC_PRG
VAR
iMaxValue: INT;
END_VAR
--> SA0162: Missing comment for 'PLC_PRG'
--> SA0162: Missing comment for 'iMaxValue'

```

\section*{See also}
- \({ }^{2}\) Chapter 6.6.3.4.1.6 "Attribute 'analysis:report-multiple-instance-calls"' on page 4231

\section*{SA0163: Nested comments}

Detects nested comments
Justification: Nested comments should be avoided because they are difficult to read.
Importance: Low
PLCopen rule: C3

\section*{Example}
```

{attribute 'do-analysis'}
(* That is
(* nested comment 1 *)
*)
PROGRAM PLC_PRG
VAR
(* That is
// nested comment 2
comment *)
iVal1: INT;
iVal2: INT;
(* That is
(* nested comment 3 *) *)
pVal3: POINTER TO DWORD;
hugo: INT;
END_VAR
(* That is
// nested comment 4
comment *)
iVal1 := iVal1 + 1;
(* That is
(* nested comment 5 *)
*)
(* Not that one *)
--> SA0163: Nested comment 'nested comment 1'
--> SA0163: Nested comment 'nested comment 2'
--> SA0163: Nested comment 'nested comment 3'
--> SA0163: Nested comment 'nested comment 4'
--> SA0163: Nested comment 'nested comment 5'

```

See also
- \({ }^{4}\) Chapter 6.6.3.4.1.6 "Attribute 'analysis:report-multiple-instance-calls"' on page 4231

\section*{SA0164: Multiline comments}

Detects multiline comments that are coded as (* comment *). Only single-line comments that are coded as / / comment are permitted.

Justification: Some programming guidelines prohibit multiline comments in code because the beginning and end of a comment could get lost and the closing comment bracket could be deleted by accident.

You can deactivate this check by means of the pragma analysis, also for comments in the declaration part.

Importance: Low
PLCopen rule: C5

\section*{Example}
```

{attribute 'do-analysis'}
(*
This is a multi-line comment // SA0164
*)
PROGRAM PLC PRG
VAR
// This is a single line comment
a: DINT;
END_VAR
(* This is not a single line comment *) // SA0164
a := a + 1;

```

\section*{See also}
- \(\Leftrightarrow\) Chapter 6.6.3.4.1.3 "Attribute 'analysis'" on page 4229

\section*{SA0165: Tasks calling other POUs than programs}

Detects tasks that call function blocks or functions instead of a program
Justification: This rule is part of the PLCopen Coding Guidelines. Therefore, compliance is also checked in CODESYS. We do not see any problems with data consistency in CODESYS if tasks would call POUs other than programs. However, problems can occur if the code is to be ported to other platforms.

Importance: Low
PLCopen rule: CP16
Tasks are inserted below the task configuration. The POUs to be called are configured in the tasks. The POUs must be the "Program" type. The "Function block" and "Function" types are not permitted.
\begin{tabular}{|c|c|}
\hline Example & \begin{tabular}{l}
自！PLC Logic
Application通）Library Manager －FB（FB） －PLC＿PRG（PRG） \\
－Task Configuration －MainTask㥜 PLC PRG
\end{tabular} \\
\hline
\end{tabular}

\section*{See also}
－② Chapter 6．6．3．4．1．6＂Attribute＇analysis：report－multiple－instance－calls＂＂on page 4231

SA0166：Max．number of input／output／in－out variables．．．
Detects whether or not a defined number of input variables（VAR＿INPUT），output variables （VAR＿OUTPUT）or VAR＿IN＿OUT variables is exceeded in a POU．In the＂Project Settings＂， double－click the rule entry \(\overline{\text { to }}\) open a dialog where you define the maximum number．
Justification：This is about checking individual programming guidelines．Many programming guidelines provide for a maximum number of POU parameters．Too many parameters make the code unreadable and the POUs difficult to test．

Importance：Medium
PLCopen rule：CP23
Example In the project settings，for Rule 166，you have defined a maximum number of 1 for VAR＿IN＿OUT variables．
```

FUNCTION BLOCK FB1
VAR_INPU\overline{T}
xIn : BOOL;
END VAR
VAR IN_OUT
xInOut1 : BOOL;
xInOut2 : BOOL;
END_VAR
--> SA0166: Too many VAR_IN_OUT variables in POU 'FB1'

```

\section*{See also}
－\({ }^{4}\) Chapter 6．6．3．4．1．6＂Attribute＇analysis：report－multiple－instance－calls＂＇on page 4231

\section*{SA0167：Temporary function block instances}

Detects function block instances that are declared as temporary variables．This affects instances that are declared in a method or function or as VAR＿TEMP，and therefore are reinitial－ ized in each processing cycle or for each POU call．
Justification：Function blocks have a state that is usually maintained over multiple PLC cycles． An instance on the stack exists only for the duration of the function call．Therefore，it rarely makes sense to create an instance as a temporary variable．Secondly，function block instances are often large and need a lot of space on the stack（which is usually restricted to controllers）． Thirdly，the initialization and often also the scheduling of a function block can take a long time．

Importance: Medium

\section*{Examples}
```

PROGRAM PLC_PRG
VAR
END_VAR
VAR_TEMP
yafb: AFB;
END_VAR
FUNCTION Fun : INT
VAR_INPUT
END_VAR
VAR
funafb: AFB;
END_VAR
METHOD METH : INT
VAR_INPUT
END_VAR
VAR
methafb: AFB; // SA0167
END_VAR
--> SA0167: Temporary function block instance: 'methafb'

```

\section*{See also}
- Chapter 6.6.3.4.1.6 "Attribute 'analysis:report-multiple-instance-calls"' on page 4231

\section*{SA0168: Unnecessary Assignments}

Detects assignments to variables which do not have any effect in the code.
Justification: When values are assigned to a variable multiple times without the variable being evaluated between assignments, the first assignments do not have any effect on the program. Importance: Low

\section*{Example}
```

PROGRAM PLC_PRG
VAR
dwVal1 : DWORD;
dwVal2 : DWORD;
END_VAR
dwVal1 := 1; // unnecessary assignment
IF dwVal2 > 100 THEN
dwVal2 := 0;
dwVal2 := dwVal2 + 1;
END_IF
dwVall := 2;
--> SA0168: The variable 'dwVall' is assigned but its value is
never used.

```

See also
- «Chapter 6.6.3.2 "Configuring and Running Static Analysis" on page 4209

\section*{SA0169: Ignored outputs}

Detects the outputs of methods and functions that are not specified when calling the method or function.

Justification: Ignored outputs can be a notice about an unhandled error or meaningless function calls because results are not used.
Importance: Medium

\section*{Example}
```

FUNCTION Fun1
VAR_INPUT
bIn : BOOL;
VAR END
VAR_OUTPUT
bOut : BOOL;
END_VAR
PROGRAM PLC_PRG
VAR
bValue :BOOl;
END_VAR
Fun1(bIn : TRUE);
--//SA0169: The output 'bOut' is ignored when called.

```

\section*{See also}
- \& Chapter 6.6.3.4.2.28.5 "SA0036: Unused output variables" on page 4251

\subsection*{6.6.4 Multi download tool}

The multi download tool for AC500 V3 PLCs in Automation Builder enables firmware updates and application downloads of the same project to several PLCs of the same type.

Preconditions - For application download: An AC500 V3 PLC with the desired target application is available as reference to create an online backup as input for the multi download tool (tbf backup file).
- For firmware download: The desired firmware must be provided as memory card image as input to the multi download tool.

It is recommended to close all Automation Builder instances and projects besides the multi download tool to avoid conflicts and performance reductions.

Create reference - For application backup: Create an online backup of your desired AC500 V3 PLC target data for download application via PLC device editor's "Backup and Restore" tab as tbf (target backup file). For further details, please refer to the help section.
\({ }^{\wedge}\) Chapter 6.4.1.21.2.8.6 "Tab 'Backup and Restore"' on page 2435
- For firmware download: Create a firmware memory card image of your target PLC application via context menu on the application node "Export \(\rightarrow\) Firmware (SD-card)...".

Usage of the The multi download tool can be launched within the Automation Builder via "Tools \(\rightarrow\) Options multi download tool


Selection The first editor section offers the possibility to select a firmware memory card image and/or an application backup file which shall be downloaded to the PLCs. Depending on the selection a firmware download and/or application download is then executed when executing the [Download] button.
The PLCs to download to can be defined in several ways:
- Execute a scan via [Scan] button and the found connected PLCs will be listed in the table.
- Import a corresponding PLC addresses configuration as csv-file into the table via [lmport list] button.
- Enter the IP addresses in the table manually or via copy and paste.

The PLCs to consider for download can be defined via the checkboxes in the first table column. A defined selection can be exported via the [Export list] button as csv-file for later reuse or external modification.

Device name, port, MAC address and serial number are shown as information only. The IP address itself is sufficient to identify the PLC.

Configuration General settings for the multi download tool can be configured by clicking the [Setting] button or settings via "Tools \(\rightarrow\) Options \(\rightarrow\) Multi download tool".
\begin{tabular}{|c|c|c|c|c|c|}
\hline Options & & & & & \(\times\) \\
\hline \begin{tabular}{l}
International Settings \\
Libraries \\
License Manager \\
Load and Save \\
Message View \\
Multi download tool \\
PLCopenXML \\
PROFINET \\
Proxy Settings \\
Python Scripting Editor \\
Refactoring \\
SFC editor \\
SmartCoding \\
Startup settings \\
Text editor \\
Visualization \\
Visualization styles \\
Visualization user management
\end{tabular} & \begin{tabular}{l}
Multi download tool \\
Network scan settings \\
Network scan timeout (seconds) \\
Blink timeout (seconds) \\
Multi download settings \\
Number of parallel downloads \\
Download timeout (minutes)
\end{tabular} & \[
\begin{aligned}
& 10 \\
& \hline 10 \\
& \hline \\
& \hline 10 \\
& \hline 5 \\
& \hline
\end{aligned}
\] & & & \\
\hline & & & QK & Cancel & \\
\hline
\end{tabular}
- "Network scan timeout": Maximum time to wait for the network scan results (by default 10 seconds).
- "Blink timeout": Maximum time to wait for the blink operation to execute (by default 10 seconds).
- "Number of parallel downloads": How many PLC downloads are executed in parallel (by default 10).
- "Download timeout": Depending on the network infrastructure the download might run into errors due to timeouts. The user will not be informed about these timeouts directly, but each timeout might result in errors. To avoid these timeouts the corresponding value can be adapted here in minutes (by default 5 minutes).

Specific settings like "Force download even when there are applications running on the PLC" and "Reboot PLC after download operation" can be defined directly in the lower part of the multi download tool editor.

Check Double-check the selection by executing the [Blink] button which triggers a display blinking of the checked PLCs which are connected to the PC. The connection status itself can be updated via the [Check connection] button indicating it in the "Connected" column.

Download The download of the defined firmware and/or application backup to the checked PLCs can be executed via the [Download] button.
If downloads to running PLCs shall be executed, the setting "Force download even when there are applications running on the PLC" must be enabled.


The PLC type of the application backup file/firmware memory card image is cross-checked with the PLC types of the checked PLCs before executing a download.

In case a firmware memory card image is selected for download process, first the firmware is download und updated. The status can be seen in the column "Firmware download". The selected application backup file is downloaded afterwards. The corresponding status is shown in the column "Application download".

If any error occurs, it is shown in the corresponding status column with additional information in the "Error message" column.
After successful download the PLCs are rebooted if the setting "Reboot PLC after download operation" is enabled.
The status of a download is shown as "Success", "Failed" or "In progress" for the firmware/application and the PLC state is shown with "Stop" and "Run" modes.

A running download can be cancelled via the [Cancel Download] button which finishes the currently running downloads and stops further executions afterwards.
An overall summary of executed and ongoing downloads can be found in the status line below the table. The results of the download can be documented via "Export Results".

\subsection*{6.6.5 Tool Calling Interface (TCI) implementation}

Automation Builder supports the Tool Calling Interface (TCI) up to conformance class 3 to engineer 3rd party safety devices.
The TCI from PROFINET organization offers a standard interface to call 3rd party configuration tools to e.g., configure iParameters in functional safety applications, calculate F_iPar_CRC values and write configurations via the PLC to the safety devices.

The corresponding vendor specific configuration tools must be installed manually on the same PC as the Automation Builder. As soon as a 3rd party safety device is added to the Automation Builder project and a matching configuration tool is found to be installed, corresponding context menus are offered on the device to call this tool via TCI (refer to screenshot below).


The offered functionality of those configuration tools strongly varies on the device manufacturer. In the following the functionalities and workflows are exemplarily shown with two tools.

TR TCI Device Tool
1. Install the "TR TCI Device Tool" on the same PC as the Automation Builder.
2. Install the required TR GSDML-files in the Automation Builder device repository.
3. Create the desired devices configuration in the Automation Builder device tree.
4. Define the required iParameter of the TR safety device in the Automation Builder.
5. Open the "TR TCI Device Tool" via context menu on the TR safety device.
6. Calculate the F_iPar_CRC in the "TR TCI Device Tool" and paste the value into the corresponding field in the Automation Builder editor.

In case an online connection is available you can also display the F-Destination-Address which can be set via rotary switches depending on the used hardware device and read/write maintenance data.
To enable an online connection, you must be online in Automation Builder, PLC needs to be in RUN and PROFINET must be running.


\section*{Siemens F-Des-tinationAddress Assign Device Tool}
1. Install the Siemens S7-Failsafe Configuration Tool on the same PC as the Automation Builder.
2. Install the required Siemens GSDML-files in the Automation Builder device repository.
3. Create the desired devices configuration in the Automation Builder device tree.
4. Change "F_Dest_Add" or "F_Source_Add" on the Siemens safety device in the Automation Builder project.
5. Set "F_Dest_Add" and "F_Source_Add" (if needed in offline mode).

6. Go online, PLC needs to be in RUN and PROFInet must be running.
7. Call the Siemens device tool and choose your preferred way of identification (LED blinking or serial number).

8. Select the check box at the module where the F-destination-address should be set.
9. Press the [Identification] button.
10. Press the [Assign PROFIsafe-address] button after you confirmed that either the correct module is blinking or the displayed serial numbers is matching.

The device tool has a build in timeout with only a view seconds. After that time the device tool cannot communicate with the slave anymore. In this case close the device tool and reopen it again.

\subsection*{6.6.6 Drive composer pro integration}

Drive Composer Pro is a start-up and maintenance tool for ABB's common architecture drives. The tool is used to view and set drive parameters, and to monitor and tune process performance.

Drive Composer Pro provides:
- Setting parameters,
- taking local control of the drive from the PC,
- event logger handling
- control diagrams,
- fast monitoring,
- working with multiple drives on the PC tool network,
- macro script editing for parameters and much more.
1. Add "Drive Composer Pro" object into the tree via add object dialog.
2. Open the "Drive Composer Pro" with double-click on the object.

In the following section important functions are described.

Import of 1. Import of FSO backup files (*.dcsafety) and Drive Parameters backup files (*.dcparamsbak) into Automation Builder project via the Drive Composer Pro object in the device tree.
2. View of integrated FSO backup files and Drive Parameters backup files in Automation Builder project - refer to figure below.

\(\Rightarrow\)
Drive Composer Pro can't be launched directly with integrated "FSO backup files" but they have to be loaded manually via context menu on the drive in Drive Composer Pro \(\rightarrow\) "Safety Settings".

Export of 1. Select the FSO and Drive Parameters backup files.
backup files
2. Export the selected file by clicking [Export].
\(\Rightarrow\) Select the desired storage path.


Remove of 1. Select the FSO and Drive Parameter backup files from Automation Builder project.
backup files
2. Remove the selected files by clicking [Remove].


View standard drive parameter backup files
1. Open the "Crane_follower12.dcparamsbak" with double-click.
2. The "Drive Composer Pro" starts automatically.

Standard Drive Parameter backup files (*.dcparamsbak) are automatically displayed under "File Drives".

3. Saved changes in the standard drive parameter backup file are automatically updated in the Automation Builder project.

\subsection*{6.6.7 Professional Version Control}

\subsection*{6.6.7.1 SVN integration in CODESYS}

Professional Version Control allows for the development of CODESYS projects under version control by Apache \({ }^{\text {TM }}\) Subversion®. Professional Version Control provides an SVN client integrated in CODESYS. The objects of your project are versioned in a central SVN repository.

As a rule, the SVN repository should be created in a server configuration and located on a server. For testing purposes, you can create a local SVN repository where you can access via file://.

Professional Version Control requires a valid license and can be installed using the Automation Builder Installer or the Automation Builder Installation Manager.

\subsection*{6.6.7.2 Getting Started}

The following steps are required in order to develop your CODESYS project with Professional Version Control with version control by Apache \({ }^{\text {TM }}\) Subversion®:
1. Install the Professional Version Control package in CODESYS.
2. Install an SVN server.
3. Create an SVN repository.
4. Open your CODESYS project in CODESYS.
5. Import the CODESYS project into the SVN project archive.
\(\Rightarrow\) The CODESYS project is saved in the SVN repository.
6. To edit and further develop the project with SVN version control, the project is edited in CODESYS and then committed to the SVN repository.

As of version 4.4.0.0, in the case of storage profile inconsistencies between the working copy and the project in the SVN repository, corresponding instructions will guide you further. Under certain circumstances, commit actions at the object level are then no longer possible.
See also
- \& Chapter 6.6.7.4 "Using an SVN Repository" on page 4316
- 帠 Chapter 6.6.7.5 "Using Working Copies" on page 4318

\subsection*{6.6.7.3 Version control}

What is version control?

Apache \({ }^{\text {TM }}\) Subversion® (SVN) is a tool for version and revision management of current and previous versions of files, such as source code, websites, and documentation. Apache \({ }^{\text {TM }}\) Subversion \(®\) is a registered trademark of the Apache Software Foundation.

Revision management (also known as version control, version management, and source code management) is the management of changes to documents, programs, and other information that is stored as computer files. Version control is employed frequently in software development when a team of employees works on the same files.
Tasks
- Tracking of changes in revisions: At any time, you can show who made which changes at which time.
- Restoring of old revisions of individual files: At any time, you can reverse accidental changes to files.
- Archiving of special revisions of a project: At any time, you can revert to older versions.
- Coordination of shared access of developers to data
- Development of a project at the same time in multiple branches

Script Engine Professional Version Control provides a scripting-interface for SVN.
SVN Add-on API

\subsection*{6.6.7.4 Using an SVN Repository}

An SVN repository usually saves information as a file system tree, a hierarchy of files, and directories. Any number of clients connects to the SVN repository and reads or writes changes to the files in revisions.

\section*{NOTICE!}

Consult with your IT specialists for more information, for example how to create an SVN repository. For production purposes, we recommend a strictly dedicated administrative SVN server.
We recommend that you create the suggested default directory structure in the SVN repository.
See also
- http://svnbook.red-bean.com/en/1.8/ svn.tour.importing.html\#svn.tour.importing.layout

Creating an SVN repository for testing purposes

\section*{NOTICE!}

Use the file: / / access method for testing purposes only.

You can reach SVN repositories that were created in format 1.8 or 1.9 via the file:// protocol.

For testing purposes, you can create a local SVN repository without installing your own server. The SVN repository is accessed via file:// and provides the same functionality as a server.

\section*{Creating a test repository with TortoiseSVN}
\(\square\) Requirement: The SVN client TortoiseSVN 1.9 is installed on the development system.
1. Create a new, empty folder on your local file system. The test repository will be created there.
\(\Rightarrow\) Example: D:\SVN repository
2. Click "TortoiseSVN \(\rightarrow\) Create repository here".
\(\Rightarrow\) The dialog "Create repository" opens.
3. Click "Create directory tree".
\(\Rightarrow\) The SVN repository is created.

See also
- Documentation TortoiseSVN Documentation TortoiseSVN

Accessing the SVN repository

Table 772: SVN repository URLs
\begin{tabular}{|l|l|}
\hline file:/// & Direct access to an SVN repository (on local hard drive) \\
\hline http:// & \begin{tabular}{l} 
Access via WebDAV protocol to Apache server that is sup- \\
ported by SVN
\end{tabular} \\
\hline https:// & As http://, but with SSL encryption \\
\hline svn:// & Access via own protocol to an svnserve server \\
\hline svn+ssh:// & As svn://, but tunneled via SSH \\
\hline
\end{tabular}
1. Open the CODESYS project that you want to save in the SVN repository.
\(\Rightarrow\) Example: A.project is open.
2. Click "Project \(\rightarrow\) SVN \(\rightarrow\) Import project to SVN".
\(\Rightarrow\) The "Browse SVN repository" dialog opens.
3. Select the directory file:///D:/SVN repository/trunk in the directory tree.
4. Select the \(\pm\) command.
\(\Rightarrow\) The "Create remote directory" dialog opens.
5. Specify the URL for the new directory.

Note: Because the new directory should contain the CODESYS project, specify the project name with extension here.
\(\Rightarrow\) file:///D:/SVN\%20repository/trunk/A.project
6. Click "OK" to close the dialog.
7. Select the new project and click "OK" to exit the "Browse SVN repository" dialog.
\(\Rightarrow\) The "Import Project to SVN" dialog opens. The directory file:///D:/SVN repository/trunk/A.project is specified in "URL of SVN repository".

See also
- \({ }^{*} \boldsymbol{y}\) Chapter 6.6.7.6.1 "Overlay Icons" on page 4319

\subsection*{6.6.7.5 Using Working Copies}

Checking out a You can copy CODESYS projects to your development system that are saved in the SVN project
Creating a working copy
1. Open CODESYS
2. Click "Project \(\rightarrow\) SVN \(\rightarrow\) Checkout".
\(\Rightarrow\) The "Checkout" dialog opens.
3. Specify the URL of the SVN repository and select a project in the SVN repository tree.

A CODESYS project is automatically identified by its file extension (.project, _project, . library or _library) as a "project" or "library project" at checkout.
4. In "Checkout to", specify the name and location of the working copy on your development system.
5. Click "OK" to close the dialog.
\(\Rightarrow\) The project opens in CODESYS. In the object tree of the project, the SVN link is shown with overlaid icons. Now the project is saved as a working copy on your development system.

See also
- „y Chapter 6.6.7.6.1 "Overlay Icons" on page 4319

\section*{Editing the working copy}

Changed working copy format in Professional Version Control V4.1.0.0 and later

Update the working copy before you start editing, especially if the project is revised by a team. This is how you avoid conflicts.
1. Open the working copy.
2. Click "Project \(\rightarrow\) SVN \(\rightarrow\) Update project" (symbol: \(\longleftarrow)\).
\(\Rightarrow\) You working copy is current.
3. Revise your project.
4. Click "Project \(\rightarrow\) SVN \(\rightarrow\) Edit SVN working copy".
\(\Rightarrow\) The dialog opens. There you can browse your changes.
5. Close the dialog.
6. If necessary, you can click "SVN \(\rightarrow\) Revert" in the context menu.
\(\Rightarrow\) The file is reverted back to the base revision and your changes are discarded.
7. If necessary, you can click "Compare" in the context menu of an edited object.
\(\Rightarrow\) The compare dialog opens. You can resolve any conflicts here.
8. Close the compare dialog.
9. Click "Project \(\rightarrow\) SVN \(\rightarrow\) Commit project" (symbol: »).
\(\Rightarrow\) The "Commit" dialog opens.
10. In "Message", specify a log entry that describes your changes. Example: Changes for customer ABC, request 1234.
\(\Rightarrow\) Your changes are saved in the SVN repository as a revision with a revision number.

See also
- \({ }^{4}\), Chapter 6.6.7.6.2.2 "Command 'SVN Repository Browser'" on page 4322

For projects in version Professional Version Control V4.1.0.0 and later, the working directory (working copy) has a new format.
If you open a project that was created with V4.0.4.0 or earlier, then the project is updated automatically to the new format when it is opened.
If you open a project that was created with V4.0.4.0 or earlier and the project is based on an older SVN version of \(1.7 . x\) or earlier, then you are prompted whether or not CODESYS should update the format. If you decline the update, then the SVN link of the project is deactivated. You can still load and edit the project.
The update does not have an effect on saving to the SVN server. You can also checkout projects with earlier versions of the client. The new format affects only the local working directory.
See also
- http://svnbook.red-bean.com/en/1.8/svn.ref.svn.c.upgrade.html

\subsection*{6.6.7.6 Reference, User Interface}

\subsection*{6.6.7.6.1 Overlay Icons}

Every object in CODESYS has a status value in the SVN repository. This status value is displayed in the object tree (in the "POUs", "Devices", or "Modules" views) for each object by overlay icons.

Table 773: Overlay icons
\begin{tabular}{|c|c|}
\hline + & Object is planned to be added to the SVN repository. \\
\hline 1 & Object conflicted \\
\hline \(\times\) & Object deleted \\
\hline \(\square\) & Object modified \\
\hline \(\square\) & Object with modification in the metadata \\
\hline \(\square\) & Object with modifications in the memory format \\
\hline * & Object normal \\
\hline - & Object write-protected (read-only) \\
\hline - & Object locked \\
\hline * & Object with deleted subobjects \\
\hline ¢ & Object ignored on commit \\
\hline \(\pi\) & External object \\
\hline ¢ & Ignored object \\
\hline \(?\) & Unversioned object \\
\hline \({ }^{\text {® }}\) & Object with modified subobjects \\
\hline T & The object is not saved in the SVN repository. It will be created again when loaded from SVN. \\
\hline - & SVN_VERSION_INFO temporarily unavailable, for example as with interface libraries \\
\hline 中 & The status of the object is not updated. \\
\hline U & The object was modified on the server (Update available). \\
\hline \% & The object was locked on the server by another user (or in another working directory). \\
\hline 4 & Tree conflict by changes to the structure of the project \\
\hline
\end{tabular}

\subsection*{6.6.7.6.2 Commands}

\section*{Availability of commands}

Not all commands are available in the logged in state because some SVN commands of the project could be changed.

Table 774: Availability of commands
\begin{tabular}{|c|c|c|}
\hline Command & Not Logged In & Logged In \\
\hline \# Chapter 6.6.7.6.2.2 "Command 'SVN Repository Browser"' on page 4322 & X & X \\
\hline \& Chapter 6.6.7.6.2.3 "Command 'Edit SVN working copy'"' on page 4323 & X & \\
\hline \# Chapter 6.6.7.6.2.4 "Command 'Import project to SVN"" on page 4326 & X & X \\
\hline « Chapter 6.6.7.6.2.5 "Command 'Checkout"' on page 4327 & X & X \\
\hline \# Chapter 6.6.7.6.2.6 "Command 'Commit', Command 'Commit Project'" on page 4328 & X & X \\
\hline * Chapter 6.6.7.6.2.7 "Command 'Compare"' on page 4331 & X & \\
\hline \# Chapter 6.6.7.6.2.8 "Command 'Compare with HEAD revision'" on page 4331 & X & \\
\hline « Chapter 6.6.7.6.2.9 "Command 'Compare with revision"' on page 4331 & X & \\
\hline «2 Chapter 6.6.7.6.2.10 "Command 'Compare to remote project..."' on page 4332 & X & \\
\hline \# Chapter 6.6.7.6.2.11 "Command 'Include externals to project', Command 'Inc/ude externals'" on page 4333 & X & \\
\hline ๕ Chapter 6.6.7.6.2.12 "Command 'Ignore on commit'" on page 4335 & X & X \\
\hline \& Chapter 6.6.7.6.2.13 "Command 'SVN Info"' & X & X \\
\hline \& Chapter 6.6.7.6.2.14 "Command 'Show properties"' on page 4336 & X & X \\
\hline « Chapter 6.6.7.6.2.15 "Command 'Get lock"' on page 4336 & X & X \\
\hline \# Chapter 6.6.7.6.2.16 "Command 'Steal locks"' on page 4337 & X & X \\
\hline \# Chapter 6.6.7.6.2.17 "Command 'Release lock"' on page 4337 & X & X \\
\hline «̌ Chapter 6.6.7.6.2.18 "Command 'Release locks recursively'" on page 4337 & X & X \\
\hline ² Chapter 6.6.7.6.2.19 "Command 'Show log', Command 'Show project log'" on page 4337 & X & X \\
\hline \# Chapter 6.6.7.6.2.20 "Command 'Revert', Command 'Revert project"' on page 4339 & X & \\
\hline \& Chapter 6.6.7.6.2.21 "Command 'Revert to revision', Command 'Revert project to revision'" on page 4340 & X & \\
\hline \# Chapter 6.6.7.6.2.22 "Command 'Update', Command 'Update project'" on page 4340 & X & \\
\hline \({ }^{4}\) ) Chapter 6.6.7.6.2.23 "Command 'Update to revision"' on page 4341 & X & \\
\hline « Chapter 6.6.7.6.2.24 "Command 'Update only this"' on page 4342 & X & \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|}
\hline Command & Not Logged In & Logged In \\
\hline ²) Chapter 6.6.7.6.2.25 "Command 'Disconnect project from SVN'" on page 4342 & X & X \\
\hline \# Chapter 6.6.7.6.2.26 "Command 'Switch'" on page 4342 & X & \\
\hline \({ }^{〔}\) Chapter 6.6.7.6.2.27 "Command 'Un-Ignore on commit"' on page 4343 & X & X \\
\hline « Chapter 6.6.7.6.2.28 "Command 'SVN Cleanup'" on page 4343 & X & X \\
\hline (4) Chapter 6.6.7.6.2.29 "Command 'Clear authentication data' " on page 4344 & X & X \\
\hline « \({ }^{4}\) Chapter 6.6.7.6.2.30 "Command 'Merge changes"' on page 4344 & X & \\
\hline ② Chapter 6.6.7.6.2.31 "Command 'Connect to existing project'" on page 4345 & X & X \\
\hline \# Chapter 6.6.7.6.2.32 "Command 'Resolve conflict' " on page 4346 & X & \\
\hline *) Chapter 6.6.7.6.2.33 "Command 'Work in offline mode"' on page 4346 & X & X \\
\hline \begin{tabular}{l}
7) Chapter 6.6.7.6.2.34 "Command 'Copy \\
(Branch/Tag)'" on page 4347
\end{tabular} & X & \\
\hline ② Chapter 6.6.7.6.2.35 "Command 'Pending Changes"' on page 4348 & X & X \\
\hline
\end{tabular}

\section*{Command 'SVN Repository Browser'}

Symbol:
Function: This command opens the SVN repository browser. The contents of an SVN repository is shown in a tree structure here. You can search through the repository in the browser.
Call: Menu bar: "Project \(\rightarrow\) SVN".
Depending on the selected object, the following commands are available in the context menu:
- "Show log"
- "Checkout"
- "Create folder"
- "Copy to"
- "Rename"
- "Delete"

Double-clicking the object with the right mouse button opens the log dialog.

\section*{Dialog 'SVN \\ Repository \\ Browser'}
\(\left.\begin{array}{|l|l|l|}\hline \text { "URL" } & \begin{array}{l}\text { URL in SVN repository } \\ \text { Example: https: / / svnserver/repository/trunk/ } \\ \text { ControlABC.project }\end{array} \\ \text { Tip: As soon as a valid SVN repository is specified, you can browse and select a } \\ \text { specific project by means of the adjacent button. }\end{array}\right]\)

See also
- © Chapter 6.6.7.6.3.3 "Dialog 'Select revision'" on page 4351

\section*{Command 'Edit SVN working copy'}

\section*{Symbol: 를}

Function: This command opens the dialog "Edit SVN working copy" and displays the working copy in a browser from the SVN view.

Call: Menu bar: "Project \(\boldsymbol{\rightarrow}\) SVN".
The functionality of the browser allows for:
- Access to and actions on objects that are not displayed in the "Devices" view.
- Actions on objects that can lead to exceptions in the "Devices" view.
- Editing of global objects that are modified, in conflict, or blocked.

\section*{Dialog 'Edit SVN}
working copy'
Table 775: "Edit SVN working copy: <project name> - <project URL>"
\begin{tabular}{|c|c|}
\hline "Path in the SVN repository" & \begin{tabular}{l}
Display of working copy from SVN view repository. The file and folder structure of the objects in the project are presented in a tree view here. As a result, the recursion depth of an object is visible. \\
Object selected for the following menu command
\end{tabular} \\
\hline "Name of object" & \begin{tabular}{l}
File name of the object \\
Example: Application
\end{tabular} \\
\hline "Node type" & The top node is the project root directory. \\
\hline "Text status" & \begin{tabular}{l}
Object status: \\
- "modified" \\
- "added" \\
- "deleted" \\
- "non-versioned" \\
- "Conflicted"
\end{tabular} \\
\hline "Property status" & \begin{tabular}{l}
Status in SVN repository: \\
- "modified" \\
- "added" \\
- "deleted" \\
- "Conflicted" \\
- "normal"
\end{tabular} \\
\hline "Revision" & Revision number \\
\hline "Conflict information" & File conflict, property conflict, or tree conflict \\
\hline "Lock" & For locked objects, the user who applied the lock is displayed. Example: b.mayer \\
\hline "Lock comment" & Lock message. Implicit, normal, or stolen lock. \\
\hline "URL" & URL of the object \\
\hline
\end{tabular}

Table 776: Menu commands
\begin{tabular}{|l|l|}
\hline\(\boxtimes\) & \\
\hline "Select \(\rightarrow\) All" & Selects all files. \\
\hline "Select \(\rightarrow\) None" & Deselects all files. \\
\hline "Select \(\rightarrow\) Modified" & Selects the modified files. \\
\hline "Select \(\rightarrow\) Conflicted" & Selects the conflicted files. \\
\hline "Select \(\rightarrow\) Locked" & Selects the locked files. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\(\leqslant\) & \begin{tabular}{l} 
Updates the working copy. Changes made by others are added from the SVN \\
repository to your working copy.
\end{tabular} \\
\hline "Update \(\rightarrow\) Project" & Updates all files of the project. \\
\hline "Update \(\rightarrow\) Selected nodes" & Updates only the selected files. \\
\hline \begin{tabular}{l} 
"Update \\
\(\overrightarrow{~ S ~ S e l e c t e d ~ n o d e s ~ a n d ~}\) \\
children"
\end{tabular} & Updates the selected files and subordinate files. \\
\hline
\end{tabular}

↔＂Clear＂ \(\left\lvert\, \begin{aligned} & \text { Discards your changes to the working copy．Then the object corresponds to the } \\ & \text { revision in the repository．}\end{aligned}\right.\) revision in the repository．
\begin{tabular}{|l|l|}
\(\times\) & \\
\hline ＂Delete \(\rightarrow\) Selected nodes＂ & Deletes the selected objects from the working copy．
\end{tabular}
\begin{tabular}{|l|l|}
\(\star\)＂Commit＂ & \begin{tabular}{l} 
Commits your changes to the SVN repository．Any locked objects will be \\
unlocked．
\end{tabular} \\
\hline ＂Commit \(\rightarrow\) Project＂ & Commits all files in the project． \\
\hline ＂Commit \(\rightarrow\) Selected nodes＂ & Commits only the selected files． \\
\hline \begin{tabular}{l}
＂Commit \\
\(\rightarrow\) Selected nodes and \\
children＂
\end{tabular} & Commits the selected files and subordinate files． \\
\hline
\end{tabular}
\begin{tabular}{|l|l|} 
白 & Commands for managing locks． \\
\hline ＂Locks \(\rightarrow\) Revalidate all＂ & \begin{tabular}{l} 
Checks the validity of locks in the working copy．Any invalid locks will be \\
unlocked．
\end{tabular} \\
\hline ＂Locks \(\rightarrow\) Release locks＂ & Releases the lock． \\
\hline ＂Locks \(\rightarrow\) Acquire locks＂ & Locks the object from editing by others． \\
\hline ＂Locks \(\rightarrow\) Steal locks＂ & \begin{tabular}{l} 
Locks the file for you and removes the lock of another user． \\
Tip：Avoid stealing a lock because the changes made by another user can be \\
lost．
\end{tabular}
\end{tabular}
\begin{tabular}{|l|l|}
\(\boldsymbol{\Delta}\) & Commands to resolve conflicts． \\
\hline ＂Conflicts \(\rightarrow\) Mark as resolved＂＂ & \begin{tabular}{l} 
Indicates a displayed conflict in the SVN repository as marked and resolved． \\
Note：Select the command if you edited and resolved the displayed conflict． \\
Then you can commit changes again．
\end{tabular} \\
\hline \begin{tabular}{l}
＂Conflicts \\
\(\rightarrow\) Resolve using theirs＂
\end{tabular} & \begin{tabular}{l} 
Resolves the conflict：In the SVN repository，the changes are accepted that were \\
committed by other users．Your changes are discarded．
\end{tabular} \\
\hline \begin{tabular}{l}
＂Conflicts \\
\(\rightarrow\) Resolve using mine＂
\end{tabular} & \begin{tabular}{l} 
Resolves the conflict：In the SVN repository，the changes to your working copy \\
are accepted and the changes by other users are discarded．
\end{tabular}
\end{tabular}

\footnotetext{
\(\square^{2}\)＂Show log＂
}

〕＂＂Change location＂

中＂Update＂
Opens the dialog＂Log－Application＂．The history of the selected node is shown here．The previous revisions are displayed with the respective actions．

Changes the storage location of the selected object within the working copy．
Example：You can resolve a tree conflict by saving the local object to another location．Then update the parent object to apply it to the locked children．
\(\square\)
｜Updates the browser view by rescanning the working copy．
Executes an SVN cleanup operation on the working copy．

See also
－\(\Leftrightarrow\) Chapter 6．6．7．6．2．22＂Command＇Update＇，Command＇Update project＇＂on page 4340
－乡y Chapter 6．6．7．6．2．20＂Command＇Revert＇，Command＇Revert project＂＇on page 4339
－\(\&\) Chapter 6．6．7．6．2．6＂Command＇Commit＇，Command＇Commit Project＇＂on page 4328
－\(\quad\) ，Chapter 6．6．7．6．2．32＂Command＇Resolve conflict＇＂on page 4346
－\(\quad \stackrel{y}{c}\) Chapter 6．6．7．6．2．15＂Command＇Get lock＇＂on page 4336
－\(\quad\) \＆Chapter 6．6．7．6．2．17＂Command＇Release lock＇＂on page 4337
- \# Chapter 6.6.7.6.2.16 "Command 'Steal locks"' on page 4337
- \(\#\) Chapter 6.6.7.6.2.26 "Command 'Switch"' on page 4342
- \& Chapter 6.6.7.6.2.19 "Command 'Show log', Command 'Show project log"' on page 4337
- \& Chapter 6.6.7.6.2.28 "Command 'SVN Cleanup"' on page 4343

\section*{Command 'Import project to SVN'}

Symbol:
Function: This command opens the "Import Project to SVN" dialog for importing a CODESYS project to the SVN repository.
Call: Menu bar: "Project \(\boldsymbol{\rightarrow}\) SVN".

\section*{Requirement}
- You have access to an SVN repository and you know its URL.
- You have read access to the entire project.

\section*{NOTICE!}
```

Projects are always saved unencrypted on the server. Therefore, take appropriate security measures (for example, respective access rights to the SVN server) for protecting your projects.

```

See also
- User and access management in Protect and save project

\section*{Dialog 'Import}

Project to SVN'
\begin{tabular}{|c|c|}
\hline "URL of SVN repository" & \begin{tabular}{l}
URL of the SVN repository with the new project folder where the files are imported \\
Example: https://svnserver/repository/trunk/ \\
ControlABC.project \\
Hint: When importing libraries, specify the extension .library or _library. For projects, specify the extension . project or project. Then the project type is recognized automatically at checkout and the options are set accordingly in the "Checkout" dialog.
\end{tabular} \\
\hline \(\cdots\) & Opens the "SVN Repository Browser" dialog The previous project structure is displayed and you can edit them here. \\
\hline "Import message" & \begin{tabular}{l}
Text for use as log message \\
Example: Control project for customer A
\end{tabular} \\
\hline "Recent messages" & Opens the "Recent Messages" dialog. There you can reuse the last log messages. \\
\hline "Generate SVN_VERSION_INFO" & \(\square\) : The object SVN_VERSION_INFO is not created automatically during the import operation. Therefore, the project does not get any global constants or variables for the project metadata. \\
\hline "OK" & Creates the current project in the SVN repository and imports the project objects. The local project in CODESYS Development System is linked to the SVN repository. Overlay icons show this in the object trees. \\
\hline
\end{tabular}

See also
- \& Chapter 6.6.7.6.2.5 "Command 'Checkout"' on page 4327
- \& Chapter 6.6.7.6.2.2 "Command 'SVN Repository Browser'" on page 4322
- * Chapter 6.6.7.6.1 "Overlay Icons" on page 4319

\section*{Command 'Checkout'}

Symbol: «৷
Function: This command opens the "Checkout" dialog. Here you can checkout a project stored in the SVN repository as a working copy.
Call: Menu bar: "Project \(\boldsymbol{\rightarrow}\) SVN".

\section*{Dialog 'Check- \\ out'}

Table 777: "URL of SVN repository"
\begin{tabular}{|c|c|}
\hline  & \begin{tabular}{l}
URL of the project in the SVN repository \\
Example: https://svnserver/repository/trunk/ \\
ControlABC.project \\
Tip: As soon as a valid SVN repository is specified, you can click the adjacent button or use the options to browse in "Revision" and select a specific project.
\end{tabular} \\
\hline 15 & \begin{tabular}{l}
Opens the dialog "Select revision". \\
The button is labeled with the currently selected revision: \\
- "HEAD": Newest revision (top revision). Default \\
- " 15 ": Revision number of the selected revision \\
- "12/23/2016 11:59:59 (UTC)": Change date of the selected revision (UTC) \\
Note: The dialog provides the same options as the "Revision" group.
\end{tabular} \\
\hline \(\ldots\) & Opens the "SVN repository browser" dialog Here you can browse the SVN repository. \\
\hline
\end{tabular}

Table 778: "Checkout to"
\begin{tabular}{|l|l|}
\hline "Name" & \begin{tabular}{l} 
Name of the working copy \\
Example: ControlABC.project
\end{tabular} \\
\hline "Location" & \begin{tabular}{l} 
Storage location of the working copy \\
Example: /D:/svn/repository/trunk/ControlABC.project
\end{tabular} \\
\hline
\end{tabular}

Table 779: "Checkout as"
\begin{tabular}{|l|l|}
\hline "Project" & The project is saved as a CODESYS project "<project name>.project". \\
\hline "Library" & The project is saved as a CODESYS library file "<project name>. library". \\
\hline "Auto-detect" & \begin{tabular}{l} 
CODESYS attempts to recognize the project type by means of the extension. \\
The current implementation checks whether the URL of the project ends with \\
" library" or ". library". In this case, the project is recognized as a library \\
or a project.
\end{tabular} \\
\hline
\end{tabular}

Table 780: "Checkout options"
\begin{tabular}{|l|l|}
\hline "Omit externals": & \(\boxed{ } \quad\) : Externals (external objects) are not copied to the working directory. \\
\hline
\end{tabular}

Table 781: "Revision"
For a description, refer to the section "Dialog 'Select revision'".
Note: The group provides the same options as the "Revision" dialog.

\section*{"OK" \\ Checks out the project from the SVN repository, saves it locally to the specified} location, and opens it in CODESYS as the primary project.


If files were encrypted when imported to the SVN repository, or if they have been committed, then note the following:

When committing to the SVN repository, the information about an encrypted project file is included. However, the type of encryption is not included (password, Wibu security key, X509 certificate). Therefore, it may be necessary to encrypt the working copy again in the project settings. In this case, a dialog opens when exiting the command to notify you of this. Then you are able to switch directly to the project settings.

\section*{See also}
- \({ }^{4}\) Chapter 6.6.7.6.3.3 "Dialog 'Select revision'" on page 4351
- \(y^{4}\) Chapter 6.6.7.6.2.2 "Command 'SVN Repository Browser'" on page 4322
- "Version control with Subversion", Section "Revision identifier"

\section*{Command 'Commit', Command 'Commit Project'}

Symbol: ォ
Function: The command commits changes that were made in CODESYS to the SVN repository. The "Commit" dialog opens for this purpose.

\section*{Call:}
- Context menu: "SVN" to commit exactly this object
- "Project \(\rightarrow\) SVN \(\rightarrow\) Commit Project" to commit all changes in the project at the same time

Requirement: At least one object was modified. An object whose contents have been modified is overlaid in the object tree with the \(\square, \square\), or \(\square\) Symbol.
When you execute the command, the lock on the objects to be committed is lifted automatically.
See also
- ⓨ Chapter 6.6.7.6.1 "Overlay Icons" on page 4319

\section*{Dialog 'Commit'}

Table 782: "Commit to: <URL project/object>"
\begin{tabular}{|l|l|}
\hline & \begin{tabular}{l} 
URL in SVN repository \\
Example: file:///D:/SVN repository/trunk/ControlABC.project
\end{tabular} \\
\hline "Log message" & \begin{tabular}{l} 
Type in a log message that comments your change. \\
Example: Bug fix error 123
\end{tabular} \\
\hline "Recent Messages" & \begin{tabular}{l} 
Opens the "Recent Messages" dialog for displaying the last log messages. You \\
can click a log message to accept it.
\end{tabular} \\
\hline
\end{tabular}

Table 783: "Changes made (double-click on object for compare, right-click on object for more operations)"

"Select/Deselect All"
All objects in the list are selected.
\begin{tabular}{|l|l|}
\hline "Keep Locks" & \(\boxed{\text { " }}\) : Your locked object remains in locked after the commit. \\
\hline "Keep Change Lists": & \(\boxed{\square}\) : The change list also remains after the commit. \\
& \(\square:\) The change list is not deleted after the commit.
\end{tabular}
\begin{tabular}{|c|c|}
\hline Button "Update Project" & \begin{tabular}{l}
Updates the project \\
Hint: Prevent conflicts by committing a previously updated project/object.
\end{tabular} \\
\hline \begin{tabular}{l}
"OK" \\
Keyboard shortcut [Ctrl]+ [Enter] \\
Keyboard shortcut [Ctrl]+ [Enter]
\end{tabular} & \begin{tabular}{l}
Checks the working copy first. Starts the commit of changes when the working copy is current. \\
Opens a dialog when the working copy is outdated. You can then select from the following: \\
- "Abort the commit, I want to investigate the issue." \\
- "Yes, I want to update this project now." \\
- "Continue with the commit, I know what I do." \\
Note: The history of the commit is displayed in the "Messages" view. \\
The messages are highlighted in color. \\
- Blue: Commit a change \\
- Green: Add an object \\
- Dark red: Delete/replace an object \\
- Black: Other messages (summary)
\end{tabular} \\
\hline
\end{tabular}

\section*{Handling external objects}

If the external object is in the same SVN repository, then changes in this external object are listed in the commit dialog and committed together with the internal project. If an external object is in another SVN repository, then you are notified about changes in the external project and you have to commit these separately.
An external object has the "externals" property.

See also
- © Chapter 6.6.7.6.2.7 "Command 'Compare"' on page 4331
- SVN help: http://svnbook.red-bean.com/en/1.7/svn.basic.in-action.html\#svn.basic.in-action.mixedrevs)

Context menu
(right-click on
object)
\begin{tabular}{|c|c|}
\hline Q "Compare" & Opens the compare dialog to compare the working copy with the top-level revision. \\
\hline Q "Compare with HEAD version" & Opens the compare dialog to compare the working copy with the HEAD revision. \\
\hline Q "Compare with Revision" & \begin{tabular}{l}
The list entries are highlighted in color according to the object status: \\
- Blue: Modified \\
- Green: Added \\
- Dark red: Deleted \\
- Red: Conflicted \\
- Black: Non-versioned (not in SVN repository) \\
Note: These objects are displayed when the "Show non-versioned objects" option is selected. \\
- Gray: Excluded from commit \\
Note: This is the case when the "Ignore during commit" option is selected for the object.
\end{tabular} \\
\hline ¢ "Clear" & Discards your changes to the working copy. Then the object corresponds to the revision in the SVN repository. \\
\hline \(0 \equiv\) "Show log" & Shows the version history of the selected object. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Properties" & \begin{tabular}{l} 
Opens the "SVN Properties" dialog. The properties are displayed there and you \\
can edit them.
\end{tabular} \\
\hline Move to change list & Note: This command has not been implemented yet. \\
\hline
\end{tabular}

\section*{Command 'Compare'}

\section*{Symbol:}

Function: This command opens a tab that shows the result of the comparison of your working copy and the BASE revision. The base revision is the top-level revision in the SVN repository.
Call:
- Menu bar: "Project \(\rightarrow\) SVN".
- Context menu

Requirement: The object is versioned, it was modified locally, and it does not contain any conflicts.
Multiple tabs can be open at the same time with the comparison of different objects.

\section*{Comparison by object type}

The comparison dialog makes use of the functionality of the CODESYS command "Project \(\rightarrow\) Compare". In this way, objects are compared according to their object type.

\section*{See also}
- \& Chapter 6.4.1.5 "Comparing projects" on page 1817

\section*{Command 'Compare with HEAD revision'}

Symbol: \&
Function: This command opens a tab that shows the result of the comparison of your working copy and the HEAD revision. The HEAD revision is the top-level revision in the branch. You can revert specific changes that were committed to the HEAD revision.
Call: Context menu: "SVN"
Requirement: The object is versioned and not conflicted.
Multiple tabs can be open at the same time with the comparison of different objects.

\section*{Comparison by object type}

The comparison dialog makes use of the functionality of the CODESYS command "Project \(\rightarrow\) Compare". In this way, objects are compared according to their object type.

\section*{See also}
- \& Chapter 6.4.1.21.3.5.21 "Command 'Compare"' on page 2607

\section*{Command 'Compare with revision'}

Symbol: Q

Function: This command opens the "Project log" dialog or "Log-<object>" where the version history is displayed from the project or an object of the CODESYS project. Here you can select a revision. A tab opens and shows the result of the comparison of your working copy and the revision.

Call: Context menu: "SVN"
Requirement: The object is versioned and not conflicted.
Multiple tabs can be open at the same time with the comparison of different objects.

\section*{Comparison by object type}

The comparison dialog makes use of the functionality of the CODESYS command "Project \(\rightarrow\) Compare". In this way, objects are compared according to their object type.

See also
- 艺 "Tab 'Project log', Dialog 'Log - <object>"" on page 4338
- \(\sum^{*}\) Chapter 6.4.1.21.3.5.21 "Command 'Compare'" on page 2607

\section*{Command 'Compare to remote project...'}

Symbol: «u
Function: This command opens the dialog "Select Remote Project for Comparison".
Call: Menu bar: "Project \(\rightarrow\) SVN".
See also
- \({ }^{*}\), Chapter 6.4.1.21.3.5.21 "Command 'Compare'" on page 2607

Dialog 'Select
Remote Project for Comparison'
Table 784: "URL of SVN repository"
\begin{tabular}{|c|c|}
\hline  & \begin{tabular}{l}
URL of the project in the SVN repository that is compared. \\
Example: file:///D:/SVN repository/trunk/Controldef.project \\
As soon as a valid SVN repository is specified, you can click the adjacent button or use the options to browse in "Revision" and select a project.
\end{tabular} \\
\hline 15 & \begin{tabular}{l}
The label on the button corresponds to the selected revision: \\
- "HEAD": Newest revision (top revision) \\
- "15": Revision number of the selected revision \\
- "12/23/2016 11:59:59 (UTC)": Change date of the selected revision (UTC) \\
After clicking the button, the dialog "Select revision" opens. \\
Note: The dialog provides the same options as the "Revision" group.
\end{tabular} \\
\hline .... & Opens the dialog "Browse SVN repository" to search the SVN repository. \\
\hline
\end{tabular}

Table 785: "Checkout options"
\begin{tabular}{|l|l|}
\hline "Omit externals": & \(\boxed{ } \quad\) : External objects are not compared. \\
\hline
\end{tabular}

Table 786: "Revision"
\begin{tabular}{|l|l|}
\hline \multicolumn{2}{|l|}{ Options for selecting a specific revision } \\
Note: the current valid selection is also displayed next to the SVN repository URL.
\end{tabular} \left\lvert\, \begin{tabular}{l|l|}
\hline "HEAD" The HEAD revision is selected. This is the latest revision (top revision) within \\
a branch.
\end{tabular}\(\quad\)\begin{tabular}{l} 
©: A specific revision is selected by the revision number. \\
Example: 3
\end{tabular}\right.

Table 787: "compare options"
\begin{tabular}{|l|l|}
\hline "Ignore Whitespace" & \begin{tabular}{l} 
V: No comparison of whitespace characters. Semantically relevant whitespaces, \\
such as in strings, are compared anyway.
\end{tabular} \\
\hline "Ignore Comments" & \(\boxed{\square}:\) No comparison of comments. \\
\hline "Ignore Properties" & \begin{tabular}{l}
\(\square:\) No comparison of properties. Folders, the property "Exclude from build", and \\
POU images are not compared. \\
See: Dialog 'Properties'
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l}
\hline "OK" & Compares the SVN project with the working copy.
\end{tabular}
See also
- \({ }^{4}\) Chapter 6.6.7.6.2.2 "Command 'SVN Repository Browser'" on page 4322
- 鼠 Chapter 6.4.1.21.4.11 "Dialog 'Properties'" on page 2753

\section*{Command 'Include externals to project', Command 'Include externals'}

Symbol: x
Function: These commands open the dialog "Include externals".
Call:
- Menu bar: "Project \(\rightarrow\) SVN".
- Context menu: "SVN"

Requirement: An object is selected in the object tree. The external objects are linked below that. If you have selected nothing or the project root directory, then the command "Include externals to project" is available. If you have selected an object, then the command "Include externals" is available.


The same external objects cannot be linked multiple times at different locations in the same project. This leads to problems in CODESYS because of conflicts with the internal identification of the object.

\section*{Dialog 'Include externals'}

Table 788: "URL of SVN repository"
\begin{tabular}{|c|c|}
\hline leillidisw repo A/tunk DSTest.proect/Source & \begin{tabular}{l}
URL of the external object that is linked. The object to be linked is versioned and can have sub-objects. \\
External objects are located at another location in the SVN repository than the project. It can even be in another SVN repository. \\
Example: file:///D:/SVN repo A/trunk/DSTest.project/ GlobalTextList \\
Note: The objects that should be linked below the selected object must have a matching object type. For example, only a task can be linked below the "Task configuration" object.
\end{tabular} \\
\hline 15 & \begin{tabular}{l}
Opens the dialog "Select revision". Here you can select a revision. \\
The button is labeled with the currently selected revision: \\
- "HEAD": Newest revision (top revision). Default \\
- "15": Revision number of the selected revision \\
- "12/23/2016 11:59:59 (UTC)": Change date of the selected revision (UTC) \\
Note: The dialog provides the same options as the "Revision" group.
\end{tabular} \\
\hline \(\ldots\) & Opens the "SVN repository browser" dialog Here you can browse the SVN repository. \\
\hline
\end{tabular}

Table 789: "Revision"
\begin{tabular}{|l|l|}
\hline \multicolumn{2}{|l|}{\begin{tabular}{l} 
Options for selecting a revision \\
Note: the current valid selection is also displayed in the buttons next to the SVN repository URL.
\end{tabular}} \\
\hline "HEAD" & \(\bullet\) : Latest revision (top revision) selected in a branch. \\
\hline "Revision" & \begin{tabular}{l} 
©: A specific revision by the revision number. \\
Example: 3
\end{tabular} \\
\hline "Date" & \begin{tabular}{l} 
©: A specific revision by the modification date. \\
Example: \(12 / 23 / 2016 \quad 11: 59: 59\)
\end{tabular} \\
\hline "Use UTC Time": & \(\boxed{V}:\) Modification date in universal time. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "OK" & \begin{tabular}{l} 
Adds the external object and its sub-objects with the property svn: externals \\
to your project (below the selected object). The working copy is updated and the \\
external object is overlaid with the 天symbol. \\
\\
Example: source (external device Source) \\
\\
\\
\\
Note: If the linking fails (for example when adding a device below a task configu- \\
ration), then the complete operation fails and reverts back. \\
\\
\\
Note: Renaming or moving individual external objects is permitted inly within an \\
external tree, whereby it is not permitted to move the top object. \\
\\
To move a complete tree, you have to remove it and link it to another location.
\end{tabular} \\
\hline
\end{tabular}

\begin{abstract}
".. You should seriously consider using explicit revision numbers in all of your externals definitions. Doing so means that you get to decide when to pull down a different snapshot of external information, and exactly which snapshot to pull. Besides avoiding the surprise of getting changes to third-party repositories that you might not have any control over, using explicit revision numbers also means that as you backdate your working copy to a previous revision, your externals definitions will also revert to the way they looked in that previous revision, which in turn means that the external working copies will be updated to match the way they looked back when your repository was at that previous revision. For software projects, this could be the difference between a successful and a failed build of an older snapshot of your complex codebase. ..."

This is cited from:
http://svnbook.red-bean.com/nightly/en/svn.advanced.externals.html).
\end{abstract}

\section*{Command 'Ignore on commit'}

Function: This command identifies an object and adds it to the "ignore-on-commit" list. Then it is deactivated in the commit dialog by default.

Call: Menu bar: "SVN"
Requirement: At least one object is available that is not in the change list ignore-oncomment.

Objects of the "ignore-on-commit" list are overlaid with the symbol in the object tree. By default, they are not selected in the commit dialog, unless a dependency of a selected object requires it. These objects can always be selected manually in the dialog.

See also
- \(\Leftrightarrow\) Chapter 6.6.7.6.2.27 "Command 'Un-Ignore on commit'" on page 4343

\section*{Command 'SVN Info'}

Function: This command provides information about the selected object in the SVN repository. The "SVN Information" dialog opens for this purpose.

Call: Context menu: "SVN"
Requirement: A versioned object (with SVN link) is selected in the object tree.

\section*{Dialog 'SVN}

Information

\section*{Example}
```

Name: Device_4\Plc Logic\Application\PLC_PRG
URL: file:///D:/SVN repository/trunk/ControlABC.project/Device/Plc
Logic/Application/PLC_PRG/svnobj
Repository Root: file:///D:/SVN repository/
Repository UUID: 185325d7-73eb-e54b-ab50-206aa23c8b42
Revision: 29
Node Kind: File
Schedule: Normal
Last Changed Author: a.mayer
Last Changed Rev: 8
Last Changed Date: 17.01.2017 12:33:51
Text Last Updated: 17.01.2017 12:33:51
Checksum: d5fb4d91ebaea06f26bcdb15942724d57932b6a3

```

\section*{Command 'Show properties'}

Symbol:
Function: This command opens the "SVN Properties" dialog. Here you can edit the properties of the versioned object.

Call: Context menu: "SVN"
Requirement: A versioned, unlocked object is selected.

\section*{Dialog 'SVN \\ Properties'}

Table 790: "properties for: <object name>"
\begin{tabular}{|l|l|}
\hline "Name" & \begin{tabular}{l} 
Name of the property \\
Example: myprop: customer-number \\
Note: SVN has some reserved properties. Example: svn:mime-type
\end{tabular} \\
\hline "Value" & \begin{tabular}{l} 
Example: 1234 \\
Double-click in the field to edit the value..
\end{tabular}
\end{tabular}
\begin{tabular}{|l|l|} 
"Add" & Opens a dialog to define another property with its value. \\
\hline "Remove" & Deletes the selected property.
\end{tabular}
"Show binary properties" |ی: The binary properties are also displayed.
\begin{tabular}{|l|l|} 
"Reset" & Resets the changes displayed in green. \\
\hline "OK" & Accepts the changes. \\
\hline
\end{tabular}

See also
- http://svnbook.red-bean.com.

\section*{Command 'Get lock'}

Symbol: e
Function: This command locks the object explicitly for you. The "Lock Message" dialog opens for this purpose.
Call: Context menu: "SVN"
Requirement: The versioned object is not locked (not overlaid with the symbol).

\section*{Dialog 'Lock \\ Message'}
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
"Enter the reason why you lock \\
the object:"
\end{tabular} & \begin{tabular}{l} 
Lock message \\
Example: Locked for processing task 123
\end{tabular} \\
\hline Button "Recent Message" & \begin{tabular}{l} 
Shows message in the dialog that have already been used. There you select one \\
in order to use the lock message.
\end{tabular} \\
\hline "Recursive" & : The object is locked with all subordinate child objects. \\
\hline "OK" & \begin{tabular}{l} 
Locks the object \\
When the lock is successful, the object (in the object tree) is overlaid with the \\
symbol.
\end{tabular} \\
\hline
\end{tabular}

\section*{Command 'Steal locks'}

Symbol: :
Function: This command steal the lock of the object. The "Lock Message" dialog opens for this purpose.

Call: Context menu: "SVN"
Requirement: The versioned object is locked by someone else (overlaid with the \({ }_{0}\) symbol).

\section*{Dialog 'Lock \\ Message'}
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
"Enter the reason why you lock \\
the object:"
\end{tabular} & \begin{tabular}{l} 
Lock message \\
Example: a.mayer had to steal the lock because the changes \\
need to be implemented so urgently.
\end{tabular} \\
\hline "Recent Message" & \begin{tabular}{l} 
Shows message in the dialog that have already been used. There you select one \\
in order to use the lock message.
\end{tabular} \\
\hline "Recursive" & 包: The lock is stolen by the object and all subordinate child objects. \\
\hline "OK" & \begin{tabular}{l} 
Steals the lock. \\
When the stolen lock is successful, the object (in the object tree) is overlaid with \\
the symbol.
\end{tabular} \\
\hline
\end{tabular}

\section*{Command 'Release lock'}

Symbol:
Function: This command releases the lock of an object.
Call: "Context menu \(\rightarrow\) SVN"
Requirement: The object is locked.

\section*{Command 'Release locks recursively'}

Symbol:
Function: This command releases the lock of an object explicitly with all of its subordinate objects.
Call: "Context menu \(\rightarrow\) SVN"
Requirement: The object is locked.

\section*{Command 'Show log', Command 'Show project log'}

\section*{Symbol: \(\equiv\)}

Function: These commands open the tab "Project log" or "Log - <object>". The version history of the project or an object of the CODESYS project is displayed in the tab.
Call:
- Menu bar: "Project \(\rightarrow\) SVN".
- "Context menu \(\rightarrow\) SVN"

If you select nothing or the base node in the object tree, then the history of the entire project is displayed ("Show project log"). If you select one or more objects, then the history of these elements is displayed ("Show log").
Multiple tabs can be open at the same time with the version history of different objects.

Tab 'Project log',
Dialog 'Log -
<object>'
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
Upper area \\
\(\bullet ~ " R e v i s i o n ": ~ R e v i s i o n ~\)
\end{tabular} \\
number
\end{tabular}\(\quad\)\begin{tabular}{l} 
List of all revisions of the project or the selected objects in the information. The \\
first 100 revisions are displayed by default. The "Next 100 " and "All" buttons are \\
provided for displaying more or all revisions. \\
- "Author" \\
- "Date" \\
"Message": Message \\
entered at commit
\end{tabular}\(\quad\)\begin{tabular}{l} 
context menu commands are described below.
\end{tabular}

Table 791: Dialog "Filter"
\begin{tabular}{|l|l|}
\hline "Revision range" & \begin{tabular}{l} 
The displayed revisions can be filtered by "Head", "Revision", or "Date". \\
V: The option fields for "Start revision" and "End revision" are editable.
\end{tabular} \\
& \begin{tabular}{l} 
"Use UTC time": Date display in universal time. \\
For more detailed information, refer to the description "Dialog 'Select revision'".
\end{tabular} \\
\hline "Message contains" & Display of revision logs that contain a special text in the "Message" \\
\hline "Author contains" & Display of revision logs of the specified author \\
\hline "Path contains" & Display of revision logs of the specified path \\
\hline
\end{tabular}

Table 792: Context menu commands of the revisions
\begin{tabular}{|l|l|}
\hline "Compare with base working \\
copy"
\end{tabular} \begin{tabular}{l} 
Compares the selected revision of the object with the base working copy (without \\
local changes).
\end{tabular}\(|\)\begin{tabular}{lll|}
\hline "Com with working copy" & Compares the selected revision of the object with the working copy. \\
\hline "Compare with HEAD revision" & Compares the selected revision of the object with the HEAD revision. \\
\hline \begin{tabular}{l} 
"Compare with previous \\
revision"
\end{tabular} & Compares the selected revision of the object with the previous revision. \\
\hline "Update item to revision" & \begin{tabular}{l} 
Updates the object to the selected revision. \\
Note: Changes of the project by this command cannot be committed. \\
For VSS users: This is comparable to loading an older version without checkout. \\
To revert a previous commit, the command "Revert to this revision" has to be \\
used.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Revert to this revision" & \begin{tabular}{l} 
Reverts the object to the selected revision. \\
This command does not have an effect on the SVN repository as long as the \\
changes are not committed. Internally, SVN reverts the merges for all changes \\
that were made after the selected revision in order to revert the changes of the \\
preceding commits.
\end{tabular} \\
\hline "Edit author" & Opens a dialog for changing the author of the revision. \\
\hline "Edit log message" & Opens a dialog for changing the log message of the revision. \\
\hline "Revision properties" & \begin{tabular}{l} 
Opens the dialog "Revision properties" where the properties are displayed. \\
In the dialog, you can activate the "Add" and "Remove" properties and the option \\
"Show binary properties".
\end{tabular} \\
\hline "Create branch/tag from this \\
revision" & Creates a branch or tag from the selected revision. \\
\hline "Browse SVN repository" & Opens the "SVN repository browser" dialog \\
\hline "Copy to clipboard" & \begin{tabular}{l} 
Copies log details of the selected revision to the clipboard This is the revision \\
number, author, date of revision, log message, and the list of changes objects for \\
each revision.
\end{tabular} \\
\hline
\end{tabular}

See also
- « \({ }^{*}\) Chapter 6.6.7.6.2.7 "Command 'Compare"" on page 4331
- \(\xi^{*}\) Chapter 6.6.7.6.2.8 "Command 'Compare with HEAD revision'" on page 4331
- \(\Leftrightarrow\) Chapter 6.6.7.6.2.9 "Command 'Compare with revision'" on page 4331
- \(\Rightarrow\) Chapter 6.6.7.6.2.2 "Command 'SVN Repository Browser'" on page 4322
- \(\quad\) Chapter 6.6.7.6.3.3 "Dialog 'Select revision"" on page 4351

\section*{Command 'Revert', Command 'Revert project'}

\section*{Symbol: 子}

Function: This command opens the "Revert" dialog. In the dialog, select the objects whose local changes should be reverted, and those that are reverted to the state of the base revision of the working copy.

\section*{Call:}
- Menu bar: "Project \(\rightarrow\) SVN".
- "Context menu \(\rightarrow\) SVN"

If you select nothing or the main node in the device tree, then all modified objects are listed in this dialog ("Revert project"). If you selected one or more objects, then only the changes to this object are listed and recursively their sub-objects ("Revert").

\section*{Dialog 'Revert'}
\begin{tabular}{|l|l|}
\hline "Group externals" & \(\checkmark\) : The external definitions are grouped by their external storage locations. \\
\hline "Keep locks" & \(\boxed{\square}\) : The lock is retained for all files that are modified by the revert command. \\
\hline "Select/deselect all" & \\
\hline
\end{tabular}

When external objects are deleted, Professional Version Control cannot restore this data in SVN offline mode. The user is prompted how to proceed:
- Switch back to SVN online mode and call the external objects.
- Connect now to the SVN server one time in order to complete the current operation, but afterwards switch back to SVN offline mode.
- Skip the retrieval of the external objects. They can be fetched later by updating the project.

See also
- \(\Leftrightarrow\) Chapter 6.6.7.6.2.21 "Command 'Revert to revision', Command 'Revert project to revision'" on page 4340

\section*{Command 'Revert to revision', Command 'Revert project to revision'}

Symbol:
Function: This command opens the "Select revision" dialog. In this dialog, you select the revision to which the project or the selected objects revert.

Call:
- "Project \(\rightarrow\) SVN"
- "Context menu \(\rightarrow\) SVN"

If nothing or the base node is marked in the object tree, then the entire project is reverted to a specific revision ("Revert project to revision"). If one or more objects are selected, then these objects and their sub-objects are reverted ("Revert to revision").

Dialog 'Select For a description of the dialog, refer to the section "Select revision".
revision'
See also
- \({ }^{4}\) Chapter 6.6.7.6.3.3 "Dialog 'Select revision'" on page 4351
- 乡y Chapter 6.6.7.6.2.20 "Command 'Revert', Command 'Revert project"' on page 4339
- \({ }^{*} \boldsymbol{y}\) Chapter 6.6.7.6.2.19 "Command 'Show log', Command 'Show project log'" on page 4337

\section*{Command 'Update', Command 'Update project'}

Symbol:
Function: This command commits changes in the SVN repository to the project. The update is performed with the HEAD revision.
Call:
- Menu bar: "Project \(\rightarrow\) SVN".
- "Context menu \(\rightarrow\) SVN"

If nothing or the main node is selected, then the entire project is updated ("Update project"). If one or more objects are selected, then these objects and their sub-objects are updated ("Update SVN").

The following cases are possible:
- Projects are added to the project that are present in the SVN repository, but not in the project. In this case, the message "Added <object>" is issued to the message view.
- Objects that no longer exist in the SVN repository, but are present in the project locally (and not marked as "added"), are treated according to the Subversion standard procedure: If local changes are present, then the object remains in the project as unversioned. If there are no local changes, then the object is also deleted locally because the user can retrieve the object from an older version at any time. In this case, "Deleted object" is issued to the message view.
- Versioned objects that exist in both the SVN repository and the project are updated if they are different. Three cases to observe:
- No local changes have been made since the last update: In this case, the local object is overwritten by the contents from the SVN repository. The message "Object updated" is issued to the message view.
- Local changes have been made since the last update and the corresponding object type can be merged. When versions have been merged successfully, the message "Objects merged" is issued to the message view. If the command is not executed successfully, then the object is marked as "Conflicted object" in the object tree and the message "Conflicted object" is issued.
- Local changes have been made since the last update and the corresponding object type cannot be merged. In this case, the object is marked as "Conflicted object" in the object tree and the message "Conflicted object" is issued.
If only some of the objects are updated, it may be that objects with the same name already exist. For example, this situation can come from moving objects to a folder.
For this conflict, you can react in the following ways:
- Do nothing and leave the conflict-causing objects as they are.
- Update (and remove) the conflicting objects in order to correct the conflict.
- Update the entire project in order to remove all conflicting objects and correct the conflict.

See also
- \& Chapter 6.6.7.6.2.23 "Command 'Update to revision"' on page 4341

\section*{Command 'Update to revision'}

Symbol:
SFunction: This command opens the "Update" dialog. In the dialog, the revision is defined for updating the project.
Call:
- "Project \(\rightarrow\) SVN"
- "Context menu \(\rightarrow\) SVN"

If you select nothing or the base node in the object tree, then the entire project is updated to a revision ("Update project to revision"). If you select one or more objects, then these objects are updated and their sub-objects are updated recursively ("Update to revision"). As an option, you can define that the sub-objects are not updated.
The behavior of the updating process (for example merging of conflicts) is similar to the "Update project" and "Update" commands.

\section*{Dialog 'Update'}
\begin{tabular}{|l|l|}
\hline "HEAD" & \begin{tabular}{l} 
©: This command behaves the same as the "Update" and "Update project" \\
commands.
\end{tabular} \\
\hline "Revision" & \begin{tabular}{l} 
O: The revision to which was last updated is selected by the revision number. \\
\(\ldots:: ~ O p e n s ~ t h e ~ d i a l o g ~ " L o g " ~ f o r ~ s e l e c t i n g ~ t h e ~ r e v i s i o n . ~\)
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Date" & \begin{tabular}{l} 
O: The revision to which was last updated is selected by the modification date. \\
"Use UTC time": \\
": The date is displayed in universal time.
\end{tabular} \\
\hline "Recursive" & \begin{tabular}{l} 
目: Default setting. The selected part is updated recursively. This means that all \\
elements below the selected object are also updated.
\end{tabular} \\
\hline "Omit externals" & \(\boxed{\square}:\) External objects are not updated. \\
\hline
\end{tabular}

See also
- \(\Leftrightarrow\) Chapter 6.6.7.6.2.22 "Command 'Update', Command 'Update project' " on page 4340
- \(\Rightarrow\) Chapter 6.6.7.6.3.3 "Dialog 'Select revision'" on page 4351

\section*{Command 'Update only this'}

Symbol: к
Function: The command updates the selected objects. In contrast to the "Update" and "Update to Revision" commands, the child objects are not updated.
Call: "Context menu \(\rightarrow\) SVN"
See also
- \(\Leftrightarrow\) Chapter 6.6.7.6.2.22 "Command 'Update', Command 'Update project' " on page 4340
- \({ }^{\circ}\) Chapter 6.6.7.6.2.23 "Command 'Update to revision'" on page 4341

\section*{Command 'Disconnect project from SVN'}

Symbol: \(x]\)
Function: This command deletes all connections of the current project to SVN by converting the project into a non-versioned project.
Call: Menu bar: "Project \(\rightarrow\) SVN".


Use the command "Connect to existing project" to connect to the SVN repository again at a later time.

See also
- \(\quad\) Chapter 6.6.7.6.2.31 "Command 'Connect to existing project"' on page 4345

\section*{Command 'Switch'}

\section*{Symbol: \(\uparrow\)}

Function: This command opens the "SVN switch" dialog. In this dialog, you specify a URL in the SVN repository to which the current working copy of the project is updated. The command switches a project from a branch or tag to another.
Call: Menu bar: "Project \(\rightarrow\) SVN".
Requirement: The project is versioned.

Dialog 'SVN
switch'
\begin{tabular}{|l|l|}
\hline "From" & Current SVN URL of the project \\
\hline "To" & Input field for the target URL in SVN \\
& \begin{tabular}{l} 
•HEAD": The "Select revision" dialog opens. \\
\\
\\
\\
\\
\hline\(\ldots .\). The "SVN Repository Browser" dialog opens. There you select the target \\
URE SVN repository.
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{\wedge}\) Chapter 6.6.7.6.2.2 "Command 'SVN Repository Browser'" on page 4322

\section*{Command 'Un-Ignore on commit'}

Function: This command removes an unversioned object from the ignore list so that the object is checked by default on commit.
Call: Context menu: "SVN"
Requirement: The command "Ignore on commit" was executed for the object. The object is marked with the symbol.
See also
- « Chapter 6.6.7.6.2.12 "Command 'Ignore on commit"" on page 4335

\section*{Command 'SVN Cleanup'}

Function: This command opens the "SVN Cleanup" dialog. In the dialog, you define actions that are performed when cleaning up the SVN working copy.
Call: Menu bar: "Project \(\rightarrow\) SVN".

\section*{Dialog 'SVN \\ Cleanup'}

Table 793
\begin{tabular}{|l|l|}
\hline \multicolumn{2}{|l|}{ "Internal SVN working copy" } \\
\hline "Update time stamps (speeds \\
up SVN status display)" & \begin{tabular}{l} 
Vhis leads to a reduction in the compare time for future checks. It is not neces- \\
sary to execute this in regular intervals in the normal workflow.
\end{tabular} \\
\hline \begin{tabular}{l} 
"Vacuum cached pristine \\
copies (may reduce the size of \\
your project file)"
\end{tabular} & \begin{tabular}{l} 
V: Cleans the buffer for the original copies by deleting older versions that are no \\
longer referenced by the current project. Advantage: The size of the project file \\
is reduced. Disadvantage: If you downgrade to older revisions, or if you switch \\
between different branches, then the retrieved data size will become larger.
\end{tabular} \\
\hline "Clear work queue and force \\
unlock of SVN internal data \\
structures (emergency only!)" & \begin{tabular}{l} 
V: Cleans up the internal SVN task queues and unlocks internal SVN data \\
structures. This should never be necessary during normal work by Professional \\
Version Control. \\
Note: Use this option only if errors occur for SVN commands due to locked \\
working copies. When this is the case, it refers to an error in Professional \\
Version Control. Then please send us an error report (if possible with steps to \\
repeat) to the CODESYS support. \\
Info: These are administrative locks that are internal locks in the SVN
\end{tabular} \\
working copy. These locks are not set up by context menu commands. For \\
more information, refer to the section "The three meanings of locks" in: \\
http://svnbook.red-bean.com/en/1.8/svn.advanced.locking.html
\end{tabular}
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
"Revert all local changes (use \\
with care!)"
\end{tabular} & Reverts all local changes to the original status in the SVN repository. \\
\hline "Release all locks" & \begin{tabular}{l} 
Releases all advisory locks in the project (locks visible to the user). These locks \\
are activated by "Acquire lock" and "Steal lock".
\end{tabular} \\
\hline \begin{tabular}{l} 
"Revalidate all locks against \\
the repository (they could have \\
been stolen)"
\end{tabular} & \begin{tabular}{l} 
Checks whether the locally available advisory locks are still valid or have been \\
stolen by someone else for example. All invalid locks are removed.
\end{tabular} \\
\hline "Status caches" & \begin{tabular}{l} 
"Clear all caches and refresh \\
status icons"
\end{tabular} \\
\begin{tabular}{l} 
Deletes all internal caches that Professional Version Control has and updates \\
the status icons. Required only if it issues an error in Professional Version \\
Control through which the caches or the status display are inconsistent.
\end{tabular} \\
\hline
\end{tabular}

\section*{Command 'Clear authentication data'}

Function: This command opens the "CODESYS" dialog. In this dialog, define the caches that will be deleted.

Call: Menu bar: "Project \(\rightarrow\) SVN".

\section*{Dialog 'CODE- \\ SYS'}

The authentication memory contains the authentication data of all SVN repositories for which the user has selected for saving the authorization data. This memory is deleted completely by this command.
"Clear the shared on-disk
The data saved on the computer is deleted. cache."
"Clear the RAM cache of this instance."

The authentication data saved on the computer is stored in \(\% A P P D A T A \%\) \Subversion \auth. This memory path is also used for most other Subversion client applications (for example, TortoiseSVN and AnkhSVN). Therefore, deleting the authentication data affects these applications as well.

\section*{Command 'Merge changes'}

Symbol: \(\uparrow\)
Function: This command opens the "Merge" dialog. In this dialog, you determine the revisions with the changes to be merged with the working copy of the project.

Call: Menu bar: "Project \(\rightarrow\) SVN".
Requirement: The project is linked to SVN.

\section*{Dialog 'Merge' 'c"}
\begin{tabular}{|c|c|}
\hline "Kind of merge" & \begin{tabular}{l}
- "Sync/Reintegrate/Symmetric merge": All missing changes from the trunk (or a different branch) are synchronized with this branch. \\
- "Cherry pick": Integrates specifically selected revisions from one branch to another branch. This is necessary, for example, if any error trapping has to be ported back to an older version.
\end{tabular} \\
\hline "Merge source" & \begin{tabular}{l}
SVN URL of the SVN repository \\
- Input field \\
- "HEAD": HEAD revision \\
- ....: Dialog "SVN Repository Browser" opens for selecting the SVN repository.
\end{tabular} \\
\hline "Define start and end revision" & Select this option to merge a cohesive range of revisions with the working copy. \\
\hline "Start revision" & Defines the range of revisions that are merged with the working copy: \\
\hline "End revision" & \begin{tabular}{l}
- "HEAD": HEAD revision \\
- "Revision": Start and end revision of the range \\
- "Date": Date of the start and end revisions
\end{tabular} \\
\hline "Define revision range" & \begin{tabular}{l}
Select this option to merge individual revisions with the working copy. You can also highlight the individual revisions in the "Log" dialog. \\
Note: When defining ranges, CODESYS SVN behaves like other graphical clients, such as Tortoise SVN), and not like the command-line client. Example: For a range of \(4-7\), revisions \(4,5,6\), and 7 are merged. \\
See also: Merging a Range of Revisions
\end{tabular} \\
\hline "Dry run (simulation)" & This command is executed without changing the working copy. Files that are changed during an actual merge are displayed, as well as ranges where conflicts occur. \\
\hline "Record only" & \: The revision is marked as "merged" without actually performing the merge. \\
\hline "Ignore ancestry" & \(\checkmark\) : SVN uses path-based differences only, not history-based differences. \\
\hline
\end{tabular}
- \({ }^{4}\) Chapter 6.6.7.6.3.3 "Dialog 'Select revision'" on page 4351
- \({ }^{4}\), Chapter 6.6.7.6.2.2 "Command 'SVN Repository Browser'" on page 4322

\section*{Command 'Connect to existing project'}

Symbol:
Function: This command opens the "Connect to SVN repository" dialog. In the dialog, you define the URL and the revision of the SVN repository with which the unversioned project is connected.

Call: Menu bar: "Project \(\rightarrow\) SVN".
Requirement: The project is disconnected from SVN.

\section*{NOTICE!}
```

Only users who have read access to the entire project (see the CODESYS user and access management) can import the project into the SVN repository or can link to an existing database project.

```

\section*{NOTICE!}

This command functions reliably only when the project has already been imported into SVN and then disconnected with the command "Disconnect project from SVN".

Dialog 'Connect
to SVN reposi-
tory'
\begin{tabular}{|l|l|}
\hline "URL of existing project" & \begin{tabular}{l} 
URL of the SVN repository \\
"HEAD": Selection of the revision in the "Select revision" dialog \\
\(\ldots\) \\
\(\ldots .\).
\end{tabular} \\
\hline "Chelection of the SVN repository in the "SVN Repository Browser"
\end{tabular}

See also
- \({ }^{\text {® }}\) Chapter 6.6.7.6.2.2 "Command 'SVN Repository Browser'" on page 4322

\section*{Command 'Resolve conflict'}

Symbol: Q
Function: This command opens the "<object>" dialog. In the dialog, the conflicts are displayed and functions for resolving conflicts are prepared in order to merge changes.
Call: Context menu of the object.
Requirement: The object has a conflict that has occurred by updating the object with local changes.

Dialog '<object>'
\begin{tabular}{|l|l|}
\hline "Compare" & \begin{tabular}{l} 
The local objects are displayed on the left side, and the version from the SVN \\
repository is displayed on the right side.
\end{tabular} \\
\hline "Use mine" & A local change is used. \\
\hline "Use yours" & A change of the version from the SVN repository is changed. \\
\hline "Apply" & \begin{tabular}{l} 
All changes are accepted that you made in this dialog. The status of the object is \\
changed.
\end{tabular} \\
\hline "Cancel" & \begin{tabular}{l} 
Cancels all changes that you made in this dialog. But the object keeps the \\
conflicted status.
\end{tabular} \\
\hline
\end{tabular}

\section*{Command 'Work in offline mode'}

Function: This command switches to SVN offline mode. In SVN offline mode, the implicit locking and all commands that access the SVN repository are not possible.

\section*{Call:}
- Menu bar: "Project \(\rightarrow\) SVN".
- Context menu: "SVN"

Requirement: The project is linked to SVN.
When switching back to SVN online mode, all present locks on the working copy are checked against the server. If this locking is invalid, then it is released.

The user on a machine wants to make changes to the project without disconnecting the connection. At the moment, there is not connection to the server. Despite this, when automatic locking is activated, work is possible because the SVN offline mode deactivates the automatic lock temporarily.

\section*{Command 'Copy (Branch/Tag)'}

Symbol: \(\uparrow\)
Function: This command opens the "SVN Copy Branch/Tag" dialog. There you can "Branch" or "Tag" a revision of your project. A specific revision of your project is saved there at this position. A branch is normally used in order to save changes isolated in one version. A tag is used for marking a specific state, for example a shipping version. Internally, it is copied not in the actual sense, but more refers to the revision.

Call: Menu bar: "Project \(\rightarrow\) SVN".
Requirement: The project is versioned.

Dialog 'SVN
Copy (Branch/
Tag)'
Table 794: "SVN repository"
\begin{tabular}{|l|l|}
\hline "From" & \begin{tabular}{l} 
SVN path of the current project \\
Example: https://svnserver/repository/trunk/ \\
ControlABC.project
\end{tabular} \\
\hline "To" & \begin{tabular}{l} 
Target path in the SVN repository for the copy operation \\
Example of tag: https://svnserver/repository/tags/V4.4.4.4/ \\
ControlABC.project \\
\(\ldots .\).
\end{tabular} \\
\hline
\end{tabular}

Table 795: "Log message"
\begin{tabular}{|l|l|}
\hline Input field & \begin{tabular}{l} 
Comment the change in a log message. \\
Example: Tag for version 4.4.4.4 created.
\end{tabular} \\
\hline "Recent Messages" & \begin{tabular}{l} 
Opens the dialog "Recent Messages" to display the last log messages. You can \\
click a log message to accept it.
\end{tabular} \\
\hline
\end{tabular}

Table 796: "Create copy from"
\begin{tabular}{|l|l|}
\hline \begin{tabular}{l} 
"Working copy (including local \\
changes)"
\end{tabular} & \begin{tabular}{l} 
The new branch/tag refers to the working copy including all local changes. The \\
local changes are committed to the SVN repository for this purpose.
\end{tabular} \\
\hline \begin{tabular}{l} 
"Base revision of working copy \\
(<revision number>)"
\end{tabular} & \begin{tabular}{l} 
The new branch/tag refers to the base revision of your working copy whose \\
revision number is displayed in the parentheses. If the working copy already \\
contains local changes, then these are not committed to the SVN repository.
\end{tabular} \\
\hline \begin{tabular}{l} 
"HEAD revision of the \\
repository"
\end{tabular} & The new branch/tag refers to the HEAD revision of your project. \\
\hline \begin{tabular}{l} 
"Specific revision in SVN \\
repository"
\end{tabular} & \begin{tabular}{l} 
The new branch/tag refers to a revision that is displayed on the adjacent button. \\
Click the button to change the revision. The "dialog opens.".
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|l|}
\hline "Switch to new location" After the dialog is confirmed, the working copy switches to the new branch/ \\
& \begin{tabular}{l} 
tag.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "OK" & \begin{tabular}{l} 
The target path is created (as a new tag ../repository/tags/V4.4.4.4 \\
or as a new branch ../repository/branches/new_feature). Then the \\
revision specified in "Create copy from" is copied there.
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{\wedge}\) Chapter 6.6.7.6.2.2 "Command 'SVN Repository Browser'" on page 4322

\section*{Command 'Pending Changes'}

Symbol: ©
Function: The command opens the "Pending Changes" view. All objects are listed there which have changed from the base revision or which are locked.

Call: "View \(\rightarrow\) Pending Changes"

View 'Pending
Changes'

The modified or locked objects are displayed in the lower half of the view. You can use the "Commit", "Revert", and "Update" commands on single or multiple objects. You will find commands for comparing and displaying the version history in the context menu of a selected object.
CODESYS SVN first checks whether or not the storage format of the working copy matches the storage format of the project in the SVN repository. If the storage formats differ, then the revert must not be executed at the object level in order to prevent inconsistencies. In this case, you can delete the entire project (working copy) first with the Save as command as the appropriate storage format, which however can result in smaller data losses. After you select one of the revert options, you will be guided by corresponding dialog prompts.

Double-clicking the object opens the project comparison.
\begin{tabular}{|l|l|}
\hline "Select" & Selection or clearing of all objects \\
\hline "Commit" & Commits local changes to the SVN repository \\
\hline "Revert" & Reverts the local changes to the state of the base revision of the working copy \\
\hline "Update" & \begin{tabular}{l} 
The command commits changes in the SVN repository to the project. The \\
update is performed with the HEAD revision.
\end{tabular} \\
\hline "Keep Locks" & \(\boxed{\square}\) Lock is not released automatically after commit \\
\hline "Storage profile update" & \begin{tabular}{l} 
Note that the storage profile (storage version) of the repository in the working \\
copy differs from the one in the SVN repository. In this case, a commit will \\
change the storage profile in the repository. \\
For actions which require an exchange between the working copy and reposi- \\
tory, you will be guided by corresponding messages to match the storage profile.
\end{tabular} \\
\hline "Recent Messages" & Shows the last used log messages. You can click a log message to accept it. \\
\hline "Messages" & \begin{tabular}{l} 
Type in a log message that comments your change. Example: Bug fix error \\
123
\end{tabular} \\
\hline
\end{tabular}

See also
- \({ }^{\Perp}\) Chapter 6.6.7.6.2.6 "Command 'Commit', Command 'Commit Project'" on page 4328
- \({ }^{\bullet} \gg\) Chapter 6.6.7.6.2.20 "Command 'Revert', Command 'Revert project'" on page 4339
- \(y^{\star}\) Chapter 6.6.7.6.2.22 "Command 'Update', Command 'Update project' " on page 4340
6.6.7.6.3 Dialogs
6.6.7.6.3.1 Dialog 'Options' - 'SVN Settings' ..... 4349
6.6.7.6.3.2 Dialog 'Project Settings' - 'SVN Settings'. ..... 4350
6.6.7.6.3.3 Dialog 'Select revision'. ..... 4351
6.6.7.6.3.4 Dialog 'Subversion Authentication'.... ..... 4352
6.6.7.6.3.5 Dialog 'Automatic locking failed'. ..... 4354
Dialog 'Options' - 'SVN Settings'
Tab 'General' Symbol:Function: This tab includes the basic settings for Professional Version Control.Call: Menu bar: "Tools \(\rightarrow\) Options".

Table 797: "Automatic locking and merging"
\(\left.\left.\begin{array}{|l|l|}\hline \text { "Merge" } & \begin{array}{l}\text { Behavior for the commands "Update", "Merge", or "Switch", when both sides } \\ \text { (working copy and SVN repository) have changed from the base version. } \\ \text { - "Mark all colliding changes as conflicts": The objects are not merged auto- } \\ \text { matically. All changes with a conflict are marked as "With conflict", even if } \\ \text { some of them can be merged automatically. }\end{array} \\ \text { "Merge mergeable changes, mark the others as conflicts": Changes that } \\ \text { can be merged are merged automatically. All others are marked as "With } \\ \text { conflict". } \\ \text { "Merge mergeable changes, ask the user for the others": Changes that can } \\ \text { be merged are merged automatically. The user is prompted for all others. }\end{array}\right\} \begin{array}{l}\text { "Always ask the user, even for mergeable changes": For all changed objects, } \\ \text { the user is prompted, even if some of them can be merged automatically. }\end{array}\right\}\)

\section*{Example of conflict markers}
```

FOO();
>>>>>>>>>>>>>>>>
I := I + 1;
================
I := I + 2;
<<<<<<<<<<<<<<<<<

```

Table 798：＂Server check＂
\begin{tabular}{|l|l|}
\hline ＂Check server for updates and \\
locks＂
\end{tabular}\(|\)\begin{tabular}{l} 
冋．Professional Version Control checks in the specified time interval that objects \\
have been updated on the server．In addition，it checks whether objects are \\
locked or locks have been stolen．
\end{tabular}

Table 799：＂Ignore for comparison＂
\begin{tabular}{|l|l|}
\hline Ignore whitespace & \begin{tabular}{l} 
目：Whitespace differences between the current project and the reference project \\
are ignored．
\end{tabular} \\
\hline Ignore comments & \(\boxed{\square}\) ：Comments in the programming code are excluded from the comparison． \\
\hline Ignore Properties & \(\boxed{\square}:\) Object properties are excluded from the comparison． \\
\hline
\end{tabular}

Some of the SVN options can be overwritten by the project－specific settings．
Project－specific settings are defined in the menu＂Project \(\rightarrow\) Project settings＂， category＂SVN Settings＂．

See also
－\({ }^{4}\) Chapter 6．6．7．6．3．5＂Dialog＇Automatic locking failed＇＂on page 4354
－\({ }^{\circledR}\) Chapter 6．6．7．6．3．2＂Dialog＇Project Settings＇－＇SVN Settings＇＂on page 4350

Tab＇SSH＇Symbol：园
Function：This tab contains the settings for the SSH protocol．
Call：Menu bar：＂Tools \(\rightarrow\) Options＂．

Table 800：＂SSH client implementation＂
\begin{tabular}{|l|l|}
\hline ＂libssh2（recommended）＂ & \begin{tabular}{l} 
Professional Version Control uses Libssh2 for establishing a connection via SSH \\
protocol．This is the recommended setting．
\end{tabular} \\
\hline \begin{tabular}{l}
＂SharpPlink（backwards \\
compatibility）＂
\end{tabular} & \begin{tabular}{l} 
Professional Version Control uses plink．exe for establishing a connection with \\
SSH servers．This option is required only for communication with outdated \\
servers that support the deprecated SSH－1 protocol．
\end{tabular} \\
\hline
\end{tabular}

The SSH configuration can be overwritten by means of the environment vari－ able SVN＿SSH or server－specific by means of the SVN configuration file．

See also
－Tunneling via SSH

\section*{Dialog＇Project Settings＇－＇SVN Settings＇}

Symbol：周
Function：The behavior of the integrated SVN version control system is configured in this dialog．
Call：Menu bar：＂Project \(\rightarrow\) Project Settings＂（＂SVN Settings＂）．
Requirement：A project is open．

Table 801: "Automatic locking and merging"
With these settings, you can overwrite the default settings that were made in the dialog "Tools \(\rightarrow\) Options", category "SVN Settings".
\begin{tabular}{|l|l|}
\hline "Merge" & \begin{tabular}{l} 
Behavior for the commands "Update", "Merge", or "Switch", when both sides \\
(working copy and SVN repository) have changed from the base version.
\end{tabular} \\
\hline "Locks" & \begin{tabular}{l} 
Behavior such as Professional Version Control objects when they are changed \\
locally.
\end{tabular} \\
\hline "Marker" & Behavior for conflicts \\
\hline
\end{tabular}

Table 802: "Settings SVN version info"
"Create SVN_VERSION_INFO constants for IEC access"
\(\checkmark\) : The object SVN_VERSION_INFO is created and includes global constants or variables for the project metadata.
\(\square\) : The object SVN_VERSION_INFO is not available.
When you activate the option, the object is created automatically. When you deactivate the option, the object is removed from the project automatically.

See also
- \({ }^{\mu}\) Chapter 6.6.7.6.3.1 "Dialog 'Options' - 'SVN Settings"' on page 4349

\section*{Dialog 'Select revision'}

Function: This dialog shows the currently selected revision. You can edit the selection there.
\begin{tabular}{|c|c|}
\hline "Revision" & \\
\hline "HEAD" & - : The latest revision (top revision) within a branch is displayed. \\
\hline "Revision" & \begin{tabular}{l}
A specific revision is displayed by the revision number. \\
Example: 3 \\
Tip: Click \(\qquad\) to show the revisions. Then the "Log" dialog opens to display the revisions and the associated actions. The revision that you select there is applied.
\end{tabular} \\
\hline "Date" & \begin{tabular}{l}
A specific revision is checked out by the modification date. This is the highest revision at the given time (the last revision before that time). \\
Example: 12/23/2016 11:59:59 \\
Tip: See section "Revision identifiers" in "Version control with Subversion"
\end{tabular} \\
\hline "Use UTC Time": & \(\checkmark\) : Modification date in universal time is used. \\
\hline "Reset recursively" & \begin{tabular}{l}
: All objects below the selected object are also reset. \\
The action fails if \\
- Objects have been moved in or out of the hierarchy below \\
- Objects outside of the hierarchy would be changed by implicit dependencies
\end{tabular} \\
\hline
\end{tabular}

See also
- \(\xi^{2}\) Chapter 6.6.7.6.2.19 "Command 'Show log', Command 'Show project log'" on page 4337
- "Version control with Subversion", Section "Revision identifier"

\section*{Dialog 'Subversion Authentication'}

The dialogs are used for authenticating the server/client connection. A server or client authentication is performed depending on the initial situation and protocol.
Overview of possible protocols and dialogs
- svn: //: The SVN protocol; either unencrypted or SSL/TLS encrypted
- Can prompt for user name and password (even for an unencrypted connection)
- Can prompt for a server certificate from the dialog for authentication in order to confirm the server if a certificate is unknown, defective, or invalid (for TLS/SSL encryption)
- As an alternative or in addition to the user name and password prompt, the client can also be authenticated with client certificates (for TLS/SSL encryption). The dialogs for authentication open with the client certificate.
- http://: SVN via http, unencrypted
- Can prompt for user name and password
- https://: SVN via http, SSL/TLS encrypted.
- Can prompt for user name and password
- Can prompt for a server certificate from the dialog for authentication in order to confirm the server if a certificate is unknown, defective, or invalid.
- As an alternative or in addition to the user name and password, the client can also be authenticated with client certificates. The dialogs for authentication open with the client certificate.
- svn+ssh: //: The SVN protocol, encrypted through an SSH tunnel. SSH (Secure Shell) is the usual networking tool in Linux/Unix for accessing other computers.
- Can prompt for user name and password
- Prompts for server certificate in the dialog for authentication if the server is still unknown in order to be sure that it is the correct server.

Dialog for authentication with a server certificate

Initial situation: CODESYS (as a client) receives an unknown or defective server certificate.
This dialog shows information about the certificate. There you can confirm the identity of the server.
\begin{tabular}{|l|l|}
\hline "Authentication area" & \begin{tabular}{l} 
Connection that is secured \\
Example: https://svn repository:443
\end{tabular} \\
\hline
\end{tabular}

Table 803: "Certificate information" (for SSL/TLS connections)
\begin{tabular}{|l|l|}
\hline "Host name" & Example: svn repository \\
\hline "Thumbprint" & \\
\hline "Valid from" & \\
\hline "Valid to" & \\
\hline "Issuer" & \\
\hline "Certificate" & \\
\hline
\end{tabular}

Table 804: "SSH server key information" (for SSH connections)
\begin{tabular}{|l|l|}
\hline "Key type" & \\
\hline "Key size (bits)" & \\
\hline "Key thumbprint" & \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Save information to RAM" & \begin{tabular}{l} 
Ø: The certificate is saved to the working memory. Then the client recognizes in \\
the current CODESYS session for future connections. \\
If you restart CODESYS, then you have to accept the certificate again.
\end{tabular} \\
\hline "Save to disk" & \begin{tabular}{l} 
Ø: The certificate is saved on the computer and it is available for future connec- \\
tions. \\
If you restart CODESYS, then the saved certificate is used.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "OK" & Authenticates and established the connection. \\
\hline
\end{tabular}
\(\square\)

\section*{See also}
- Version Control with Subversion

Dialog for Initial situation: The SVN server requires a client certificate for authentication.
authentication
with a client cer
In this dialog, you select the client certificate in order to confirm the identity. tificate

The certificate memory is secured cryptographically and distributed with other SVN clients.
\begin{tabular}{|l|l|}
\hline "Authentication area" & \begin{tabular}{l} 
Connection that is secured \\
Example: https://svn repository:443
\end{tabular} \\
\hline
\end{tabular}

Table 805: "The SSL server requires a client certificate file."
\begin{tabular}{|l|l|}
\hline "File" & Client certificate file \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Save information to RAM" & \begin{tabular}{l} 
Ø: The certificate is saved to the working memory. Then the client recognizes in \\
the current CODESYS session for future connections. \\
If you restart CODESYS, then you have to accept the certificate again.
\end{tabular} \\
\hline "Save to disk" & \begin{tabular}{l} 
Ø: The certificate is saved on the computer and it is available for future connec- \\
tions. \\
If you restart CODESYS, then the saved certificate is used.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "OK" & Authenticates and established the connection. \\
\hline
\end{tabular}

Dialog for authentication with a pass
phrase
\begin{tabular}{|l|l|}
\hline "Authentication area" & \begin{tabular}{l} 
Connection that is secured \\
Example: https://svn repository:443
\end{tabular} \\
\hline
\end{tabular}

Table 806: "A pass phrase is needed to unlock the certificate."
\begin{tabular}{|l|l|}
\hline "Pass phrase" & Example: \(* * *\) \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Save information to RAM" & \begin{tabular}{l}
\(\boxed{\checkmark}\) : The pass phrase is saved to the working memory. Then the client recognizes \\
in the current CODESYS session for future connections. \\
If you restart CODESYS, then you have to accept the certificate again.
\end{tabular} \\
\hline "Save to disk" & \begin{tabular}{l}
\(\square:\) The pass phrase is saved on the computer and it is available for future \\
connections. \\
If you restart CODESYS, then the saved certificate is used.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "OK" & \begin{tabular}{l} 
Authenticates with client certificates by means of a pass phrase and establishes \\
the connection.
\end{tabular} \\
\hline
\end{tabular}

Dialog for authentication with a user name and password
\begin{tabular}{|l|l|}
\hline "Authentication area" & \begin{tabular}{l} 
Connection that is secured \\
Example: https://svn repository:443
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "User name" & Example: \(a\). mayr \\
\hline "Password" & Example: \(* * *\) \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "Save information to RAM" & \begin{tabular}{l}
\(\checkmark\) \\
CODESYS session for future connections. \\
Con \\
If you restart CODESYS, then you have to accept the certificate again.
\end{tabular} \\
\hline "Save to disk" & \begin{tabular}{l}
\(\checkmark\) : Saved on the computer and it is available for future connections. \\
If you restart CODESYS, then the saved certificate is used.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline "OK" & Establishes the connection and authenticates it. \\
\hline
\end{tabular}

\section*{Dialog 'Automatic locking failed'}

The dialog shows a list of all objects for which an automatic locking was not possible. In the optoins you define how Professional Version Control will resolve the confilict.

Table 807: "Automatic Locking and Merging"
\begin{tabular}{|l|l|}
\hline - "Try to steal the lock for the \\
\begin{tabular}{l} 
affected objects" \\
"Activate the "Offline Mode" \\
to temporarily suppress \\
locking"
\end{tabular} & These options are displayed if another user has locked the object. \\
\hline - "Update the affected \\
\begin{tabular}{l} 
objects to the newest \\
revision" \\
"Update the whole project \\
to the newest revision" \\
"Activate the "Offline Mode"" \\
to temporarily suppress \\
locking"
\end{tabular} & \begin{tabular}{l} 
These options are displayed if there exists a more current version of the object \\
on the server.
\end{tabular} \\
\hline - "Activate the "Offline Mode"" & These options are displayed if no connection can be established to the server. \\
to temporarily suppress \\
locking"
\end{tabular}\(\quad\)\begin{tabular}{l} 
"SVN Project Settings" \\
\hline "SVN Settings"
\end{tabular} \begin{tabular}{l} 
Opens the SVN project settings dialog (menu "Project \(\rightarrow\) Project Settings"). \\
\hline
\end{tabular}

See also
- \(\&\) Chapter 6.6.7.6.3.1 "Dialog 'Options' - 'SVN Settings"' on page 4349
- \(\quad\) Chapter 6.6.7.6.3.2 "Dialog 'Project Settings' - 'SVN Settings'" on page 4350

\subsection*{6.6.7.6.4 Objects}
6.6.7.6.4.1 Object 'SVN_VERSION_INFO'.

4355

\section*{Object 'SVN_VERSION_INFO'}

Symbol: 䦗
The object contains the SVN metadata of the project as global constants or variables in a variable list. It is located in the "POUs" view. You can specifically retrieve the data of the global constants or variables by the application. By calling specific data, you can also reduce the memory usage on the controller.

The SVN metadata is provided for this purpose, subdivided over multiple global variable lists (GVLs):
- "SVN_VERSION_INFO"
- "SVN_Info_Summary"
- "SVN_Info_SummaryW"
- "SVN_Info_URI"
- "SVN_Info_Revisions"
- "SVN_Info_Flags"
- "SVN_info_LastChange"

The SVN_VERSION_INFO object is created automatically when a project is versioned by importing it into a SVN repository. To do this, the "Create SVN_VERSION_INFO" option has to be selected in the "Import project to SVN" dialog.
Furthermore you can create the object or remove it from the project with the option "Generate SVN_VERSION_INFO constants for IEC Access" (Dialog "Project \(\rightarrow\) Project Settings", category "SVN Settings").

Table 808: Global Constants
\begin{tabular}{|c|c|c|}
\hline Name & Data type & Description \\
\hline \begin{tabular}{l}
MINREVISIO \\
N
\end{tabular} & LINT & Lowest revision number of the working copy \\
\hline \begin{tabular}{l}
MAXREVISIO \\
N
\end{tabular} & LINT & Highest revision number of the working copy \\
\hline PARTIAL & BOOL & \begin{tabular}{l}
TRUE: The working copy is incomplete. \\
Example: Cancellation during the last update due to a network error or a checkout.
\end{tabular} \\
\hline MODIFIED & BOOL & TRUE: Local changes were made. \\
\hline SWITCHED & BOOL & TRUE: Parts of the project were branched (with the "Switch" command). \\
\hline VERSION & STRING & \begin{tabular}{l}
Version identification, similar to Apache \({ }^{\text {TM }}\) Subversion \({ }^{\circledR}\) (subversion.exe) \\
Example: 12:34M, means MINREVISION = 12, MAXREVISION = 34, MODIFIED = TRUE \\
For more information, refer to the documentation for Apache \({ }^{\text {TM }}\) Subversion \({ }^{\circledR}\).
\end{tabular} \\
\hline CLEAN & BOOL & \begin{tabular}{l}
TRUE: The version is clean. \\
This is the case when MINREVISION is equal to MAXREVISION, the working copy is complete, and non-versioned, and is was not switched.
\end{tabular} \\
\hline URL & WSTRING & \begin{tabular}{l}
SVN-URL of the project \\
Example: https://svnserver/repository/trunk/ \\
ControlABC.project
\end{tabular} \\
\hline
\end{tabular}

If a controller does not support the data type WSTRING, then a compiler error is issued when accessing the object SVN_VERSION_INFO.

\section*{See also}
- \({ }^{\sharp}\) Chapter 6.6.7.6.3.2 "Dialog 'Project Settings' - 'SVN Settings'" on page 4350
- \({ }^{4}\), Chapter 6.6.7.6.2.4 "Command 'Import project to SVN'" on page 4326

\subsection*{6.6.8 Subversion}

\subsection*{6.6.8.1 Project Version Control with Subversion}

\subsection*{6.6.8.1.1 Introduction}

Automation Builder projects can be stored in Subversion (SVN) repositories by using the Project Version Control. The Project Version Control can be used to track changes on a project and to have access to historic versions of the whole project or objects in the project. It is possible to hold different versions of a project in branches and to compare these versions. The Project Version Control enables multiple engineers to work collaboratively on the same project.

\subsection*{6.6.8.1.2 Preconditions}

Basic knowl- Make yourself familiar with the concepts of SVN.
This manual about Project Version Control is additionally to the following information and describes mainly the specific behavior of Subversion in Automation Builder.
- Homepage of Subversion: http://subversion.apache.org/
- Online user manual for Subversion: http://svnbook.red-bean.com/
- Documentation on SVN integration in Automation Builder: Refer to subfolder .
\begin{tabular}{ll} 
Automation & - In Automation Builder, the Project Version Control must be installed. \\
Builder & - A valid license for the Project Version Control must be activated. \\
& - All collaborating users working on the same project need: \\
& - Automation Builder installed in the same version with the same features. \\
& - Sicense for same edition. \\
& - Same set of optional third party device descriptions. \\
&
\end{tabular}

SVN server The Project Version Control can be used in combination with an SVN server in version 1.6 or newer, the repository format should be 1.5, 1.6 or 1.7. Newer repository formats are not yet supported.
The usage of local repositories in the local file system or even on a network share is strongly discouraged.

\subsection*{6.6.8.1.3 Working with Project Version Control}
- All objects in the device tree or POU tree are represented by an object in the SVN repository, there might be hidden objects that are not visible in the tree but that exist in SVN.
- The smallest unit in the SVN repository is one object including all its data like name, parameters, device identification.
- Objects are identified in the SVN repository by their name. Renaming one object in Automation Builder means to delete it from the SVN repository and add a new one to the SVN repository. Renaming an object causes a break in the history of that object.
- By default objects are locked before they are changed to prevent other users from changing the object. The locking strategy can be changed in the user options.
- Objects can be compared to other versions of the same object, many differences/changes between the current object in the Automation Builder project and the compared object can be merged into the object in the Automation Builder project. Merging changes could be used to resolve conflicts in case concurrent changed can not be avoided.
- To ensure consistency it is required and also enforced that some changes can be committed or reverted only together.
- All changes to device objects in the hardware tree that are sub-nodes to the same top level device. Note: Objects that are not devices are excluded, e.g. the application node.
- All changes below the AC500 PLC application node.
- Most SVN operations can not be performed while other external applications like CODESYS or Panel Builder work on files that are embedded in Automation Builder project.
- Some operations like changing the target or updating the project to the latest device (description) versions do a recursive lock of the whole AC500 PLC. If the lock can't be obtained the operation is aborted.
- Some objects contain internal data that has no meaning to the end user but is also important. Changes on such data are not shown in the compare dialog or are summarized by a placeholder like "There are hidden changes".
- Including externals is not supported.

\subsection*{6.6.8.1.4 Recommendations on Working with Project Version Control}

Be collaborative - Multiple users that work collaboratively on the same project should agree on their responsibility for certain parts of the project where they do changes to avoid conflicts and tree conflicts.
- Agree on locking strategy used by all users working on the same project.
- Distribute the work between multiple users meaningful.
- It is suggested to setup the hardware structure at first before other users checkout a project to work on it and limit structure changes in the hardware tree to the minimum.
- Before adding objects, especially top level objects, users should agree that only one user adds objects at top level or below the same parent, or agree on unique names for the objects to add. The default naming scheme for new objects bears the risk of name conflicts. These conflicts could be resolved only by reverting the changes of the user who later tried to commit the changes.

Be careful - The SVN integration (and also project compare) gives lot of power to the user, users should be sure to do only things they fully understand. Especially by merging changes incomplete it is possible to create inconsistent data.
- Adding devices, removing devices or even changing parameters can have side effects to other devices, do not change objects/parameters to their original state by merging that were not done explicitly.
- Commit changes frequently to SVN.
- To release locks that you don't longer need.
- To reduce the risk of conflict with co-workers.
- To keep the sets of changes to commit small.
- Do frequent updates when collaborating in a team.
- To be up-to date.
- To keep the sets of changes to get from SVN small.
- To reduce the risk of losing work results in case of conflicts.
- To avoid conflicts, it is suggested to stay with the default setting to automatically lock objects before doing changes. Consider explicit recursive locks of sub-trees where you plan bigger changes.
- Prefer a clean checkout over using the switch command to change between different branches.
- Do not use the switch command to change between unrelated projects, this could corrupt the Automation Builder project (local copy, not in SVN) easily.
- Commit local changes to the SVN repository before creating a branch.

Be effective - Give objects good/correct names after adding them and use renaming of objects already committed to SVN sparely to maintain a continuous history in the SVN repository.
- The goal to revert only single changes of all changes done that must be committed/reverted together, could be achieved by using project compare or the object compare dialog.
- If changes can't be committed to the SVN repository because of locks hold by other users, it is possible to create a branch, use the switch command to change to this branch and commit the changes there. The branch and base line could be merged together later.

\subsection*{6.6.8.1.5 Known Issues and Troubleshooting}

Not all changes are shown for all objects, but hidden changes are also important.
The device pool may be changed as side effect of several operations, including opening the project.
When a project was corrupted (by performing an update that tried to add an AC500 communication module) it is possible to save this project and merge changes to a project that has been cleanly checked out by project compare.

\subsection*{6.6.8.2 SVN Support Examples}

\subsection*{6.6.8.2.1 Importing Automation Builder Project to SVN Repository}
1. In the Automation Builder main menu, go to "Project \(\rightarrow\) SVN
\(\rightarrow\) Import Project to Subversion".
2. Enter user credentials and click "OK".
3. Select SVN server repository to import Automation Builder project and click "OK".
\(\Rightarrow\) The Automation Builder project is imported into the selected repository and connected automatically to the repository. The imported project nodes are identified with green indicators.


\subsection*{6.6.8.2.2 Logging in User2}
1. In the Automation Builder main menu, go to "Project \(\rightarrow\) SVN \(\rightarrow\) Checkout".
2. Enter user credentials and click OK.
3. Select the repository location, project folder and revision if any and click "OK".
\(\Rightarrow\) The project will checked out of the repository, saved in the selected location and opened as a primary project.

The tables below provide the descriptions of the options availlable in the check-out dialog.

\section*{Checkout options}
\begin{tabular}{|l|l|}
\hline Omit externals & Do not checkout external objects. \\
\hline As library project & Saves the project as a CODESYS library file. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline Revisions & Checks out the Head revision. \\
\hline HEAD & \begin{tabular}{l} 
Allows to select the required revision of the \\
project.
\end{tabular} \\
\hline Revision & Allows to select a revision date of the project. \\
\hline Date
\end{tabular}

The following instances can occur in the workflow.
- If the project contains any updates, the specific project level node is indicated with
- If a new object is added to the project, the newly added node is indicated with
- If the project node is deleted, the specific node is indicated with \(\mathbf{x}\).

\subsection*{6.6.8.2.3 Examples}

Example 1 If User1 modifies Panel_CP600 project, then the node indicator turns to orange with lock symbols. If User2 need to modify the same Panel_CP600 project, the Panel_CP600 project appears with a lock symbol.
\begin{tabular}{|c|c|}
\hline Devices \(\quad \bigcirc \quad \square \times\) & Devices \(\quad\) - \(\quad \times\) \\
\hline  &  \\
\hline
\end{tabular}

To steal the lock of an affected object, proceed as follows:
1. Double-click "Panel_CP600" project.
\(\Rightarrow\) Automatic lock failed dialog is displayed.
2. Enable "Try to steal the lock for the affected objects" and click "OK" to steal the lock.
3. In the Lock Message window, enter the reason to steal the lock and click "OK".
4. User2 can modify and commit the project.

Example 2 If User1 adds a new object to the project and commit the changes, then User2 can update the project to see the latest modifications.
```

Devices * п >

# 人, -1 Project1

    Panel_CP600_1 (CP600 Control Panel)
    目f Panel Project
    PLC_AC500 (PM564-ETH)
    -- 周纤 Application
App
OBIO (6DI+6DO}+2\textrm{AI}+1\textrm{AO}
IO_Bus
*) Interfaces
El] COM1_Online_Access (COM1 - Online Access)
COM2_None (COM2 - None)
Ethernet
ETH1 (ETH1)
Protocols (Protocols)
zenon_Project

```

Example 3 The user can revert to any of the available project revisions．
1．Right－click on object node and select＂SVN \(\rightarrow\) Revert to Revision＂．
2．Select or enter the revision number and click＂OK＂．
\(\Rightarrow\) The revision command reverts local changes of this object back to the specific revision of the working copy．

3．Right－click on the object node and select＂SVN \(\rightarrow\) Commit＂．
4．In the commit window，enter the reason to change the project and click＂OK＂to make the changes．
\(\Rightarrow\) The project node is updated with the latest changes．

Example 4 SVN server allows to select the required revisions of Automation Builder project．You can checkout the project using＂Project \(\rightarrow\) SVN \(\rightarrow\) Checkout＂and then enter the credentials and click＂OK＂．
In the check－out dialog，do the following：
1．Select the project repository．
2．Activate＂Revision＂and select or enter the revision number and click＂OK＂．

The user can work on the selected revision．To commit the changes to the project，right－click on project and select＂SVN \(\rightarrow\) Commit Project＂．

\section*{6．6．9 Mail service with SMTP／SMTPS}


With the integration of the＂Mail Service SL＂library，the prerequisite for sending mails with the PLC has been created．After appropriate programming of the user requirements and the provision of appropriate hardware and software in the periphery（router，mail program）this service is available．

Mail Service SL This library can send, receive and delete emails. An SMTP server is required for sending, and library

Simple Mail
Transfer Protocol (SMTP)

SMTP is a communication protocol for the exchange of data (e-mails) in networks or in the internet. SMTP is primarily used for sending and forwarding e-mails. Other, specialized protocols (POP3/IMAP) are used to fetch e-mails.

\section*{Simple Mail \\ Transfer Protocol Secure \\ (SMTPS)} SMTPS is an independent further development of the SMTProtocol and authenticates the communication of the e-mail partners by additional encryption protocols (SSL/TSL). This happens before the message is even sent.

Application
example
How to send e-mails via the SMTP protocol and how to create the technical requirements for this is described in detail in the application example. AC500 V3 - Sending e-mails via the SMTP protocol

\subsection*{6.6.10 SNMP}

Simple Network Management Protocol (SNMP)

SNMP is a network protocol to monitor and control routers, servers, switches, printers, computers etc. from a central station. The protocol controls the communication between the devices and the monitoring station.

SNMP Service Use the SNMP Service SL library to read out information from devices via the SNMP protocol. SL Library The library provides function blocks for sending and receiving SNMP messages and is part of the software package "IloT Libraries SL".

In the integrated V3 library documentation (within the library manager) you find further information on a respective library as well as on the functionality and application area of the contained function blocks.

\begin{abstract}
Application example

The application example AC500 V3 - SNMP Protocol describes the SNMP communication in a scenario in which an AC500 PLC acts on the one hand as SNMP agent and on the other hand as SNMP manager. In this case, the SNMP protocol is used for network monitoring. The AC500 PLC can receive SNMP requests and respond to them as an agent. As a manager, it can query managed information from any network device that supports SNMP.
\end{abstract}

\subsection*{6.6.11 Scripting with Python}

\subsection*{6.6.11.1 Python}

\subsection*{6.6.11.1.1 Python script support}

Using scripts Scripting allows python scripts to be used to automate project configuration in Automation Builder. Parameters can be added to scripts, so that a generic script can be customized before execution. The user can add a script to most parts of the device tree. A script can be started either from the user interface (by a command or with the python scripting editor) or from the Windows command line and is saved with the project.

With the scripting feature commands or complex program operations can be automated.
Examples of use cases:
- Integration of Automation Builder in automatic build server environments:
- continuous integration (CI)
- continuous delivery (CD)
- continuous testing
- Integration with third-party software, for example:
- code generators
- creation of projects that are custom tailored to a specific machine configuration
- Creation of documentation
- Updating of libraries: Setting of project information during the release process
- Automatic testing: Mostly in connection with the Professional Test Manager
- Outputting variables via monitoring APIs

Licensing A valid license is required to use the scripting. If you open a project with the existing script object without a valid license, you are not allowed to add or edit the scripts. However, the scripts are not removed from the project.

> Scripting language

> The Automation Builder scripting language is modular and based on IronPython. For this purpose, the Automation Builder "ScriptEngine" component combines the IronPython interpreter with the Automation Builder development environment which makes the extensive python framework libraries available including file access in networks and much more.

\subsection*{6.6.11.1.2 Working with script objects}

Scripts to execute can be added to and stored in the Automation Builder project. Additionally, parameters can be added to scripts, so that generic scripts can be customized before execution.

\section*{Adding a script object to the project}
1. In the device tree, right-click on a node (e.g. a PLC node) and click "Add object".
2. Under "Scripting category" select "Script \(\rightarrow\) Add object".
\(\Rightarrow\) The 'Add Script' dialog is displayed.
3. Browse and select a script from the file system or create a new script by clicking [Add]. \(\Rightarrow\) A script is added below the selected node and the editor is opened.

4. The default parameter values are read from the script. The user can edit the default values as required.


The script objects can be reused within the project via copy-and-paste around the device tree.

\section*{Execution The user can execute the script with the parameter values via the execute button in the editor or} via right-click on the script object in the device tree by selecting "Script \(\rightarrow\) Execute".

Import The user can import the script from the file system. This will replace the contents of the current script object with the contents of the imported file. Optionally, parameter values will be preserved if the imported script has a matching named parameter. In the device tree, right-click on a script object and select "Script \(\rightarrow\) Import".

Export The user can export the selected script and saved it as a new file in the file system. The exported file does not include any edited parameter values. In the device tree, right-click on a script object and select "Script \(\rightarrow\) Export".

Parameters The following instructions help the user to create parameters in the python script:
- Parameters must be defined in the script.
- Parameters and values are optional.
- The ParameterName and the ParameterValue must be delimited with symbols. The format must be as follows:
"\#AutomationBuilder_Parameter \{"ParameterName"\} \{= "ParameterValue"\}
- \{ParameterName\} is the name given to the parameter. This allows the values to be referenced in the python script.
- \{ParameterValue\} is the default value given to the parameter. This value can be modified in the editor.
The example below shows the format of the ParameterName and ParameterValue in the script.
- \#AutomationBuilder_Parameter "numWidgets": creates a new parameter called numWidgets.
- \#AutomationBuilder_Parameter "numWidgets" = "4": creates a new parameter called numWidgets and initializes to the value 4.
Using parameters within the python script:
- Parameters can be used in the script by creating an instance of the parameter helper: parameterHelper = AutomationBuilder_Parameters.create()
- Individual parameters are retrieved by calling:

GetParameter(parameterName). devicename = parameterHelper.GetParameter("Name")

Python script A set of python script examples are available in the path \%Public\%\Documents\Automationexamples BuilderlExamples\Python scripts.

\subsection*{6.6.11.1.3 Execute Script}

You can execute Python script files (<file name>.py), which contain a sequence of commands for activating CODESYS functionalities, in the following way:
- In the CODESYS user interface by means of commands in the "Tools \(\rightarrow\) Scripting" menu
- From the CODESYS user interface by means of a customized, configured toolbar
- From the Windows command-line

\section*{Calling Scripts from Menu Commands}

Requirement: A valid Python script file <file name>. py is located in the file system. The CODESYS user interface is open.
1. (optional) To monitor the processing of individual commands used in the script, click "Tools \(\rightarrow\) Scripting \(\rightarrow\) Enable Script Tracing".
2. In CODESYS, click "Tools \(\rightarrow\) Scripting \(\rightarrow\) Execute Script File".
\(\Rightarrow\) The statements in the script are executed and, if script tracing is activated, listed in the message view.

\section*{Starting Scripts from the Command Line}

In automated environments such as Cl (Continuous Integration) servers or if scripts in CODESYS have to be controlled by other programs, menu commands are not appropriate for executing scripts. For these kinds of requirements, you can use the Windows command line to start CODESYS and execute scripts.

Requirement: A valid Python script file <file name>. py is located in the file system.
1. Create a cmd file using start to start CODESYS and with the option --runscript executes the script file. Other options are possible, for example --noUI, if the CODESYS- User interface should not be opened.
2. Open the Windows "Command prompt" and execute the CMD file.

You can pass arguments with additional information to the script. Python scripts can access arguments with the sys.argv [] list. The first element (Index 0) is always the name or path of the Python script that is executed, followed by the "actual" parameters. (This is similar to argclargv in C.) In addition, scripts can also access environment variables that are set before CODESYS is started with the corresponding Python or .NET APIs.
```

Example A CMD batch file argvtestbat.cmd has the following contents (all in one line).

```
```

"C:\Program Files (x86)\CODESYS

```
"C:\Program Files (x86)\CODESYS
3.5.17.0\CODESYS\Common\CODESYS.exe" --profile="CODESYS V3.5
3.5.17.0\CODESYS\Common\CODESYS.exe" --profile="CODESYS V3.5
SP17" --runscript="D:\Dokumente\Scripting\ArgvTestScript.py" --
SP17" --runscript="D:\Dokumente\Scripting\ArgvTestScript.py" --
scriptargs:'username password 3.14 "path=\"C:\temp\\\""' --noUI
```

scriptargs:'username password 3.14 "path=\"C:\temp<br>\""' --noUI

```

You have a matching script file ArgvTestScript. py.
```

from __future__ import print_function
import sys
print("sys.argv: ",
len(sys.argv),
" elements:")
for arg in sys.argv:
print(" - ", arg)

```

Now when you execute the CMD file, CODESYS starts and executes the script without opening the CODESYS main window. Then CODESYS is exited:


For a complete reference of all possible command line parameters, see the help page for the command-line interface in CODESYS in the section for "--runscript".
For information about the Python API, see:
https://docs.python.org/2/library/os.html\#process-parameters
For information about the .NET-API, see:
https://msdn.microsoft.com/de-de/library/77zkk0b6\%28v=vs.110\%29.aspx
- See also: \({ }^{\mu}\) Chapter 6.4.1.16 "Using the Command-Line Interface" on page 2028

\section*{Calling Scripts from Toolbar Icons}

You can provide your own toolbar in the CODESYS user interface with up to 32 icons for calling script files. For this you need an ICO file where the icon is stored, and a PY file where the Python script to be called is stored.
In the installation directory or in the program files directory under CODESYS, you create a subdirectory named Script Commands. In that directory, you create the config.json configuration file. Specify the call information outline for each icon in the file. Here you can configure a maximum of 16 icons. You can also store the ICO and PY files in the same directory.

\section*{Location}
- <CODESYS installation directory> \CODESYS \(\backslash\) Script Commands
- Users\<username>\AppData\Local\CODESYS\Script Commands

\section*{Example Default installation on Windows 10}
\[
\begin{aligned}
& C: \backslash \text { Program Files (x86) \CODESYS 3.5.14.0\CODESYS } \backslash \text { Script Commands } \\
& \text { C:\Users } \backslash r . \text { smith } \backslash \text { AppData\Local } \operatorname{CODESYS\backslash Script~Commands~}
\end{aligned}
\]

If you store a config. json file with different call information at each of the storage locations, then you can configure up to 32 different icons.

\section*{Configuration file}

Outline of the configuration file for 2 icons
```

[

```
    <icon call information>
    \},
    \{
        <last icon call information>
    \}
]

Outline of the call information <icon call information>
"Name": "<tooltip of the symbol button>",
"Desc": "<description of the symbol button>",
"Icon": "<icon file name>",
"Path": "<path of the script file>"

Table 809: Call information
\begin{tabular}{|l|l|}
\hline "Name" & \begin{tabular}{l} 
Required \\
Displayed as symbol tooltip \\
Example: "Name": "Pause"
\end{tabular} \\
\hline "Desc" & \begin{tabular}{l} 
Optional \\
Comment for the icon \\
Example: "Desc": "Operation pause" \\
Note: Not yet displayed in the user interface
\end{tabular} \\
\hline "Icon" & \begin{tabular}{l} 
Required \\
File path: <directory path>\<icon name>. ico of the icon \\
Example: "Icon": "pause.ico" \\
Hint: If the file is in the same folder as the config.json file, then the file name is \\
enough.
\end{tabular} \\
\hline "Path" & \begin{tabular}{l} 
Required \\
Path of the Python script <directory path>\<script name>. ico \\
Example: "Path": "stop.py" \\
Hint: If the file is in the same folder as the config.json file, then the file name is \\
enough.
\end{tabular} \\
\hline "Params" & Note: These parameters are not currently assessed. \\
\hline
\end{tabular}
```

Example File config.json
[
"Name": "Start",
"Desc": "Starts processing",
"Icon": "start.ico",
"Path": "goon.py"
},
{
"Name": "Pause",
"Desc": "Pause operation",
"Icon": "pause.ico",
"Path": "stop.py"
},
{
"Name": "Processing",
"Desc": "Process again",
"Icon": "VarStatSmall.ico",
"Path": "process.py"
}
]
The following files are located in the Script Commands:

```
```

config.json

```
config.json
goon.py
goon.py
stop.py
stop.py
process.py
process.py
start.ico
start.ico
pause.ico
pause.ico
VarStatSmall.ico
```

VarStatSmall.ico

```

\section*{Creating script calls for a toolbar button}
1. Create the Script Commands folder in one of the storage locations.
```

C:\Users\<username>\AppData\Local\CODESYS

```
2. Create executable Python files there.
\(\Rightarrow\) Example:
File copy.py
        print("The script COPY.PY is executed")
        File delete.py
        print("The script DELETE.PY is executed")
3. Create the ICO files for the scripts.
\(\Rightarrow\) Example: Copy_before.ico, Copy_below.ico, CopyAll.ico
4. Create a configuration file config.json there.
\(\Rightarrow\) The folder Users \<username>\AppData\Local\CODESYS \(\backslash\) Script Commands has the following content:

\section*{Name}

5. Open config.json and add the outlined call information.
\(\Rightarrow\) [
\{
        "Name": "Copy Before",
            "Desc": "Copy something",
            "Icon": "Copy_before.ico",
            "Path": "copy.py
    \},
    \{
        "Name": "Copy Below",
        "Desc": "Copy something",
        "Icon": "Copy_below.ico",
        "Path": "copy.py"
    \},
    \{
        "Name": "Copy All",
        "Desc": "Copy something",
        "Icon": "CopyAll.ico",
        "Path": "copy.py"
    \},
    \{
        "Name": "Delete",
        "Desc": "Delete something",
        "Icon": "Delete.ico",
        "Path": "delete.py"
    \}
    ]
6. Start CODESYS.
\(\Rightarrow\) The script files, configuration file, and symbol files are read and provided in the "Tools \(\rightarrow\) Customize" dialog on the "Command Icons" tab, in "ScriptEngine Commands" category.
7. Open the dialog "Tools \(\rightarrow\) Customize" and click on the tab .
8. Select the empty toolbar there and click the "Add Toolbar" button.
\(\Rightarrow\) A line editor opens at the empty toolbar.
9. Type in a name (example: User defined toolbar).
\(\Rightarrow\) The custom toolbar is displayed in the CODESYS window.
10. Add the recently imported commands and close the dialog.

11. Click one of the icons.
\(\Rightarrow\) The following output is displayed in the message view.

\section*{Script Messages}

\section*{Description}

The script COPY.PY is executed

\subsection*{6.6.11.1.4 Python script editor}

In Automation Builder a browser-based python script editor is integrated. This allows the user to modify the existing python script, to create a python script from the scratch and to finally execute the script. Moreover, it assists the user in writing the script with the following features:
- Auto suggest
- IntelliSense suggestions for the python syntax during typing.
- IntelliSense for CODESYS script engine and Automation Builder injected script objects.
- Built-in language service that provides complete code intelligence for objects, properties and methods.
- Details of the object with [CTRL] + [spacebar].
- Auto completion

Press the Enter key on a function suggested by IntelliSense in order to insert it.
- Python syntax highlighting (basic syntax colorization)

The function and its respective namespace is automatically colored in order to match colors.
- Matching brackets

Matching brackets are highlighted as soon as the cursor is near to one of them using the command palette.
- Zoom

Changes the font size of the editor's content.
- Find and replace

Support of 'Find' (search for a keyword) and 'Find and replace' (search and replace a keyword). This feature is supported in the editor, however not integrated in Automation Builder platform.
- Minimap

High level overview of the script for a quick navigation and code understanding.
- Copy/paste

Support of 'copy and paste' of the script text within and into the editor.
- Undo/redo

Support of 'undo/redo' for editing actions. This feature is supported in the editor, however not integrated in Automation Builder platform.
- Keyboard shortcuts

Keyboard shortcuts allow to perform most tasks directly from the keyboard (e.g. [CTRL]
\(+[Z],[C T R L]+[Y])\) including copy and paste. For further keyboard shortcuts refer to the command palette ([F1]).
- Folding

Support of folding and expanding script regions.
- Comment/uncomment the code

Support of commenting ([CTRL] + [K]) and uncommenting ([CTRL] \(+[C]\) ) code through shortcuts.
- 'Execution' button Executes the script directly in the editor window.
- In order to start a new script from the scratch the user can start with an empty editor. This can be done via the 'Add script' dialog without script file selection.
For further features that can be used in the python script editor refer to the command palette ([F1]).

Limitations with CODESYS script engine IntelliSense
- No IntelliSense available for return type of a property.
- No support of IntelliSense for keyword "None".
- No IntelliSense support for method overloading.
- No IntelliSense support for methods that return an object.
- Private methods are also part of IntelliSense. Refer to the CODESYS script engine document to verify the access modifier.

\subsection*{6.6.11.1.5 Configure port number for python editor server}

The python editor server is required for the usage of python script editor.
Earlier versions of Automation Builder than 2.6.0 do not have an option to change the port number of Python editor server via user interface. The user had to update the port number in Automation Builder configuration file if the port was already in use.

With Automation Builder 2.6 onwards, the user has an option to install multiple versions of Automation Builder on a system with separate Python editor servers. Since multiple servers cannot run on a single port, users are given an option to configure the port number from user interface ("Tools \(\rightarrow\) Options \(\rightarrow\) Python Scripting Editor") if the port number are in use.
Default port number of Python editor server is 11030 for Automation Builder versions before 2.6 and 11040 for Automation Builder 2.6 onwards.

\subsection*{6.6.11.1.6 Creating a Python Script}

Python is a dynamic language. You can start in a simple linear programming style ("batch files") and later add the necessary and more powerful means, such as conditions, loops, functions, exceptions, classes, and modules. The focus of the language is on easy and expressive code. Python is more typical at runtime and uses an automatic garbage collector to protect the programmer from accidental damage to the entire system.

IronPython is an implementation of Python for .NET and allows for full access to the .NET framework and classes. The implementation of the IronPython interpreter is based on Python Version 2.7.

There are a variety of free manuals and help pages on the Internet. See the following links for an introduction and detailed introduction about IronPython.
- http://forum.codesys.com/viewforum.php: "Script language Python..." area in the CODESYS forum
- Especially for CODESYS-specific questions
- Also includes some examples
- https://docs.python.org/2/tutorial/index.html:: Python tutorial in the official Python documentation
- http://docs.python.org/release/2.7/: Official documentation for Python 2.7
- http://wiki.python.org/moin/BeginnersGuide: Useful manuals for learning IronPython
- http://wiki.python.org/moin/GermanLanguage: Collection of links for German help pages
- http://stackoverflow.com/: General community for programming For general question about (Iron)Python, not CODESYS-specific
- http://ironpython.net/: IronPython homepage
- http://ironpython.net/support/: Mailing list, FAQ, etc.
- https://gitter.im/IronLanguages/ironpython:: Chat channel for IronPython developers


Version incompatibility to Python V3.x
The Python programming language will soon be available in the new version V3.x. Some of the older program modules have been removed. CODESYS GmbH is planning an update to this new version. Scripting developers should take this into consideration and design their scripts accordingly, for example by using the expression from
future import print function. You can find more informa\(\overline{\text { tion }}\) about this topic at http://wiki.python.org/moin/Python2orPython3 and http://docs.python.org/release/3.1.2/whatsnew/3.0.html

\section*{Getting Started with Python for CODESYS}

See below for an simple application of a Python script in CODESYS:
1. In any text editor, create a text file hello.py with the following contents:
print("Hello, automation!")
2. Start CODESYS and click "Tools \(\rightarrow\) Scripting \(\rightarrow\) Execute Script File". Select the file hello.py in the file system.
\(\Rightarrow\) See the result in the message view:


For more detailed examples of Python scripts for different use cases with CODESYS, see the following help pages.

\section*{Tips for Python Programmers about .NET API Documentation}

The current prerelease of the script interface documentation has been generated automatically from the underlying .NET and C\# sources. Therefore, the documentation includes some expressions that are not familiar to Python programmers. The following overview provides some tips about how these expressions can be understood from the Python perspective.
- An interface is the contract that tells an instance of a class that implements the interface which members (methods, properties) it has to prepare. In IronPython, you can implement one or more .NET interfaces in one class by inheriting from a superclass. If a method is needed for the interface but is not available in the class definition, then an exception is thrown. (The DeviceImportFromSvn. py example shows a class that implements the ImportReporter interface.)
- Each parameter and each method in .NET is strictly typed. The type of parameter is separated by one space character before the parameter name, and the type of return value from a method by one space character before the method name. You can use instances from subclasses when you define a class (or interface). A method without a return value is marked void.
- You can overload methods. As a result, multiple methods with the same name can exist in one class. However, the number or the types of parameters have to be different. IronPython automatically takes care of the most appropriate method overloading being called.
- The data type int corresponds to an integer from -2,147,483,648 to 2,147,483,647. The data type bool corresponds to the Python type bool (True and False).
The data type string corresponds to the Python types str or unicode, which are identical in IronPython.
The IDictionary<object, object> data type corresponds to an ordinary Python dictionary. IronPython automatically converts between Python and .NET data types.
- When a T type is inherited from IBaseObject<T>, it means that this type can be extended by other plug-ins for additional members. The actual use of this extended type as a parameter or return value is marked by IExtendedObject<T>.
- The IEnumerable<T> interface to a \(T\) type means that you can use every Python sequence (generators, lists, tuples, etc.) which returns T type values (or a subclass). Returns the sequence of incompatible objects, and throws an exception at runtime.
- The IList<T> interface to a T type identifies a typified list which guarantees to include only elements of type \(T\) (or a subclass). An exception is thrown at runtime when any attempt is made to add an incompatible object.
- The params T[] name id for a parameter of type T corresponds to the Python mechanism * name for variable argument lists.
- In Python, enumerations (enum) do not exist as language constructs. Its purpose is to define an exact number of constant values for a specific purpose, for example the days of the week. Access to .NET enumerations from IronPython works by using "Name.Member", for example via OnlineChangeOption.Try.
There are also different approaches to emulate enums in Python.
For example, see: http://pypi.python.org/pypi/enum/
- The syntax \(T\) name \{ get; set; \} defines a property with name as the name and \(T\) as the type. If set; is missing, then the property is read-only. In Python, the respective construct is @property decorator.

\section*{Basic Syntax of Python (with Examples)}

Python is similar to the languages of the "C family", but there are some significant differences and unique properties.

The most obvious syntactic difference between Python and language such as C and ST is that the Python parser recognizes block structures by their indentation. There is no BEGIN/END or braces \{\} to identify the blocks of IF/ELSE conditions, FOR and WHILE loops, or functions.

Comments start with \# and extend to the end of the line. In the first and second line of the source code, you can set a special marker to declare the encoding of the file. We recommend that you use UTF-8 as the encoding if ASCII characters are not required.

For debugging purposes, you use print for easy output. With the \% operator, you achieve functionality similar to the \(C\) function printf(). The output is displayed in the message view of CODESYS.
```

Example:
print

```
```


# encoding:utf-8

```
# encoding:utf-8
# defining a function with the parameter i
# defining a function with the parameter i
def do something(i):
def do something(i):
        # if branch
        # if branch
        if i>0:
        if i>0:
            print("The value is: %i" % i)
            print("The value is: %i" % i)
+= i
+= i
            print("The new sum is: %i" % sum)
            print("The new sum is: %i" % sum)
        # else if (optional, there can be none or several elif branches)
        # else if (optional, there can be none or several elif branches)
        elif i=0:
        elif i=0:
            print("The sum did not change: %i" % sum)
            print("The sum did not change: %i" % sum)
        # and the final else branch (also optional).
        # and the final else branch (also optional).
        else:
        else:
            handle_error()
            handle_error()
# an endless while loop
# an endless while loop
while True:
while True:
    print("I got stuck forever!")
```

    print("I got stuck forever!")
    ```

Everything that belongs to the same block has to be indented the same distance. The size of the indentation is irrelevant. Elements such as brackets and braces have a higher priority than indentations. Therefore, the following code segment is completely correct, even if it is written in a "poor programming style":

\section*{Example:}

\section*{Indentation}
```


# warning: bad style below. Kids, don't try this at home!

if foo >= bar:
print("foobar")
else:
print(
"barfoo"
)

```

To avoid ambiguity, you should not mix tabs and spaces in a file.
```

                    At this time, mixing tabs and spaces gilt in Python 3 qualifies as a syntax error.
                    The official Python Style Guide recommends indentation of four spaces and
                    includes some examples of good and poor style. The Python tutorial provides a
            summary of coding style.
    Python is "case-sensitive", similar to C and in contrast to ST. Keywords, such as def, if, else, and while, have to be lowercase (in contrast to the ST rule: keywords are uppercase). Two identifiers, such as "i" and "I", also identify two different variables.
The following keywords are reserved in Python and not permitted for use as identifiers for variables, functions, etc.: and | as | assert | break | class | continue | def | del | elif | else | except | exec | finally | for | from | global | if | import | in | is | lambda | not | or | pass | print | raise | return | try | while | with | yield.

```

Python 3 defined four other keywords: False | None | True | nonlocal. While the first three are really new, the first three were already predefined constants in Python 2 and should not be used for any other purposes.

For more information, see: Python Style Guide and Python Tutorial

\section*{Variables and data types}

Python is a powerful, dynamically typed language -- all type information is evaluated at runtime. Variables hold references to objects, and the object knows its type, not the variable. When a programmer attempts to execute an operation that is not possible (for example, adding an integer and a string), Python throws an exception at runtime.

Consequently, there are no declarations of variables and their types. In Python, variables are created only to assign values to them. This is completely different in C and ST where types are strong and static. Every variable is declared with a type, and at compile time the compiler checks that the type and operators are permitted.
See the following examples for handling variables:

\section*{Example: Vari- \\ ables}
```


# assign the integer 1 to the variable i (also "creates" the

variable")
i = 1

# assign the string "foobar" to the variable s

s = "foobar"

# Add 5 to the integer i - this is equivalent to i = i + 5

i += 5

# i now holds the integer 6.

# Try to add i and s - this will throw an exception when executed

# TypeError: unsupported operand type(s) for +: 'int' and 'str'

result = i + s

# variables can also be "undeclared" by deleting them.

# Further access to the variable i will throw a NameError exception,

# as the variable does not exist any more.

del i
i += 5 \# now throws an exception: NameError: name 'i' is not defined

```

All existing variables reference one value only. There is not any unassigned or uninitialized variables in Python. To express the absence of a value, Python provides a special object: None. In C or ST, you would use a null pointer. Its only purpose is to express "no value here", although None is actually an existing instance of the class NoneType.

\section*{Numeric types and floating points}

In contrast to the dozens of integer types in IEC or C, there is only one integer type in Python. Integer types in Python do not have a fixed size. Instead, they grow as needed and are limited only by available memory.
```

Example:
Integers.py
from __future__ import print_function
i = 1
print(i)
j = 0x1234 \# hex number, is 16\#1234 in IEC and 4660 in decimal
k = 0o123 \# octal number, is 8\#123 in IEC and 83 decimal
l = 0b101010 \# binary number, is 2\#101010 in IEC and 42 in decimal
print(j, k, l)
m = (2 + 3)*10 \# k is 50 now
print(m)
n = 10 ** 100 \# 10 to the power of 100
print(n)
Resulting output:

```
```

Messages - Total 0 error(s), 0 warning(s), 0 message(s)
Script Messages * \$0 error(s) © 0 warning(s) ©0 message(s)}
Description
1
46608342
50
10000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

```

There is also only one floating-point type in Python which is similar to the IEC data type LREAL. It provides 64-bit IEEE floating point arithmetic.
The syntax is like C-based languages for the most part:

\section*{Example: Floating-point types}
```


# A simple float...

a = 123.456

# A float containing the integral value 2

b}=2

# Leading zeroes can be left off

c = . 3 \# same as 0.3

# Exponential / scientific representation

d = -123e-5

```

Two special cases are True and False, two constants that define the Boolean truth values. They behave similar to the integer values 0 and 1, except when they are converted into strings and return their names.

\section*{Example: \\ Booleans.py}
```


# booleans behave like integers, except when converted to strings.

# The built-in function "type" can be used to query the type of a

value.
print("True: ", True, type(True))
print("False: ", False, type(False))
print("1: ", 1, type(1))
print("False + 0: ", False + 0, type(False + 0))
print("True * 5: ", True * 5, type(True * 5))

```

Resulting output:
\begin{tabular}{l} 
True: True <type 'bool'> \\
\hline False: False <type 'bool'> \\
\hline 1: \(1<\) type 'int'> \\
\hline False \(+0: 0<\) <type 'int'> \\
\hline True * \(5: 5<\) type 'int'> \\
\hline
\end{tabular}

\section*{Strings}

In IronPython, strings are always in Unicode and any length. It does not make any difference if they are enclosed in ' or ". Strings can also have triple quotation marks "" " or ' ' ' , which allows for multiline string literals.
Similar to C, special characters can be excluded by means of backslash characters. As a comparison, the dollar sign (\$) is used in IEC for this purpose.
There are also raw strings that have other rules for the backslash. This is practical when the string should have literal backslashes. Example: Windows file paths or regular expressions.

\section*{Example： \\ Strings．py}
```


# encoding:utf-8

from __future__ import print_function
a = "a simple string"
b = 'another string'
c = "strings may contain 'quotes' of the other type."
d = "multiple string literals" ' are concatenated ' '''by the
parser'''
e = "Escaping: quotes: \" \' backslash: <br> newline: \r\n ascii
code: \x40"
f = """triple-quoted strings may contain newlines, "single"
'quotes' and '''multiquotes''' of the other type"""

```

```

h = r"c:\raw\strings\retain\backslashes.txt"

# we iterate over a sequence of all the variables defined above:

for i in (a,b,c,d,e,f,g,h):
print(i) \# prints the contents of the variable

```

Resulting output：
ascii code：＠
    triple-quoted strings may contain newlines, "single"
    'quotes' and "'multiquotes"' of the other type
    Üñiçóđe is also possible: 北京, Mockba, A日ŕva, القاهرة
    c:Vaw|strings\retain\backslashes.txt

Python does not have characters types．Characters are expressed by the use of strings with a length of 1 ．In this way，iteration via a string，or indexing in a string，returns a single－string．

\section*{Lists and tuples（data sets）}

Lists and tuples basically correspond to arrays in C and IEC，but there are some noticeable differences：
－The index access is always checked．Accessing a list or a tuple with an invalid index throws an exception．
－Both lists and tuples can contain elements of different types（also other lists and tuples）．In contrast in C and IEC，arrays can contain only elements of a single type．
－Lists are dynamic，and elements can be added，removed，or replaced at any time．
－Tuples are not changeable：Once a tuple is created，it cannot be modified anymore．
Lists are created with the list（）constructor．As an alternative，you can use brackets［］．
Tuples are created with the tuple（）constructor or parentheses（）．
```

Example:
list_tuples.
py

```
```

from __future__ import print_function

```
from __future__ import print_function
print("Testing tuples and lists")
print("Testing tuples and lists")
# We define a tuple with the numbers from 1 to 10:
# We define a tuple with the numbers from 1 to 10:
t = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
t = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
print("Tuple:", t)
print("Tuple:", t)
# We can access the 6th element of the tuple.
# We can access the 6th element of the tuple.
# As in C, index counting starts with 0.
# As in C, index counting starts with 0.
print("Element 5:", t[5])
print("Element 5:", t[5])
# Subscription is more powerful using the range syntax:
# Subscription is more powerful using the range syntax:
print("Range[2:5]:", t[2:5]) # lower bound is inclusive, upper
print("Range[2:5]:", t[2:5]) # lower bound is inclusive, upper
bound is exclusive.
bound is exclusive.
print("Range[2::2]:", t[2::2]) # start with 3rd element, and print
print("Range[2::2]:", t[2::2]) # start with 3rd element, and print
every 2nd element.
every 2nd element.
print("Range[-3:-1]:", t[-3:-1]) # Start with the 3rd last element,
print("Range[-3:-1]:", t[-3:-1]) # Start with the 3rd last element,
end just before the last element (upper bound is exclusive)
end just before the last element (upper bound is exclusive)
print("Range[::-1]:", t[::-1]) # negative step with - print
print("Range[::-1]:", t[::-1]) # negative step with - print
backwards
backwards
# lists are similar to tuples...
# lists are similar to tuples...
l = [11, 12, 13, "8", t] # contains mixed types: 3 integers, a
l = [11, 12, 13, "8", t] # contains mixed types: 3 integers, a
string, and the tuple defined above.
string, and the tuple defined above.
print("List:", l)
print("List:", l)
# ... but elements can be added or removed dynamically.
# ... but elements can be added or removed dynamically.
l.append(9) # Add a 9 to the list.
l.append(9) # Add a 9 to the list.
print("List with 9:", l)
print("List with 9:", l)
print("List Range[3:6:2]:", l[3:6:2]) # print the 4th and 6th
print("List Range[3:6:2]:", l[3:6:2]) # print the 4th and 6th
element.
element.
del l[1] # remove the element at index 1, the 12.
del l[1] # remove the element at index 1, the 12.
print("Removed[1]:", l)
print("Removed[1]:", l)
del l[1:3] # Remove the elements at index 1 and 2, the 13 and the
del l[1:3] # Remove the elements at index 1 and 2, the 13 and the
'8'.
'8'.
print("Removed[1:3]:", l)
```

print("Removed[1:3]:", l)

```

\section*{Resulting output:}
```

List: $[11,12,13, ~ ' 8 ', ~(1, ~ 2, ~ 3, ~ 4, ~ 5, ~ 6, ~ 7, ~ 8, ~ 9, ~ 10)] ~] ~$
List with 9: $\left[11,12,13,{ }^{\prime} 8\right.$ ', (1, 2, 3, 4, 5, 6, 7, 8, 9, 10), 9]
List Range[3:6:2]: [8', 9]
Removed[1]: [11, 13, '8', (1, 2, 3, 4, 5, 6, 7, 8, 9, 10), 9]
Removed[1:3]: [11, (1, 2, 3, 4, 5, 6, 7, 8, 9, 10), 9]

```

\section*{Dictionary}

Python also has a hash table type ( also "hashmap"). In contrast to the list, it can be indexed with any elements, for example strings. Its constructor is dict () and its literals are declared with braces \{\}.

The sample script dictionaries. py creates the output displayed below. In the last line, the script is terminated with a "KeyError" exception:

\section*{Example: dictionaries \\ . py}
```

from future__ import print function
print("Testing dictionaries")

# Declare a dictionary with three entries, the third being a list

d = {1: "a", 2: "b", "my list": [1, 2, 3]}
print(d)

# print the value of the key 1

print(d[1])

# remove the value with the key "my list"

del d["my list"]

# Add a value 4 with the key 3

d[3] = 4
print(d)

# The "get" method returns the second argument if the key cannot be

found.
print(d.get(1, 42))
print(d.get(23, 42))

# print all keys in the dictionary

for key in d:
print(key)

# index access for unknown keys will throw a "KeyError" exception!

print(d[23])

```

\section*{Resulting output:}
\begin{tabular}{|l|}
\hline Testing dictionaries \\
\(\{1:\) 'a', 2: 'b', 'my list': \([1,2,3]\}\) \\
\hline a \\
\(\{1:\) 'a', \(2:\) 'b', \(3: 4\}\) \\
\hline a \\
\hline 42 \\
\hline 1 \\
\hline 2 \\
\hline 3 \\
\hline
\end{tabular}

And then in the last line, the script terminates:


\section*{Python Control Structures (with Examples)}

Loops
As opposed to C and ST , for loops in Python do not count loop variables, but iterate over a sequence. This kind of sequence can be a "dictionary", a list, a tuple, the characters in a string, or lines in a file.
The following example shows some for loops:
```

Example:
loops.py

```
```

from __future__ import print_function

```
from __future__ import print_function
print("Enumerating over a simple list:")
print("Enumerating over a simple list:")
for i in (1,2,3,4):
for i in (1,2,3,4):
    print(i, end=", ") # end= replaces the newline with ", "
    print(i, end=", ") # end= replaces the newline with ", "
print() # but we still need a newline at the end of
print() # but we still need a newline at the end of
this case.
this case.
print("Enumerating over the characters in a string:")
print("Enumerating over the characters in a string:")
for i in "CODESYS": # characters are representet as strings of
for i in "CODESYS": # characters are representet as strings of
length 1.
length 1.
    print(i, end=", ")
    print(i, end=", ")
print()
print()
print("Enumerating over the integers 1 to 4:")
print("Enumerating over the integers 1 to 4:")
for i in range(1, 5): # upper bound is exclusive.
for i in range(1, 5): # upper bound is exclusive.
    print(i, end=", ")
    print(i, end=", ")
print()
print()
print("Enumerating using xrange:")
print("Enumerating using xrange:")
for i in xrange(5): # xrange is similar to range, but needs less
for i in xrange(5): # xrange is similar to range, but needs less
memory for large ranges.
memory for large ranges.
    print(i, end=", ")
    print(i, end=", ")
print()
print()
print("Enumerating including the item number:")
print("Enumerating including the item number:")
for i, v in enumerate("CODESYS"):
for i, v in enumerate("CODESYS"):
    print(i, v)
    print(i, v)
Resulting output:
\begin{tabular}{l} 
Enumerating over the characters in a string: \\
\(\mathrm{C}, \mathrm{O}, \mathrm{D}, \mathrm{E}, \mathrm{S}, \mathrm{Y}, \mathrm{S}\), \\
Enumerating over the integers 1 to 4 : \\
\(1,2,3,4\), \\
Enumerating using xrange: \\
\(0,1,2,3,4\), \\
Enumerating including the item number: \\
0 C \\
1O \\
2D \\
\hline 3 E \\
\hline 4S \\
\hline 5 Y \\
\hline 6 S \\
\hline
\end{tabular}
```

If you require an index or number in addition to the item, then you should use enumerate as shown in the last case of the sample script. The following code is considered as poor style:

Example: Poor
style

```
text = "CODESYS"
for i in range(len(text)): # BAD STYLE!
    v = text[i] # DON'T TRY THIS AT HOME!
    print(i, v)
```

Besides for loops, Python also has while loops which are very similar to those in C and ST:

## Example of "while" loop

```
i = 0
while i < 3;
    print(i)
    i += 1
```

Note: This example is not very practical. You would more likely use a for loop with a range.

## IF / ELSE

The if/else construct is similar to those in other programming languages. Here is a short example:

```
Example:
"if_else.py"
from __future__ import print_function
\(i=i n t\left(s y s t e m . u i . q u e r y \_s t r i n g(" P l e a s e ~ e n t e r ~ a n ~ i n t e g r a l\right.\)
number..."))
if i \(<0:\)
    print("Your number was negative.")
elif i > 0:
    print("Your numer was positive.")
else:
    print("It seems your number was zero.")
```

The else branch is optional and there can be zero, one, or many elif branches.

## Functions, classes, and methods

Python allows for defining functions and classes with methods. A class with methods is basically similar to a function block in ST, or classes in languages such as C++, Java, or C\#. However, Python does not support interfaces.

For detailed information, see the Python documentation for defining Functions and Classes.

```
Examples:
Functions,
classes, and
methods
#defining a function with name sum and two parameters a and b:
def sum(a, b):
        return a + b # we return the sum of a and b.
# we can now call the function defined above:
print(sum(5,7))
# Now we define a class Foo:
class Foo:
    # The class gets a method "bar".
    # Note: for methods, the first parameter is always "self" and
    # points to the current instance. This is similar to "this" in
    # ST and other languages.
    def bar(self, a, b):
                print("bar(%s,%s)" % (a,b))
# We create an instance of the class:
f = FOO()
# We call the method bar on the instance.
f.bar("some", "params")
```


## Modules and standard libraries

In IEC, you can import libraries for reuse by other written code. As a pendant, there is the possibility in Python of importing modules.

The Python standard library contains many modules for different purposes, such as:

- String processing
- Date and time handling
- Collections
- Threading
- Mathematical functions
- File handling
- Persistence
- Compression and archiving
- Database access
- Encryption services
- Network and Internet access
- Sending of emails

To create your own modules, write a Python file that defines the functions and classes that you want to provide. Save this file to the same directory as our sample script. If you name the file mymodule.py, then you can import it with import mymodule.

Here is an example of importing and using the cosine function and the pi constant from the math module:

```
Example:
Import mathe-
matical func-
tion
```

```
from math import cos, pi
```

from math import cos, pi
print(pi) \# prints 3.14159265359
print(pi) \# prints 3.14159265359
print(cos(pi)) \# prints -1.0

```
print(cos(pi)) # prints -1.0
```

The following contains more examples that access information about the operating system, the Python version, and the interpreter:

## More import examples

```
import os
print(os.environ["OS"])
from sys import platform, version, executable
print(platform)
print(version)
print(executable)
```

```
Windows_NT
cli
2.7.1 (CODESYS.exe CODESYS V3.5 SP4, ScriptEngine.plugin 3.5.4.0)
CODESYS.exe
```

There is a special module $\qquad$ future $\qquad$ for activating new language features. Above all, it is used when Python developers introduce new functionalities that are backward compatible. These kinds of functionalities have to be activated with special "_future imports". One example that we use in most of our sample scripts here is the activation of the new power syntax of print as a function instead of a statement.

## Example:

```
"__future__
```

\# make print() a function instead of a statement
from __future__ import print_function

The Python documentation provides a complete list of all imports.
In addition to the normal Python modules, IronPython code can also access .NET assemblies as if they were Python modules. This opens the access to the .NET framework class library and third-party libraries. Here is an example how to open a dialog by means of the Windows Forms library:

## Example: <br> Opening <br> a .NET dialog

```
import clr
clr.AddReference("System.Windows.Forms")
from System.Windows.Forms import MessageBox
MessageBox.Show("Hello")
```


## Transitioning from Python 2 to Python 3

With Python version 3, Python developers introduced some incompatible changes and removed some obsolete functionalities. At this time, the Python community is still in the transitional phase from version 2 to version 3.

IronPython does not yet support Python 3, but it is being worked on. As the Python community no longer supports Python 2, we intend to up0grade to Python 3 as soon as it is supported by IronPython.

Although we strive for a smooth transition, script writers should take care that their scripts are created in a future-proof style. For example, by using the expression from __future__ import print_function.

For more information, see: Python 2 or Python 3? and New in Python 3

## Comparison of IronPython and cPython

There are some small differences and Incompatibilities between IronPython and "standard" Python ("cPython"). Some are direct errors in IronPython and should be removed in future versions. However, others are considered "implementation details" and will remain. Some of them are very challenging topics.
The difference that is most obvious for users is the handling of strings. Original cPython has two different string types for "byte strings" and "Unicode strings". This concept is similar to the data types STRING and WSTRING in IEC. IronPython simply uses .NET strings that are always Unicode-capable and use UTF-16 internally. However, IronPython implements a trick to hide the difference to cPython from the programmer. (Interesting: For the new Python version 3, the developers have completely reworked their string handling. The result was a model which is much closer to IronPython. Unicode strings are always used after that and there is a separate data type for handling raw bytes)
Python modules that are written in C cannot be imported into IronPython because cPython uses internal data structures that are completely different from IronPython. Most of the standard library modules were reimplemented in IronPython. However, some modules (such as the TK interface) are not available as long as they are not ported to IronPython explicitly. On the other hand, IronPython provides access to .NET assemblies including the .NET framework (as shown above), which more than compensates for this feature.
While cPython uses reference counting and a deterministic garbage collector for cleaning up cyclic garbage, IronPython relies on the non-deterministic .NET garbage collector. In most cases, this difference does not matter. But when you open files or other resources from the Python standard library or the .NET framework, you should make sure to close them later. It is best to use the with statement with the Python context manager or .NET IDisposable instances.
For more information, see: Content Managers and .NET IDisposable

### 6.6.11.1.7 Using Scripts to Access CODESYS Functionalities

All objects and command which CODESYS provides for scripting are also available in the "scriptengine" Python module. Whenever a script is started, an implicit <code>from scriptengine import *</code> results. This allows for easy access to CODESYS. However, if your script imports modules that require access CODESYS APIs, then these modules have to import the module scriptengine themselves.
In the following table, you will find the main objects (categories) that can be used in Python scripts as entry points. For comprehensive documentation about entry points, see the for the CODESYS ScriptEngine.

| Objects | Description |
| :---: | :---: |
| System | Access to general CODESYS functionalities <br> Examples: <br> - Exiting CODESYS <br> - Handling the general user interface <br> - Access to the message memory (including compiler messages) <br> - Control of delay and progress bars |
| projects | Access to the CODESYS project as an object tree that combines the three navigator views (devices, POUs, modules) in one project tree <br> Also allows for the loading, creating, saving, and closing of projects <br> For most objects in a project, there are special methods with detailed functionality, for example compiling, access to ST POUs, export, import, device configuration, etc. |
| online | Access to online functionalities <br> Examples: <br> - Log in to devices and applications <br> - Management of access data (user name, password) <br> - Performance of network scans <br> - Gateway management |
| librarymanager | Permits the management of library repositories and viewing, installation, and removal of libraries |
| device_repository | Handling of device repositories; import and export of device descriptions |
| modulerepository | Management of CODESYS Application Composer modules and CODESYS Application Composer repositories |

See the following specific sample scripts for ways to access CODESYS functionalities. For detailed information, see the for the CODESYS.

## Example: Printing the device tree of the current project

The script PrintDeviceTree. py is an example for navigating in a project. It creates the output of a hierarchical display of all devices in the open project.

Load a project that contains some device objects and execute the script.

```
Example:
PrintDeviceT
ree.py
# encoding:utf-8
# We enable the new python 3 print syntax
from __future__ import print_function
# Prints out all devices in the currently open project.
print("--- Printing the devices of the project: ---")
# Define the printing function. This function starts with the
# so called "docstring" which is the recommended way to document
# functions in python.
def print_tree(treeobj, depth=0):
    """ Print a device and all its children
    Arguments:
    treeobj -- the object to print
    depth -- The current depth within the tree (default 0).
    The argument 'depth' is used by recursive call and
    should not be supplied by the user.
        """
        # if the current object is a device, we print the name and
device identification.
        if treeobj.is_device:
            name = treeobj.get_name(False)
            deviceid = treeobj.get device identification()
            print("{0}- {1} {2}".format("--"*depth, name, deviceid))
        # we recursively call the print_tree function for the child
objects.
        for child in treeobj.get_children(False):
            print_tree(child, depth+1)
# We iterate over all top level objects and call the print_tree
function for them.
for obj in projects.primary.get_children():
        print_tree(obj)
print("--- Script finished. ---")
```

The device tree (from the "Devices" view) is displayed in the message view and all non-device objects are left out:


## Example: Reading of variables

The script ReadVariable.py logs in to the device and starts the application if necessary. Then the value of the variable PLC_PRG.iVar1 is read and output. To try the script, you have to modify the project path and variable names.

```
Example:
ReadVariable
.py
```

```
# encoding:utf-8
```


# encoding:utf-8

from __future__ import print_function
from __future__ import print_function

# close open project if necessary:

# close open project if necessary:

if projects.primary:
if projects.primary:
projects.primary.close()
projects.primary.close()

# opens project

# opens project

proj = projects.open(r"D:\data\projects\Ampel.project")
proj = projects.open(r"D:\data\projects\Ampel.project")

# set "Ampel.project" to active application

# set "Ampel.project" to active application

app = proj.active_application
app = proj.active_application
onlineapp = online.create_online_application(app)
onlineapp = online.create_online_application(app)

# login to device

# login to device

onlineapp.login(OnlineChangeOption.Try, True)
onlineapp.login(OnlineChangeOption.Try, True)

# set status of application to "run", if not in "run"

# set status of application to "run", if not in "run"

if not onlineapp.application_state == ApplicationState.run:
if not onlineapp.application_state == ApplicationState.run:
onlineapp.start()
onlineapp.start()

# wait 1 second

# wait 1 second

system.delay(1000)
system.delay(1000)

# read value of iVar1

# read value of iVar1

value = onlineapp.read_value("PLC_PRG.iVar1")
value = onlineapp.read_value("PLC_PRG.iVar1")

# display value in message view or command line

# display value in message view or command line

print(value)
print(value)

# log out from device and close "Ampel.project"

# log out from device and close "Ampel.project"

onlineapp.logout()
onlineapp.logout()
proj.close()

```
proj.close()
```

In the extension of the script ReadVariable.py, the script MailVariables.py loads variables and expressions from a recipe file and reads their current values from the controller. Then these values are written back to the same file. In addition, is uses the Python SMTP library to send an email with an attachment containing a list of all variables.
To use the script, you have to modify the paths, email address, and the name of the SMTP server to your environment.

```
Example:
MailVariable
s.py
# encoding:utf-8
from __future__ import print_function
# Close current project if necessary and open "ScriptTest.project"
if not projects.primary == None:
    projects.primary.close()
project = projects.open("D:\\Data\\projects\\scriptTest.project")
# retrieve active application
application = project.active_application
# create online application
online_application = online.create_online_application(application)
# login to application.
online_application.login(OnlineChangeOption.Try, True)
# start PLC if necessary
if not online_application.application_state == ApplicationState.run:
    online_application.start()
# wait 2 seconds
system.delay(2000)
# open recipe file to read values.
recipe_input_file = open("D:\\Data\\projects\\RecipeInput.txt", "r")
watch_expressions = []
for watch_expression in recipe_input_file:
        watch_expressions.append(wātch_expression.strip())
print watch_expressions
# read values from the controllerd
watch_values = online_application.read_values(watch_expressions)
print watch_values
# open output file to write values
recipe_output_file = open("D:\\Data\\projects\\RecipeOutput.txt",
"w")
for i in range(len(watch_expressions)):
    recipe_output_file.write(watch_expressions[i])
    recipe_output_file.write(" = ")
    recipe_output_file.write(watch values[i])
    recipe_output_file.write("\n")
# Close files
recipe_input_file.close()
recipe_output_file.close()
# send Email
# import respective libraries
import smtplib
from email.mime.text import MIMEText
#open output file
recipe_output_file = open("D:\\Data\\projects\\RecipeOutput.txt",
"r")
mail = MIMEText(recipe_output_file.read())
```

```
recipe_output_file.close()
#email address sender and recipient
fromm = "info@example.com"
to = "info@example.com"
# set sender and recipient
mail["Subject"] = "Attention value has changed"
mail["From"] = fromm
mail["To"] = to
# send email
smtp = smtplib.SMTP("name of smtp server")
smtp.sendmail(fromm, [to], mail.as_string())
smtp.quit()
# logout and close application
online_application.logout()
projec\overline{t.close()}
```


## Example: Creating and editing of POUs

The script CreateDut. py creates the objects MyStruct, MyAlias, and MyUnion in the CODESYS project. The folder DataTypes already has to be present.

## CreateDut.py

```
# encoding:utf-8
from __future__ import print_function
STRUCT_CONTENT = """\
    a : BOOL;
    b : BIT;
    c : BIT;
UNION_WHOLE = """\
TYPE M
UNION
        Zahl : INT;
        Prozent : MyAlias;
        Bits : MyStruct;
END_UNION
END_TYPE
"""
proj = projects.primary
folder = proj.find('DataTypes', recursive = True)[0]
# Create a struct DUT and insert the list of variables just into
the right
# place in line two, row 0 (line numbering starts with line 0)
struktur = folder.create_dut('MyStruct') # DutType.Structure is the
default
struktur.textual_declaration.insert(2, 0, STRUCT_CONTENT)
# Alias types get their "content" via the base type, which will
just end up
# as one line in the declaration part:
# TYPE MyAlias : INT (0..100); END_TYPE
bereich = folder.create_dut('MyAlias', DutType.Alias, "INT
(0..100)")
# Instead of injecting the variables into the existing declaration,
# one can also just replace the complete declaration part,
including the
# boilerplate code.
union = folder.create_dut('MyUnion', DutType.Union)
union.textual_declaration.replace(UNION_WHOLE)
```

Example: User interface / Interaction with the user
In some cases, scripts have to interact with the user. We provide some simple APIs for the most common interactions. The sample script System_UI_Test. py shows all of the possible functions in this regard.

```
Example:
System_UI_Te
st.py
    # encoding:utf-8
    from __future__ import print_function
"""Performs some tests on the messagestore and UI."""
print("Some Error, Warning and Information popups:")
system.ui.error("Fatal error: Everything is OK. :-)")
system.ui.warning("Your bank account is surprisingly low")
system.ui.info("Just for your information: 42")
print("Now, we ask the user something.")
res = system.ui.prompt("Do you like this?", PromptChoice.YesNo,
PromptResult.Yes);
print("The user selected '%s'" % res)
print("Now, the user can choose between custom options:")
res = system.ui.choose("Please choose:", ("First", 2, 7.5,
"Something else"))
print("The user selected option '%s'" % str(res)) # res is a tuple
print("Now, the user can choose several options:")
res = system.ui.select_many("Please select one or more options",
PromptChoice.OKCancel, PromptResult.OK, ("La Premiere", "The
Second", "Das Dritte"))
print("The returned result is: '%s'" % str(res)) # res is a tuple
print("Now, the user can select files and directories")
res = system.ui.open_file_dialog("Choose multiple files:",
filter="Text files (*.txt)|*.txt|Image Files(*.BMP;*.JPG;*.GIF)|
*.BMP;*.JPG;*.GIF|All files (*.*)|*.*", filter_index = 0,
multiselect=True)
print("The user did choose: '%s'" % str(res)) # res is a tuple as
multiselect is true.
res = system.ui.save_file_dialog("Choose a file to save:",
filter="Text files (\overline{*}.txt)|*.txt|Image Files(*.BMP;*.JPG;*.GIF)|
*.BMP;*.JPG;*.GIF|All files (*.*)|*.*", filter_index = 0)
print("The user did choose: '%s'" % res)
res = system.ui.browse_directory_dialog("Choose a directory",
path="C:\\")
print("The user did choose: '%s'" % res)
print("Now we query a single line string")
res = system.ui.query_string("What's your name?")
print("Nice to meet you, dear %s." % res)
print("Now we query a multi line string")
res = system.ui.query_string("Please tell me a nice story about
your life!", multi_line=True)
if (res):
    print("Huh, that has been a long text, at least %s
characters!" % len(res))
else:
    print("Hey, don't be lazy!")
print("Username and passwort prompts...")
res = system.ui.query_password("Please enter your favourite
password!", cancellable=True)
if res:
    print("Huh, it's very careless to tell me your favourite
```

```
password '%s'!" % res)
else:
    print("Ok, if you don't want...")
res = system.ui.query_credentials("Now, for real...")
if res:
    print("Username '%s' and password '%s'" % res) # res is a
2-tuple
else:
    print("Sigh...")
```


## Example: Manipulating the "Project Information" object

In the script ProjectInfoExample.py, we set some information in the "Project Information" object. The most important information items, such as "Title" and "Version", have explicit properties. However, you can read and write any other information fields by means of the dictionary syntax. For example, those that are recommended for the properties of a library project.
The example below may seem somewhat unrealistic, but similar code is used in build servers that create, test, and possibly release automatic library projects and other projects. The ScriptEngine is one of the key elements for creating Cl (Continuous Integration) and CD (Continuous Delivery) systems.

```
Example:
ProjectInfoE
xample.py
```

```
# encoding:utf-8
```


# encoding:utf-8

from __future__ import print_function
from __future__ import print_function
proj = projects.load("D:\Some.library")
proj = projects.load("D:\Some.library")
info = proj.get_project_info()
info = proj.get_project_info()

# Set some values

# Set some values

info.company = "Test Library Ltd"
info.company = "Test Library Ltd"
info.title = "Script Test Project"
info.title = "Script Test Project"
info.version = (0, 8, 15, 4711)
info.version = (0, 8, 15, 4711)
info.default_namespace = "testlibrary"
info.default_namespace = "testlibrary"
info.author = "Python von Scriptinger"
info.author = "Python von Scriptinger"

# some values recommended in the library toolchain

# some values recommended in the library toolchain

info.values["DefaultNamespace"] = "testlibrary"
info.values["DefaultNamespace"] = "testlibrary"
info.values["Placeholder"] = "testlibrary"
info.values["Placeholder"] = "testlibrary"
info.values["DocFormat"] = "reStructuredText"
info.values["DocFormat"] = "reStructuredText"

# now we set a custom / vendor specific value.

# now we set a custom / vendor specific value.

info.values["SpecialDeviceId"] = "PLC0815_4711"
info.values["SpecialDeviceId"] = "PLC0815_4711"

# Enable generation of Accessor functions, so the IEC

# Enable generation of Accessor functions, so the IEC

# application can display the version in an info screen.

# application can display the version in an info screen.

info.change_accessor_generation(True)
info.change_accessor_generation(True)

# And set the library to released

# And set the library to released

info.released = True;
info.released = True;
proj.save()

```
proj.save()
```


## Example: Calling external commands and importing PLCOpenXML files

The sample script DeviceImportFromSVN.py gets a PLCOpenXML file from an external program (in this case a SVN client) and imports it into a new created CODESYS project.
To use the script, you have to modify the paths to your environment.

## Example: <br> DeviceImport <br> FromSVN.py

```
# encoding:utf-8
# Imports a Device in PLCOpenXML from Subversion via command line
svn client.
# We enable the new python 3 print syntax
from
```

$\qquad$

``` future
``` \(\qquad\)
```

import print_function
import sys, os

# some variable definitions:

SVNEXE = r"C:\Program Files\Subversion\bin\svn.exe"
XMLURL = "file:///D:/testrepo/testfolder/TestExport.xml"
PROJECT = r"D:\test.project"

# clean up any open project:

if projects.primary:
projects.primary.close()

# Fetch the plcopenxml data from subversion.

# We'll catch the output of the program into the xmldata variable.

# The 'with' construct automatically closes the open pipe for us.

with os.popen('"' + SVNEXE + '" cat ' + XMLURL, 'r') as pipe:
xmldata = pipe.read()

# create a new project:

proj = projects.create(PROJECT)

# import the data into the project.

proj.import_xml(xmldata, False)

# and finally save. :-)

proj.save()
print("--- Script finished. ---")

```

\section*{Advanced example: Calling a library from SVN and installing it in CODESYS}

The following sample script can perform the calling and installing of a library as part of a CT (Continuous Testing) environment so that they can be tested. In addition to standardCODESYS, the CODESYS SVN add-on also has to be installed with a valid license.

\section*{Example}
```

import tempfile
if projects.primary:
projects.primary.close()
tempdir = tempfile.mkdtemp()
URL = "svn://localhost/testrepo/trunk/SvnTestLibrary/"
proj = svn.checkout(URL, tempdir, "testlibrary", as_library=True)
proj.save()
repo = librarymanager.repositories[0]
librarymanager.install_library(proj.path, repo, True)
proj.close()

```

\subsection*{6.7 Human machine interface}

\subsection*{6.7.1 Control panels}

The CP600-eCo, CP600 and CP600-Pro control panels in different screen sizes provide comprehensive HMI functions for a wide range of applications. The engineering tool Panel Builder, part of Automation Builder, ensures easy scalability on the platform of the control panels.

\section*{Application examples/ application notes}

Control panels can be used with the AC500 PLCs for different purposes and different application environments. Example projects (application examples; ending with *.zip) and step-by-step instructions (application notes; ending with *.pdf) that demonstrate a specific use case can be found on the following topics:

A CP600 project can be converted to a project for control panels of the \(2^{\text {nd }}\) generation. The conversion procedure is described in the application note Converting a CP600 project.

Installation, licensing and configuration of the control panel runtime is described in the application note Control panel PC runtime.

How to enable and handle the runtime logging on control panels is demonstrated in the application example Control panel runtime logging.

How to dump data like trends, alarm buffers, audit trail buffers, recipes, etc. from the control panels to a network shared folder is described in the application note How to store data on a network shared folder.

Modbus RTU can be used for communication between an AC500-eCo V3 PLC and a control panel. The connection principle is described using the CP604 control panel as an example: CP604 connection via Modbus RTU.

How to upload a disk image from a control panel is described in the application note Upload disk image.

A summary of frequently asked questions related to the Boot Support Packages (BSP) and the control panels can be found in the application note FAQ on BSP.

How to check the BSP version of a CP600 HMI is described in the application note Check BSP version.
For a description of how to update the BSP in order to improve the control panel security and how to use enhanced product functionality see the application note Update BSP version.

How to update or reinstall a micro browser on an CP600-WEB control panel is described in the application note CP600-WEB and micro browser.

The control panels support web visualization via Chromium browser. How to setup and configure the Chromium browser for the usage in control panels is described in the application note Control panels and Chromium browser.

CP600 control panels contain a buzzer that can be used to give acoustic signals on certain events. The application note CP600 buzzer control describes how the buzzer can be controlled by using AC500 PLC tags on the connected PLC.

How to design and create control panel projects that comply with 21 CFR part 11 standard is described in the application note PB610 Panel Builder 600 V2.8.

The procedure for changing from a CP600 device to a CP600 device of the 2nd generation is described in the application example Converting a CP600 project. A description of the differences between both device variants can also be found in this application example.

How to connect an MQTT message broker with a CP600 is described in the application example CP600 MQTT.

\subsection*{6.7.2 Panel Builder interface}

\subsection*{6.7.2.1 General}

This document describes HMI CP600 Control Panel configuration in Automation Builder and starting HMI configuration and programming software Panel Builder 600 from Automation Builder. The Panel Builder project created for the HMI CP600 is stored within the Automation Builder project.

\section*{6．7．2．2 Adding desired AC500 PLC to the project}

\section*{Configuring the} Symbol File

1．In the Automation Builder device tree，right－click the＂Application＂node and click＂Add object＂
\begin{tabular}{|c|c|c|c|}
\hline \multicolumn{3}{|l|}{Add object below ：Application} & x \\
\hline \multicolumn{4}{|l|}{Object path：} \\
\hline \multicolumn{4}{|l|}{PLC＿AC500＿V3\Plc Logic／Application} \\
\hline \multicolumn{4}{|l|}{Object name：Symbols} \\
\hline \multicolumn{4}{|l|}{Categories Search object name．．} \\
\hline \multicolumn{2}{|l|}{\multirow[t]{15}{*}{Generic objects
IEC61131－3
Other
Scripting
Visualization}} & Name & － \\
\hline & & \multicolumn{2}{|l|}{\multirow[t]{3}{*}{\begin{tabular}{l}
Alarm Configuration \\
CheckConfiguration DUT
\end{tabular}}} \\
\hline & & & \\
\hline & & & \\
\hline & & \multicolumn{2}{|l|}{－O Interface} \\
\hline & & \multicolumn{2}{|l|}{8 Network Variable List（Receiver）} \\
\hline & & \multicolumn{2}{|l|}{6 Network Variable List（Sender）} \\
\hline & & \multicolumn{2}{|l|}{TPersistent Variables} \\
\hline & & \multicolumn{2}{|l|}{畋POU} \\
\hline & & \multicolumn{2}{|l|}{睹 POU for implicit checks} \\
\hline & & \multicolumn{2}{|l|}{M．Recipe Manager} \\
\hline & & \multicolumn{2}{|l|}{\(\square\) Script} \\
\hline & & \multicolumn{2}{|l|}{\(\square_{\text {－4 }}\) Symbol Configuration} \\
\hline & & \multicolumn{2}{|l|}{国Text List－} \\
\hline & & 1 & \\
\hline \multicolumn{4}{|l|}{，Close this dialog after each transaction \(\square\) Disp} \\
\hline Reset filter & & & \\
\hline
\end{tabular}

2．Click on＂Symbol Configuration＂and click＂Add object＂
\(\Rightarrow\) A＂Symbol Configuration＂object is added to the＂Application＂node．
3．Double－click on the＂Symbol Configuration＂object，then click on＂Build＂

\(\Rightarrow\) A list of all variables in the project is generated．Single variables or groups of variables can be selected by checking the corresponding item in the list．
4. After the symbols have been configured, download the project or click "Build \(\rightarrow\) Generate code" in the Automation Builder to create an .xml file containing all the variables read to be imported in the Tag Editor.

\subsection*{6.7.2.3 Creating a Panel Builder project}

\section*{Adding a panel} object
1. Right-click in the Automation Builder device tree and click "Add object \(\rightarrow\) Panel-CP600"
2. Click on "CP600 Control Panel" and click "Add object"
\(\Rightarrow\) A Control Panel object is added to the Automation Builder device tree.

\section*{Starting a Panel Builder project}
1. In the device tree, double-click "Panel CP600" object to start Panel CP600 screen.

2. Select the required PLC and enable the checkbox in the 'Use Standard Connection Settings' column to use it as a standard gateway connection.
You can set communication settings using the application program or by creating custom communication settings. Custom communication settings can be configured by clicking the button in the 'Details' column.
3. Enable the "Update Panel Builder project on launch" checkbox and click [Launch Panel Builder Editor].

If you update Automation Builder project with new variables and data types or if there are changes in existing Automation Builder project variables and data types (new, modified, deleted), recompile CODESYS application to refresh the symbol file, then launch Panel Builder editor.
4. Select "New" and click "Open" to create a new HMI project.
\(\Rightarrow\) A project wizard is displayed.

If you want to import an already existing Panel Builder project file from the file system, select "Import existing project file" and proceed.
5. Select the required panel type and orientation and click "Finish".
\(\Rightarrow\) A new project wizard starts only if the Panel project is empty.


The panel projects can be compared in Automation Builder using the "Compare Objects" option.

\section*{Changing panel type}
1. In the Panel project, double-click "Project properties" to change the panel type to the panel which is used.
\(\Rightarrow\) The Properties dialog is displayed.

2. In the Properties dialog, expand "Project" and click "Project Type ".
\(\Rightarrow\) A project wizard dialog is displayed.
3. Select the desired panel type and click "Finish".

Project informa- The project information view provides an overview of the Panel Builder project without opening tion the project. To open the project information, double-click the "Panel_CP600" object.

The project information is updated every time the Panel Builder project is edited. You can rename the Panel Builder project via context menu.

The project name is internally used as a base for the Panel Builder project file name. Therefore, the project name has to comply with general file name restrictions.

The Panel Builder project information shows the list of PLCs added to the project.


\subsection*{6.7.2.4 Configuring Panel Builder}

Configuring communication protocols

The user can configure a panel project manually in Panel Builder editor when there is a need to create individual panel projects. Otherwise, the configuration is updated in the panel project while launching Panel Builder editor in Automation Builder.
1. In the Panel Builder project structure, double-click "Config \(\rightarrow\) Protocols".
2. Click \(\pm\) to add a protocol.

3. Select "OPC UA Client" to ensure an encrypted communication between AC500 V3 devices and the control panels. This is necessary to protect passwords and other data in terms of cyber security.
Set the IP address, port, protocol type and PLC models. Click [OK].

\section*{Importing tags}
1. In the Panel project view, click "Config \(\rightarrow\) Tags".
2. Select the protocol from the drop-down list and click \(\sqrt{\boxed{ } 1}\) to import tags.

If the Panel Builder contains multiple tag importers, a dialog is displayed to select the required importer type.
3. Select the symbol file which was exported to the file system.
4. In the lower part of the tag editor, mark the desired tags and click "Import Tag (s)" to import the tags to the Panel Builder project.


Attaching tags to widgets
1. In the project view, expand "Pages" and double-click Page1.
2. In the Panel Builder 600 main menu, select "View \(\rightarrow\) Toolbars and Docking Windows \(\rightarrow\) Widget Gallery".
3. Drag-and-drop the desired widget to the page editor.
4. Right-click on the widget value and select "Attach To" to attach a tag to the widget.

5. Select the desired tag and select the desired option for the authorization "Read Only" or "Read/ Write" or "Write Only". Then, click [OK].

\section*{Downloading a project to panel}
1. In the Panel Builder main menu, click "Run \(\rightarrow\) Download To Target".
2. Select the CP600 project from the drop-down list and click "Download".

Importing an existing Panel Builder project
1. In the Automation Builder device tree, right-click the Panel project and click "Import \(\rightarrow\) Panel Builder Project".
System prompts to overwrite the exiting project object data.
2. Click "Yes" to confirm.
3. Select the existing Panel Builder 600 project from the file system and click "Open".
\(\Rightarrow\) The imported project is displayed.

\section*{Exporting Panel} Builder project
1. In the Automation Builder device tree, right-click the Panel Builder 600 project and click "Export \(\rightarrow\) Panel Builder Project".
2. Click "Browse" and select the desired location in the file system and save the project file.
\(\Rightarrow\) A success message is displayed, if the project file exports successfully.

When you double-click the Panel Builder project node, the compressed information of the node is extracted into a temporary folder and then the external Panel Builder program is started. After the external Panel Builder program is closed, the corresponding Panel Builder files can be compressed back into the node and saved in the Automation Builder project.

We recommend to edit the Panel Builder project by starting Panel Builder through the Automation Builder. You can also export a Panel Builder project to the file system to edit the project by using the external Panel Builder. Then, reimport it to Automation Builder.

\subsection*{6.7.3 SCADA Integration}

\subsection*{6.7.3.1 General}

This document describes SCADA integration configuration in Automation Builder using zenon editor. The configured device network address information and variables are synchronized with zenon editor to avoid double entry.
The Automation Builder supports both standard and multi-user functionality.

\subsection*{6.7.3.2 Creating Workspace and Project}
1. In the device tree, double-click "zenon_Project".

\(\Rightarrow\) To launch the zenon editor, click [Launch Zenon Editor].
To update the zenon project with latest changes of application program, click [Update zenon project].
2. Select the required PLC and select the "Use Standard Conn. Settings" option to use as a standard gateway connection.

This enables the user to use the same communication settings that Automation Builder uses to communicate to the PLC.

The configured gateway communication settings made in Automation Builder are displayed in the column 'Connection Type'.

As an alternative you can create custom communication settings: Deselect the "Use Standard Conn. Settings" option and click the button in the 'Details' column.
3. Click [Launch Zenon Editor] to create a new workspace and project.


Fig. 356: Connect to zenon project

4. Select the "Create a new workspace" option and select the file location to create a new workspace.
5. Select the "Create new project" option to create a project.
\(\Rightarrow \mathrm{ABB}\) zenon editor is displayed.

If you update or change an Automation Builder project with new variables or data types (new, modified, deleted), recompile the application to refresh the symbol file and click [Update zenon project].

After creating the project and workspace in Automation Builder, it is not required to set it again for the zenon object. A double-clicking on the zenon project shows the previously configured zenon project and the workspace.

\subsection*{6.7.3.3 Loading existing Workspace and Project}

You can load an existing workspace and project to ABB zenon supervisor.
1. In the zenon_Project screen, click [Update zenon project].
\(\Rightarrow\) Connection to the zenon project dialog is displayed.
2. In the workspace area, enable "Load existing workspace" and select the location.
3. In the project area, enable "Select loaded project" and click [OK].
\(\Rightarrow\) Zenon editor loads the selected existing workspace and the project.

\subsection*{6.7.3.4 Checking the Gateway Settings in a Zenon Project}

The gateway settings configured in Automation Builder can be checked in a zenon project. The IP address configured in Automation Builder are displayed in the zenon driver configuration.
In the Project Manager structure of the zenon editor, click "Variables \(\rightarrow\) Drivers" to configure the driver configuration.


The "Settings" tab shows all gateway settings based on the number of configured PLCs in Automation Builder. The IP address should be similar to the project gateway settings in Automation Builder.

In the zenon project window, the Connect column should be checked to transfer the desired number of PLC connection settings to the zenon editor.

\subsection*{6.7.3.5 Creating a symbol configuration}

Requirement: The project can be compiled without any errors.
1. Select the "Application" object in the device tree.
2. Click "Project \(\rightarrow\) Add Object \(\rightarrow\) Symbol Configuration".
\(\Rightarrow\) The "Symbol Configuration" object is added to the device tree and the objects editor opens.
3. Open the "View" menu of the editor and activate the categories of variables that should be provided in the configuration editor. Click "Build" in the symbol configuration editor.
\(\Rightarrow\) All variables (according to the currently defined filter in the "View" menu) are displayed in a tree structure.
4. Select the check boxes of individual variables.

Pay attention to the current settings (see the "Settings" button in the menu bar of the editor).
\(\Rightarrow\) In the field below the menu bar of the editor, information is provided about the current situation with accompanying instructions, as well as controls for corrective actions.
5. Follow the prompt in the field below the menu bar. In the following case, this should be only the information that the modified symbol configuration is transferred with the next download or online change.
6. Click "Build Generate Code" on the CODESYS menu bar.
\(\Rightarrow\) The <project name>.<device name>.<application name> .xml file is generated in the project directory.

CODESYS transmits the symbol configuration to the PLC for an application download or online change.

\subsection*{6.7.3.6 Updating Standard Data Types}

The standard data types created in CODESYS application can be updated to the zenon project by clicking on "Update zenon project".


Data types and variables can be updated from the desired number of PLCs configured in the zenon project of Automation Builder.

In the zenon project, double-click "Variables" and check the updated standard data type.

\subsection*{6.7.3.7 Creating Data Types}
1. In the CODESYS application open the "Data types" tab. Right-click "Data types \(\rightarrow\) Add object" to create a new data type.
2. Enter the user defined data type name.
3. In "POUs" tab, add the user defined variable data type and compile.
\(\Rightarrow\) The user defined data type is created and can be imported in the zenon editor.

If you modify or delete the data types in CODESYS application, compile with
"Rebuild all option".

\subsection*{6.7.3.8 Importing Data Types in zenon Editor}
1. In the zenon project, click [Update zenon project] to update the data types.
2. Click "Update" to update the variables and data types to the zenon project.
\(\Rightarrow\) The user defined variables and data types are imported to the zenon project.


Fig. 357: User defined variables

\subsection*{6.8 System technology}

\subsection*{6.8.1 General}

This chapter provides advanced information on the system technology of AC500 control systems from a general perspective. It provides information to link the details from the hardware descriptions (provided in the device specifications section) with detailed information on configuring/programming a corresponding library (provided in the individual library sections).

Configuration of a specific device with Automation Builder is described in the PLC configuration section.

\section*{6．8．2 System technology of CPU and overall system}

\section*{6．8．2．1 Handling of remanent variables for AC500 V3 products}

\section*{68．8．2．1．1 Introduction}
The retain／persistent memory must be buffered by a battery TA521 for the
PLCs PM56xx－2ETH．Following described functionalities are only working if a
battery is inserted．Take care about the handling for TA521 battery．
』 4 Chapter 5．2．8．2．3＂TA521－Battery＂on page 1319
The AC500－eCo V3 PLCs，PM50xx－ETH PLCs don＇t need a battery．
』＂Detailed data＂on page 188

All operands supported by CODESYS are described in＇Configuring I／O links＇\({ }^{\mu}\) Chapter 6．4．1．8 ＂Configuring I／O Links＂on page 1835．For the memory sizes of the different CPUs，see for AC500－eCo V3＇Memory sizes＇圱＂AC500－eCo V3 processor modules＂on page 4413 and for AC500 V3＇Memory sizes＇«＂AC500 V3 processor modules＂on page 4413.

This part of the documentation describes the declaration of remanent variables for AC500 V3 products．

\section*{Different handling of remanent variables in AC500 FW \(\geq\) V3．0．2}
－No more \％R memory area（use instead \％M with \｛no＿init\} \({ }^{\mu}\) Chapter 6．8．2．1．6＂Initialization of \％M variables＂on page 4417）
－Creating of addresses for＂VAR RETAIN PERSISTENT＂variables automati－ cally by IEC Compiler

It is NOT possible to change the structure（e．g．add，delete，change order，．．）of retain／persis－ tent variables of a project and update the project via memory card．
Up to version of SystemFW 3．4．x the boot project will be deleted（renamed into application．err） and the PLC will not load the boot project anymore．Also download of new／other project with Automation Builder failes．

Workaround：
－Automation Builder \(\rightarrow\) PLC Shell \(\rightarrow\) clearsram all \(\rightarrow\) retain persistent／retain area is deleted （not the \％M area）\(\rightarrow\) application is running after reboot with initialized／retain／persistent data
－Automation Builder \(\rightarrow\) PLC Shell \(\rightarrow\) clearsram all \(\rightarrow\) retain persistent／retain area is deleted \(\rightarrow\) sram c m（clear \％M）\(\rightarrow\) \％M area is deleted \(\rightarrow\) reboot \(\rightarrow\) application is running with initialized／retain／persistent／\％M data
－Automation Builder empty project \(\rightarrow\) Reset Origin Device \(\rightarrow\) retain persistent／retain area／ \(\% \mathrm{M}\) area is deleted \(\rightarrow\) new update via SD card \(\rightarrow\) application is running after reboot with initialized data
－For midrange reboot without battery \(\rightarrow\) application is running after reboot with initialized data

The application is running with initialized persistent and／or retain after updating the application via memory card and rebooting the PLC．
In version of SystemFW 3．5．x the changed retain persistent／retain area is deleted．Application is running after reboot with initialized／retain／persistent data．
In case of trouble use the above described workaround．

\subsection*{6.8.2.1.2 Memory sizes}

AC500-eCo V3 processor modules
\begin{tabular}{|c|c|c|c|c|c|}
\hline PLC type & system RAM disk & userdisk PlcLogic ... & Retain, \%M area & flash disk & memory card \\
\hline \[
\begin{aligned}
& \text { PM5012-x- } \\
& \text { ETH }
\end{aligned}
\] & \multirow[t]{4}{*}{Dynamically /max. 7.6 MB} & \multirow[t]{4}{*}{30 MB} & \begin{tabular}{l}
8 kB \\
Retain and persistent 4 kB (of which 88 byte are reserved for allocation table and are not available to the user) \\
\%M 4 kB
\end{tabular} & \multirow[t]{4}{*}{None} & \multirow[t]{4}{*}{๕ Chapter 5.2.8.2.1 "MC5102 Micro memory card with adapter" on page 1309} \\
\hline \[
\begin{aligned}
& \text { PM5032-x- } \\
& \text { ETH }
\end{aligned}
\] & & & \begin{tabular}{l}
\[
32 \mathrm{kB}
\] \\
Retain and per-
\end{tabular} & & \\
\hline \[
\begin{aligned}
& \text { PM5052-x- } \\
& \text { ETH }
\end{aligned}
\] & & & sistent 16 kB (of which 88 byte are reserved for allocation table and are not available to the user) \%M 16 kB & & \\
\hline \begin{tabular}{l}
PM5072- \\
T-2ETH(W) \\
PM5082- \\
T-2ETH
\end{tabular} & & & \begin{tabular}{l}
100 kB \\
Retain and persistent 36 kB (of which 88 byte are reserved for allocation table and are not available to the user) \\
\%M 64 kB
\end{tabular} & & \\
\hline
\end{tabular}

AC500 V3 processor modules
\begin{tabular}{|c|c|c|c|c|c|}
\hline PLC type & system RAM disk & userdisk PlcLogic ... & \begin{tabular}{l}
SRAM \\
Retain, \%M area
\end{tabular} & flash disk & memory card \\
\hline \[
\begin{aligned}
& \text { PM5630-2ET } \\
& \mathrm{H}
\end{aligned}
\] & Dynamically /max. 7.6 MB & \begin{tabular}{l}
40 MB \\
30 MB (as of V3.4.0)
\end{tabular} & \multirow[t]{2}{*}{\begin{tabular}{l}
256 kB \\
Retain and persistent 128 kB (of which 68 byte are reserved for allocation table and are not available to the user) \\
\%M 128 kB
\end{tabular}} & \multirow[t]{2}{*}{None} & \multirow[t]{2}{*}{\begin{tabular}{l}
4) Chapter 5.2.8.2.2 \\
"MC5141 - \\
Memory card" on page 1315 \\
«2) Chapter 5.2.8.2.1 \\
"MC5102 Micro memory card with adapter" on page 1309
\end{tabular}} \\
\hline \[
\begin{aligned}
& \text { PM5650-2ET } \\
& \text { H }
\end{aligned}
\] & Dynamically /max. 16 MB & \[
\begin{aligned}
& 246 \text { MB (as } \\
& \text { of V3.0.x) } \\
& 381 \text { MB (as } \\
& \text { of V3.1) } \\
& 285.75 \text { (as of } \\
& \text { V3.4.0) }
\end{aligned}
\] & & & \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|}
\hline PLC type & system RAM disk & \begin{tabular}{l}
userdisk \\
PlcLogic
\end{tabular} & \begin{tabular}{l}
SRAM \\
Retain, \%M area
\end{tabular} & flash disk & memory card \\
\hline PM5670-2ET & \multirow[t]{2}{*}{Dynamically /max. 69 MB} & \multirow[t]{2}{*}{\[
\begin{array}{|l|}
\hline 858 \mathrm{MB} \\
643.50 \mathrm{MB} \\
\text { (as of V3.4.0) }
\end{array}
\]} & \multirow[t]{2}{*}{\begin{tabular}{l}
1536 MB \\
1 MB retain and persistent (of which 68 byte are reserved for allocation table and are not available to the user) \\
512 kB \%M
\end{tabular}} & & \\
\hline PM5675-2ET
H & & & & 8 GB & \\
\hline
\end{tabular}

It is not possible to use 100 \% of a device's memory space. About \(10 \%\) of the total available space must remain unused at any time to maintain normal device operation.

\subsection*{6.8.2.1.3 Adding a global list of persistent/retain variables}

A global list of persistent variables will be added with the standard definition for persistent variables "VAR RETAIN PERSISTENT" « Chapter 6.4.1.20.2.14 "Retain Variable - RETAIN" on page 2124 \({ }^{〔}\) Chapter 6.4.1.20.2.13 "Persistent Variable - PERSISTENT" on page 2122.

First steps:
1. Expand the object path of your PLC
2. After right click on "App" select "Add object" in the context menu.

The window "Add object below: Application" appears.

3. Select "Persistent Variables" and click [Add object].

The object name can be chosen freely. In the application it will be reused to reference the persistent variables.

\subsection*{6.8.2.1.4 Declaring a new variable in global list}

Declare a new variable in the window "GlobPersist".


Afterwards the variable can be selected in the program.


In this way the persistent variable can be accessed directly.

Do not use the same persistent variable in different IEC tasks, to avoid problems with consistency.
6.8.2.1.5 Declaring a new persistent/retain variable in local POU

It is also possible to declare a persistent/retain variable in a local POU and not in the global list of persistent variables.

It is not recommended to declare a large number of persistent variables locally, due to the potentially effect to performance.


The auto-declare mechanism declares always a persistent variable locally and not in the global list. If the program will be executed, the following warning appears in the message window:

C0244: No VAR_PERSISTENT-list is part of the application to enter instance path for variable PLC_PRG.new_var1 The locally declared persistent variable has to be added to the global list.

\section*{NOTICE!}

For the initialization of a Retain/Persistent variable the value of the global list is used NOT the value of the local declaration.

For further information see "RetainPersistentExample.project".

1. Right-click in window "PersistentVars".
2. Select option "Add all instance paths".
\(\Rightarrow\) Afterwards the persistent variables are added.


The application can be downloaded to the PLC


> It is NOT recommended to declare a new persistent variable in the application due to performance problems.

For example PM5650-2ETH:
1000 DWORD \(\approx 600 \mu\) s additional cycle time of task.

\subsection*{6.8.2.1.6 Initialization of \(\% M\) variables}

After download or restart, all \%M variables will be initialized to 0 . This can be prevented by setting the "no_init" attribute.

In doing so the \%M variables behave similar to the "VAR RETAIN PERSISTENT" variables.
```

cnt : DWORD := 0;
setvar : BOOL := FALSE;
(attribute 'no init'}
ProzMivar AT \&MW0: WORD;
ProzMivarField AT \&MW1: ARRAY [1..1000] OF WORD;
ProzMivar1 AT \&MW1001: WORD;
adतn UarDeva1 . mNTNTPD m ntave.

```

In the example above variable "ProzMivar" has the attribute "no_init". This variable will not be initialized and keeps its last value.

The attribute "no_init" is always and only valid for the next following variable \({ }_{\mu} \Rightarrow\) Chapter 6.4.1.20.6.3.32 "Attribute 'noinit"' on page 2299.

The following two variables "ProzMivarField" and "ProzMivar1" will be further on initialized to 0.

\subsection*{6.8.2.1.7 Behavior of retain variables}

The declaration of the retain variables strictly follows the 3 Standard (see Remanent variables « Chapter 6.4.1.20.2.14 "Retain Variable - RETAIN" on page 2124 y Chapter 6.4.1.20.2.13 "Persistent Variable - PERSISTENT" on page 2122).

For retain variables it does not matter if they are declared locally in a program or in the global variable list.
6.8.2.1.8 PLC shell command for import and export of retain/persistent variables

The syntax of the command is: sram <direction><area><path>

\section*{Supported options:}
\begin{tabular}{|l|l|}
\hline Direction: & i=import, \(\mathrm{e}=\) export \\
\hline Area: & rp=Retain/Persistent, \(\mathrm{m}=\% \mathrm{M}\) area \\
\hline Path: & Any pathname \\
\hline
\end{tabular}

The file will be stored in the user partition of the PLC. This data can be imported or exported via the FTP-Server or the Files dialog in Automation Builder.
If no path is indicated, the files are saved under "PlcLogic/<ApplicationName>/<ApplicationName>.ret or .prozm".
If a path is indicated, the files are saved under or accessed via
"<path>/<ApplicationName>.ret or .prozm".
A non-existing path is created with the exception of the memory card. The path for the memory card must be an existing path. On the memory card a non-existing path leads to an error message.
\begin{tabular}{|l|l|l|}
\hline Data area & File extension & Path \\
\hline Retain/Persistent & .ret & \begin{tabular}{l} 
PlcLogic/<ApplicationName>/<Application- \\
Name>.ret \\
<path>/<ApplicationName>.ret
\end{tabular} \\
\hline \%M (memory area) & .prozm & \begin{tabular}{l} 
PlcLogic/<ApplicationName>/<Application- \\
Name>.prozm \\
<path>/<ApplicationName>.prozm
\end{tabular} \\
\hline
\end{tabular}

\section*{Examples:}
\begin{tabular}{|l|l|l|}
\hline Application & Command & File \\
\hline \multirow{3}{*}{ myApp } & sram e rp & PlcLogic/myApp/myApp.ret \\
\cline { 2 - 3 } & sram e ep data & data/myApp.ret \\
\hline Application & sram i m & \begin{tabular}{l} 
PlcLogic/Application/Applica- \\
tion.prozm
\end{tabular} \\
\cline { 2 - 3 } & sram i m data & data/Application.prozm \\
\hline
\end{tabular}

If the path "data" does not exist, tha path is created. The path for the memory card must be an existing path. The path "sdcard/data" leads to an error message if the path "data" does not exists on the memory card.

Only if the application uses Retain or Retain/Persistent variables the command generates an output file.

\subsection*{6.8.2.1.9 Import and export of retain/persistent variables by library functions}

\section*{General}

It is also possible to import or export the Retain/Persistent variables and the \%M markers via system function calls from the PLC Application. The required system functions are implemented in the IEC library ABB_IntUtils_AC500.library.
It provides the following functions or function blocks:
- SRAM_IMPORT

Chapter 6.8.2.1.9.2 "SRAM_IMPORT" on page 4419
- SRAM_EXPORT
* \({ }^{\wedge}\) Chapter 6.8.2.1.9.3 "SRAM_EXPORT" on page 4419
- SRAM_CLEARED
\({ }^{4}\) ) Chapter 6.8.2.1.9.4 "SRAM_CLEARED" on page 4419

\section*{SRAM_IMPORT}

The function block SRAM_IMPORT is used to import the \%M markers and the Retain/Persistent variables from the specified files in the userdisk.

Import only those \%M markers and/or Retain/Persistent variables that are compatible to the application running in the PLC.

位 area is not accessed by the application.

Otherwise inconsistencies are possible.

For a complete description of the function block see ABB_IntUtils_AC500.library.

\section*{SRAM_EXPORT}

The function block SRAM_EXPORT is used to export the \%M markers and the Retain/Persistent variables to the specified files in the userdisk.

Export only those \%M markers and/or Retain/Persistent variables that are compatible to the application running in the PLC.

It is recommended to export only when the \%M and/or the Retain/Persistent area is not accessed by the application.
Otherwise inconsistencies are possible.

For a complete description of the function block see ABB_IntUtils_AC500.library.

\section*{SRAM_CLEARED}

The Function SRAM_CLEARED is used to check if the SRAM was deleted.
For a complete description of the Function see ABB_IntUtils_AC500.library.

\subsection*{6.8.2.2 System processing}

\subsection*{6.8.2.2.1 System start-up / Program processing}

\section*{Definitions: PLC system start-up}

\section*{Cold start}

Warm start - A warm start is performed by switching power OFF/ON with a battery connected.
- All RAM memory modules are checked and erased except of the buffered operand areas and the RETAIN variables \({ }^{*}>\) Chapter 6.4.1.21.3.7.11 "Command 'Reset Warm"' on page 2634.
- If there is a user program stored in the Flash EPROM, it is loaded into RAM.
- The default operating modes set by the PLC configuration are applied.
- By booting the CPU the connected IO modules are re-initialized and parameterized, therefore the IO output fails for a short time.

RUN -> STOP - RUN -> STOP means pressing the RUN function key on the PLC while the PLC is in run mode (AC500 PLC display "run", AC500-eCo PLC "RUN LED" is ON).
- If a user program is loaded into RAM, execution is stopped.
- The behavior of outputs in stop state is configurable. Double click on PLC_AC500_V3 in the project tree. In the "PLC_AC500_V3" tab click on "PLC Settings".
- To activate update of I/O while in stop state, click the checkbox left of "Update I/O while in stop".
- To specify the behavior of the outputs in stop state, select an option in the pull down menu right of "Behavior for outputs in stop". Options are:
"Keep current values"
"Set all outputs to default"
"Execute program" (If the "Execute program" option is selected, the "Input Assistant" must be opened via the [...] button and a program must be selected which is executed exactly once.)
- Variables keep their current values, i.e., they are not initialized.
- The AC500 PLC display changes from "run" to "StoP", AC500-eCo "RUN LED" changes from ON to OFF.

START -> STOP - START -> STOP means stopping the execution of the user program in the PLC's RAM using the menu item "Online/Stop" in the programming system.
- The behavior of the outputs in stop state is configurable. Double click on PLC_AC500_V3 in the project tree. In the "PLC_AC500_V3" tab click on "PLC Settings".
- To activate update of I/O while in stop state, click the checkbox left of "Update I/O while in stop".
- To specify the behavior of the outputs in stop state, select an option in the pull down menu right of "Behavior for outputs in stop". Options are:
"Keep current values"
"Set all outputs to default"
"Execute program" (If the "Execute program" option is selected, the "Input Assistant" must be opened via the [...] button and a program must be selected which is executed exactly once.)
- Variables keep their current values, i.e., they are not initialized.
- The AC500 PLC display changes from "run" to "StoP".

Reset - Performs a START -> STOP process.
- Preparation for program restart, i.e., the variables (VAR) (exception: RETAIN variables) are set to their initialization values.
- Reset is performed using the menu item "Online/Reset" in the programming system or pressing the function key RUN for \(\geq 5 \mathrm{~s}\) in STOP mode.

Reset (cold) - Performs a START -> STOP process.
- Preparation for program restart, i.e., the variables (VAR) (also RETAIN variables) are set to their initialization values.
- Reset (cold) is performed using the menu item "Online/Reset (cold)" in the programming system.

Reset (original) - Resets the controller to its original state (deletion of Flash, SRAM (\%M, area, \%R area, RETAIN, RETAIN PERSISTENT), Communication Module configurations and user program!).
- Reset (original) is performed using the menu item "Online/Reset (original)" in the programming system.

STOP \(\rightarrow\) RUN - STOP \(->\) RUN means short pressing the RUN function key on the PLC while the PLC is in STOP mode (AC500 PLC display "StoP", AC500-eCo "RUN LED" is ON). "RUN LED" is OFF of the toggle switch of an AC500-eCo CPU.
- If a user program is loaded into RAM, execution is continued, i.e., variables will not be set to their initialization values.
- The AC500 PLC display changes from "StoP" to "run", AC500-eCo "RUN LED" changes from OFF to ON.

STOP -> START • STOP -> START means continuing the execution of the user program in the PLC's RAM using the menu item "Online/Start" in the programming system.
- If a user program is loaded into RAM, execution is continued, i.e., variables will not be set to their initialization values.
- The AC500 PLC display changes from "StoP" to "run", AC500-eCo PLC "RUN LED" changes from OFF to ON.

Download
- Download means loading the complete user program into the PLC's RAM. This process is started by selecting the menu item "Online/Download" in the programming system or after confirming a corresponding system message when switching to online mode (menu item "Online/Login").
- Execution of the user program is stopped.
- In order to store the user program to the Flash memory, the menu item "Online/Create boot project" must be called after downloading the program.
- Variables are set to their initialization values according to the initialization table.
- RETAIN variables can have wrong values as they can be allocated to other memory addresses in the new project!
- A download is forced by the following:
- changed PLC configuration
- changed task configuration
- changed library management
- changed compile-specific settings (segment sizes)
- execution of the commands "Project/Clean all" and "Project/Rebuild All".

Online change - After a project has changed, only these changes are compiled when pressing the key <F11> or calling the menu item "Project/Build". The changed program parts are marked with a blue arrow in the block list.
- The term Online Change means loading the changes made in the user program into the PLC's RAM using the programming system (after confirming a corresponding system message when switching to online mode, menu item "Online/Login").
- Execution of the user program is not stopped. After downloading the program changes, the program is re-organized. During re-organization, no further online change command is allowed. The storage of the user program to the Flash memory using the command "Online/ Create boot project" cannot be initiated until re-organization is completed.
- Online Change is not possible after:
- changes in the PLC configuration
- changes in the task configuration
- changes in the library management
- changed compile-specific settings (segment sizes)
- performing the commands "Project/Clean all" and "Project/Rebuild All".

Data buffering - Data buffering, i.e., maintaining data after power ON/OFF, is only possible, if a battery is connected for AC500 CPU and the buffering will take place in FLASH with AC500-eCo V3 CPU. The following data can be buffered completely or in parts:
- Data in the addressable flag area ( \(\% \mathrm{M}\) area)
- RETAIN variable
- PERSISTENT variable (number is limited, no structured variables)
- PERSISTENT area (\%R area)
- In order to buffer particular data, the data must be excluded from the initialization process
* Chapter 6.8.2.1 "Handling of remanent variables for AC500 V3 products" on page 4412.

\section*{Start of the user program}

The user program (UP) is started according to the following table. It is assumed that a valid user program is stored to the Flash memory \({ }^{\wedge}>\) Chapter 5.2.9.4 "Storage device details" on page 1336.
\begin{tabular}{|c|c|c|c|c|}
\hline Action & \begin{tabular}{l}
No memory card with UP installed \\
Auto run = ON
\end{tabular} & \begin{tabular}{l}
No memory card with UP installed \\
Auto run = OFF
\end{tabular} & \begin{tabular}{l}
Memory card with UP installed \\
Auto run = ON
\end{tabular} & \begin{tabular}{l}
Memory card with UP installed \\
Auto run = OFF
\end{tabular} \\
\hline Voltage ON or Warm start or Cold start & UP is loaded from Flash into RAM and started from Flash. & No UP is loaded from Flash. When logging in, the message "No program available in the controller ..." is displayed. & UP is loaded from the memory card into Flash memory and RAM and then started from RAM. & UP is loaded from the memory card to the Flash memory. RAM remains empty. When logging in, the message "No program available in the controller ..." is displayed. \\
\hline STOP -> RUN & UP in RAM is started. & UP in RAM is started & UP in RAM is started. & UP in RAM is started. \\
\hline STOP -> START & UP in RAM is started. & UP in RAM is started. & UP in RAM is started. & UP in RAM is started. \\
\hline Download \({ }^{1)}\) & The UP currently stored in the CPU's RAM is stopped. The built UP is loaded from the PC into the PLC's RAM. & The built UP is loaded from the PC into the PLC's RAM. & The UP currently stored in the CPU's RAM is stopped. The built UP is loaded from the PC into the PLC's RAM. & The built UP is loaded from the PC into the PLC's RAM. \\
\hline Online Change \({ }^{2)}\) & Processing of the UP currently stored in the CPU's RAM is continued. The changes made to the UP are loaded from the PC into the PLC's RAM. The UP is reorganized and processed. & The changes made to the UP are loaded from the PC into the PLC's RAM. The UP is reorganized. & Processing of the UP currently stored in the CPU's RAM is continued. The changes made to the UP are loaded from the PC into the PLC's RAM. The UP is reorganized and processed. & The changes made to the UP are loaded from the PC into the PLC's RAM. The UP is reorganized. \\
\hline
\end{tabular}

\section*{Remarks:}
1): After the download is completed, the program is not automatically stored to the Flash memory. To perform this, create a boot project \({ }^{\wedge} \geqslant\) Chapter 6.4.1.11.7 "Generating boot applications" on page 1978. If the UP is not stored to the Flash memory, the UP is reloaded from the Flash memory after voltage OFF/ON. Start the program either by pressing the RUN/STOP function key or using Automation Builder.
2): After the online change process is completed, the program is not automatically stored to the Flash memory. For this, after reorganization is completed create a boot project. During reorganization and flashing, no further online change command is allowed. If the UP is not stored to the Flash memory, the UP is reloaded from the Flash memory after voltage OFF/ON.
2): After the online change process is completed, the program is not automatically stored to the Flash memory. For this, after reorganization is completed create a boot project \({ }^{\wedge} \Rightarrow\) Chapter 6.4.1.11.7 "Generating boot applications" on page 1978. During reorganization and flashing, no further online change command is allowed. If the UP is not stored to the Flash memory, the UP is reloaded from the Flash memory after voltage OFF/ON.

\section*{User flash memory for AC500 V3 products}

AC500 processor modules for V3 products (PM56xx) are equipped with non-removable and non-volatile onboard user flash memory for program and data storage. The integrated flash management, including a wear levelling algorithm and a power-fail protected file system, is designed for robustness and operation in industrial environments and applications. The user flash memory can be accessed from the user program using the CAA_File library .


\section*{Health monitoring}

AC500 V3 products are equipped with non-removable and non-volatile onboard user flash memory for program and data storage. The integrated flash management, including a wear levelling algorithm and a power-fail protected file system, is designed for robustness and operation in industrial environments and applications.

Keep the amount of cyclic written data low to assure long availability of the user flash memory. The spent/remaining lifetime information of the user flash memory can be acquired with the function block PmDiskStatus and PmDiskLifetimeUsed.
« Chapter 5.2.9.4.2 "Flash disk" on page 1336

Further information is provided in the documentation of the AC500_Pm library. \(\Leftrightarrow\) Chapter 6.5.14 "Reference, function blocks" on page 4086

Since FW version 3.3.0, there is also a diagnosis message issued when the user flash memory reaches the end of its lifecycle. Please refer to the diagnosis documentation for more info.

\section*{PID controller}

Using PID function blocks, such as HaModPidFixCycle, a PID controller can be configured and used in AC500 PLCs. The procedure is described in the application note PID - function block.

\section*{Task configuration}

\section*{General}

\section*{This statement is applicable to PM5032-x-ETH, PM5052-x-ETH, PM5072-T-2ETH(W) and PM5082-T-2ETH.}

If the main task cycle is faster than 10 ms , remove the onboard inputs I8..I11 and I/O channels of option boards or option boards for serial communication from the main task cycle, but use a separate task cycle.

The task model processes the following kind of tasks:
- Non-real-time system tasks: system tasks with no real-time property (e.g. file access, Ethernet communication, OPC UA, ...)
- Non-real-time IEC tasks: IEC tasks with no real-time property
- Real-time system tasks: system tasks with real-time property
- Real-time IEC tasks: IEC tasks with real-time property

The possible number of tasks depends on the type of processor module. How to distribute the IEC tasks over multiple CPU cores and on how to use the IEC task configuration for Automation Builder is described in detail in the CODESYS task configuration section.
- Task configuration \({ }^{〔}\) Chapter 6.4.1.9.17 "Task Configuration" on page 1914
- Tab 'Configuration' «̌ Chapter 6.4.1.21.2.30.2 "Tab 'Configuration'" on page 2538

\section*{Watchdog handling in IEC tasks}

If a new project is created or a new task is inserted in the task configuration \(\Leftrightarrow\) Chapter 6.4.1.9.17 "Task Configuration" on page 1914, the task is created with the "default task settings" priority \(=15\) and cycle time \(=10 \mathrm{~ms}\). The watchdog is activated, set to 20 ms and sensitivity \(=1\).


The watchdog handling depends also on the setting of the CPU parameter "Missed cycle behavior":
\begin{tabular}{|c|c|c|c|c|c|}
\hline Parameter - Error LED & \begin{tabular}{l}
Type \\
Enumeration of BYTE
\end{tabular} & Value On & Default Value On & Unit & Descripti Error LED: \\
\hline - Check battery & Enumeration of BYTE & On & On & & Check bat \\
\hline - Stop on error class & Enumeration of BYTE & Diagnosis of at least error class 2 & Diagnosis of at least error class 2 & & Prevents : \\
\hline Diagnosis - Add PLC name to node name & Enumeration of BYTE & Off & Off & & Diagnosis \\
\hline - PLC behaviour after voltage dip & Enumeration of BYTE & Halt & Halt & & Behaviour \\
\hline - Diagnosis history & Enumeration of BYTE & On & On & & Enables / \\
\hline - Max. diagnosis history entries & WORD(0..50000) & 1000 & 1000 & & Max. num \\
\hline - Missed cycle behavior & Enumeration of BYTE & Next & Next & & Behavior I \\
\hline -. Communicaton Schema & Enumeration of BYTE & Default & Default & & Communic \\
\hline
\end{tabular}

This parameter configures the behavior of a real-time task if the processing time of the task is longer than the cycle time.
"Next" means - skip the missed cycle and start the task on the next cycle on time. This might result in skipped tasks, but at least the highest priority task is always started on time, if it is not skipped (= default value).
"ASAP" means - start the task immediately when possible.

This parameter is valid for all real-time tasks (priorities \(0 \ldots 15\) ) of the PLC application.

Example 1: default task settings, \(\mathrm{t}_{\text {Task }}\) - processing time of the task in [ms]
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline \(\mathrm{t}_{\text {Task }}\) & Par. & \multicolumn{9}{|l|}{} \\
\hline 6 & Next & & & & & & & & & \\
\hline & & & - & & & - & , &  & & \\
\hline 12 & ASAP & & & & & & , & & & \\
\hline 12 & Next & & & & & & & & & \\
\hline & & & & & & & & & & \\
\hline
\end{tabular}

No watchdog occurs, also if the processing time of the task is longer than the cycle time (cases 2 and 3 ) since the processing time is shorter than the watchdog time.

Example 2: default task settings, \(\mathrm{t}_{\text {Task }}\) - processing time of the task \(24 \mathrm{~ms}, \mathrm{~S}_{\mathrm{WD}}\) - sensitivity of the watchdog


Watchdog occurs in all 3 cases since the processing time of the task is longer than the watchdog time. According to the setting of the sensitivity the watchdog occurs after 1 or 3 cycles.

Beside the task watchdog there is the so-called "omitted cycle watchdog" (OMCW). The omitted cycle watchdog is only active if a watchdog has been configured for the task.
The "normal" Watchdog triggers only if the processing time of the task exceeds the set Watchdog value.
The omitted cycle watchdog on the other hand checks completely "failed" cycles. E.g. if the scheduler has a problem and the task never executes its cycle again, then the "normal" watchdog will not be triggered. Therefore, the run time does an additional check, if a task has been executed within the double cycle time or the double watchdog time (the bigger of both is valid). If not, the omitted cycle watchdog is triggered.

Example 3: default task settings, \(\mathrm{t}_{\text {Task }}\) - processing time of the task in ms, \(\mathrm{C}_{\text {lost }}\) - lost cycles
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|c|c|}
\hline \(\mathrm{t}_{\text {Task }}\) & \(\mathrm{C}_{\text {lost }}\) & \multicolumn{11}{|l|}{} \\
\hline 6 & \(3 \ldots 6\) & & & & & & & & & & & \\
\hline & & & & & & & & &  & & & \\
\hline 12 & \(3 \ldots 6\) & & & & & & & 0 & & & & \\
\hline \(\infty\) & 2 ff & & & & 詨 & & & & & & & \\
\hline
\end{tabular}

Omitted cycle watchdog occurs after double watchdog time ( \(2 \times 20 \mathrm{~ms}=40 \mathrm{~ms}\) ).

Example 4: Two tasks with following settings:
\begin{tabular}{|l|l|l|l|l|l|l|}
\hline Task & Priority & \begin{tabular}{l} 
Cycle time \\
[ms]
\end{tabular} & \begin{tabular}{l} 
Watchdog \\
time \([\mathrm{ms}]\)
\end{tabular} & Sensitivity & Parameter & \begin{tabular}{l} 
Task pro- \\
cessing \\
time \([\mathrm{ms}]\)
\end{tabular} \\
\hline 1 & 10 & 10 & 20 & 1 & Next & 6 \\
\hline 2 & 15 & 50 & 50 & 1 & & 30 \\
\hline
\end{tabular}


Watchdog of task 2 is triggered since the task cannot run in the defined task cycle.

Example 5: Two tasks with following settings
\begin{tabular}{|l|l|l|l|l|l|l|}
\hline Task & Priority & \begin{tabular}{l} 
Cycle time \\
[ms]
\end{tabular} & \begin{tabular}{l} 
Watchdog \\
time [ms]
\end{tabular} & Sensitivity & Parameter & \begin{tabular}{l} 
Task pro- \\
cessing \\
time \([\mathbf{m s}]\)
\end{tabular} \\
\hline 1 & 10 & 10 & 20 & 1 & ASAP & 12 \\
\hline 2 & 15 & 30 & 60 & 1 & 6 \\
\hline
\end{tabular}


Watchdog of task 2 is triggered since the task cannot start in the defined task cycle.

Example 6: Two tasks with following settings
\begin{tabular}{|l|l|l|l|l|l|l|}
\hline Task & Priority & \begin{tabular}{l} 
Cycle time \\
{\([\mathrm{ms}]\)}
\end{tabular} & \begin{tabular}{l} 
Watchdog \\
time \([\mathrm{ms}]\)
\end{tabular} & Sensitivity & Parameter & \begin{tabular}{l} 
Task pro- \\
cessing \\
time \([\mathrm{ms}]\)
\end{tabular} \\
\hline 1 & 10 & 10 & 20 & 1 & Next & 12 \\
\hline 2 & 15 & 30 & 60 & 1 & 6 \\
\hline
\end{tabular}


No watchdog is triggered, but task 1 is running in 20 ms cycle instead of configured 10 ms cycle. Task 2 is running alternating every 20 ms or 40 ms .

\section*{PLC utilization}

\section*{General}

Non real time
system tasks
(priority 16)

\section*{Non real time \\ IEC tasks \\ (priority 16)}

File access,
OPC UA, etc.

\section*{cpuload}

Real time IEC


\section*{picload}

The parameters cpuload and plcload represent the actual CPU load or PLC load of the system.
- cpuload: This value represents the time the PLC requires to calculate all processes running on the PLC. For a good system performance this value should be less than \(80 \%\). In case of a higher value, the degree of utilization should be reduced by using a more powerful PLC or by reducing the amount of processes.
- plcload: This value represents the time the PLC requires to calculate all real-time processes. Real-time processes are either high priority system tasks or IEC tasks with a priority between 0 and 15. For a good system performance this value should be less than \(60 \%\). In case of a higher value, the degree of utilization should be reduced by using a more powerful PLC.

During commissioning we recommend to monitor the CPU and PLC values online with one of the following methods:

\section*{Automation Builder}
- Commissioning via 'PLC shell commands' \(\Leftrightarrow\) Chapter 6.9.2.4.4 "PLC shell commands" on page 4647 (command 'plcload' and 'cpuload').
- Commissioning via 'Application at runtime' \({ }^{\Perp}\) Chapter 6.4.1.13.3 "Data Recording with Trace" on page 2007. In order to display the load of the CPU or PLC, create a new Device Trace object in your PLC project. Then upload the data into the views 'Command 'Upload Trace' \({ }^{4}\) ) Chapter 6.4.1.21.3.22.19 " Command 'Upload Trace"' on page 2742.

IEC applications/IEC program

To access the parameters plcload and cpuload please use system functions as follows:
- plcload: SchedGetProcessorLoad() included in library 'CmpSchedule’.
- cpuload: SysMCGetLoad() included in library 'SysCpuMultiCore'.

\section*{Managing priorities by selecting the appropriate communication schema}

The AC500 V3 PLCs have an integrated preemptive real-time operating system that supports 100 priorities from 0 (lowest priority) to 99 (highest priority). Hereof \(0 \ldots 49\) in the non-real-time area and 50 ... 99 in the real-time area.

For real-time tasks in the IEC user program 16 priorities 0 (highest priority) ... 15 (lowest priority) can be used. They correspond to the operating system priorities 67 (for IEC task priority 0 ) ... 52 (for IEC task priority 15).

The file system, the memory card and flash tasks run on lowest real-time priority 50.
The non-real-time IEC task priority 16 runs in the non-real-time area. Likewise, all Ethernet protocols and the diagnosis system run in the non-real-time area.

As of Automation Builder 2.4.1 and "SystemFW" 3.4.1 provides a new PLC boot parameter Communication Schema (non-real time vs. real-time Ethernet data) for AC500 V3:
- Name: "Default"

Description: Balanced priority for communication via communication modules (CMs) and onboard Ethernet communication.
๕ Further information on page 4429
- Name: "Communication modules"

Description: Priority and high performance for communication module (CM) based communication via sync tasks. Lower priority for onboard Ethernet and I/O bus.
\& Further information on page 4430
- Name: "Onboard Ethernet"

Description: Priority for onboard Ethernet communication (e. g. via Modbus TCP). Lower priority for communication via communication modules (CMs)
\(\boldsymbol{y}\) Further information on page 4431
- Name: "Realtime onboard Ethernet"

Description: Very high priority for onboard Ethernet communication (e. g. EtherCAT, PROFINET, Ethernet/IP). Low priority for communication via communication modules (CMs)出) Further information on page 4431


The value "Realtime onboard Ethernet" is reserved for later use and has currently the same settings as "Onboard Ethernet" and in addition the I/O bus on the same priority as the Ethernet.

In version of "SystemFW" 3.5.0 the priority of onboard CAN interface has been adapted and is now included in the priority schemas.
In addition the parameter Communication Schema is now also available for the eCo-V3 PLCs.

\section*{The „Default" priority schema in "SystemFW" 3.4.1}

The default value of the boot-parameter Communication Schema is the balanced priority for communication via communication modules (CMs) and onboard Ethernet communication. The following figure gives an overview about the main task priorities in the AC500 V3 PLCs.


The default setting of the priorities is suitable for most applications and corresponds to the settings in the firmware versions 3.4.0 and before.

The "Communication modules" priority schema in "SystemFW" 3.4.1
The communication modules priority schema has been established for priority and high performance for communication module (CM) based communication via sync tasks. Lower priority for onboard Ethernet and I/O bus.


The priority schema communication module should be used in applications with one or more communication modules CM5xx with sync mode. As of Automation Builder 2.4.1 these are the CM579-ETHCAT EtherCAT master and CM598-CN CANopen master communication modules.

The sync task with priority 0 will be interrupted only by system interrupts. Since the IEC scheduler observer task is located below IEC priority 0 , the watchdog for this task is also ineffective. However, this should not interfere with a sync task.

If more than one CM 5 xx are used in sync mode, the priority order must be defined. The sync task of highest priority CM5xx receives IEC priority 0 , the next priority 1 and so on.

In a mixed PLC configuration with communication modules with and without sync mode the interrupts of the communication modules without sync mode will be handled on the priority of the lowest sync task. Currently supported communication modules without sync mode are CM579-PNIO PROFINET IO controller and SM560-S Safety PLC.

\section*{The "Onboard Ethernet" priority schema in "SystemFW" 3.4.1}

The "Onboard Ethernet" priority schema has been established for priority for onboard Ethernet communication (e. g. via Modbus TCP).


This priority schema should be used for applications with much Ethernet communication, e. g. Modbus TCP communication with a high number of Modbus TCP clients/servers.

\section*{NOTICE!}

Since the Ethernet interrupt task is running in this mode in the real-time priority area, the Ethernet communication can block IEC tasks with priorities 12-15.

Working with real-time priority at onboard Ethernet and using a high number of Modbus TCP client connections can force high CPU load. To avoid this, it is recommended to call the Modbus function blocks in steps.

Example
100 Modbus TCP client connections shall be used in a 20 ms task.
Call 20 function blocks in a first cycle, 20 function blocks in a second cycle and so on.


\section*{The „Default" priority schema in "SystemFW" 3.5.0}

The default value of the boot-parameter Communication Schema is the balanced priority for communication via communication modules (CMs) and onboard Ethernet communication.

The following figure gives an overview about the main task priorities in the AC500 V3 midrange PLCs in version "SystemFW" 3.5.0.
\begin{tabular}{|c|c|}
\hline  & \begin{tabular}{l}
The highest IEC task priority 0 should be used for high prior functions in the PLC with short execution time. \\
The communication modules interrupts (CM5xx), the I/O bus, the IEC scheduler observer task and the CAA event task are processed with higher priority than all user IEC tasks. \\
Ethernet runs on non-real-time priority. \\
The CAN onboard interrupt is moved from non-real-time priority to real-time priority 69, one priority above the CAN transmit and receive threads.
\end{tabular} \\
\hline
\end{tabular}

The default setting of the priorities is suitable for most applications and corresponds to the settings in the firmware versions 3.4.0 and before.

\section*{The "Communication modules" priority schema in "SystemFW" 3.5.0}

The communication modules priority schema has been established for priority and high performance for communication module (CM) based communication via sync tasks. Lower priority for onboard Ethernet and I/O bus.


The highest IEC task priority 0 should be used for the sync task of the highest priority communication module, e. g. CM579-ETHCAT, priority 1 for the sync task with second highest priority, and so on.
Only the communication modules interrupts (CM5xx) and the CAA Event task is processed with higher priority than the IEC tasks.

The IEC scheduler observer task has been moved to priority 66, means below IEC task priority 0 and on the same level as IEC task priority 1.
The priority of the I/O bus has been moved to priority 59 , means inside the IEC task priority area, but below the external event task priorities.

The CAN onboard interrupt has the same priority as the I/O bus and the CAN transmit and receive threads one priority below, means 58.
Ethernet runs on non-real-time priority.

The priority schema communication module should be used in applications with one or more communication modules CM5xx with sync mode. As of Automation Builder 2.4.1 these are the CM579-ETHCAT EtherCAT master and CM598-CN CAN master communication modules.

The sync task with Prio 0 will be interrupted only by system interrupts. Since the IEC scheduler observer task is located below IEC Prio 0, the watchdog for this task is also ineffective. However, this should not interfere with a sync task.
If more than one CM5xx are used in sync mode, the priority order must be defined. The sync task of highest priority CM5xx receives IEC Prio 0, the next Prio 1 and so on.
In a mixed PLC configuration with communication modules with and without sync mode the interrupts of the communication modules without sync mode will be handled on the priority of the lowest sync task. Currently supported communication modules without sync mode are CM579-PNIO PROFINET IO controller and SM560-S safety PLC.

\section*{The "Onboard Ethernet" priority schema in "SystemFW" 3.5.0}

The "Onboard Ethernet" priority schema has been established for priority for onboard Ethernet communication (e. g. via Modbus TCP).


This priority schema should be used for applications with much Ethernet communication, e. g. Modbus TCP communication with a high number of Modbus TCP clients/servers.

\section*{NOTICE!}

Since the Ethernet interrupt task is running in this mode in the real-time priority area, the Ethernet communication can block IEC tasks with priorities 12-15.

Working with real-time priority at Onboard Ethernet and using a high number of Modbus TCP client connections can force a high CPU Load. To avoid this, we recommend calling the Modbus FB's in steps.

100 Modbus TCP client connections shall be used in a 20 ms task.
Call 20 function blocks in a first cycle, 20 function blocks in a second cycle and so on.
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline Cycle & 1 & 2 & 3 & 4 & 5 & X \\
\hline 1 & \multicolumn{6}{|l|}{} \\
\hline 2 & \multicolumn{6}{|l|}{} \\
\hline 3 & \multicolumn{6}{|l|}{} \\
\hline & -. & -- & .. & -. & -. & -. \\
\hline Y & \multicolumn{6}{|l|}{} \\
\hline
\end{tabular}

\section*{Communication schema onboard Ethernet/EtherCAT}

\section*{General}

Due to the broad variety of functionality which can run in parallel on AC500 V3 PLCs, it is recommended, or even mandatory depending on the desired functionality, to adjust system behavior to better match the needs of the individual application.

Due to hardware limitations and constraints, it's necessary to prioritize functionality, especially if it's intended to execute multiple features in parallel. Prioritizing can be very difficult due to the complexity of the underlying software components. Besides of task priorities, it might also be necessary to adjust low level system configuration to achieve the best possible performance for a specific feature.
To reduce complexity, and thus the risk of negative side-effects the CPU parameter Communication schema has been introduced. This parameter enables the user to specifically define the primary purpose of the corresponding PLC and lay focus on a specific functionality.
The following table shows the available communication schemas for both, PM50xx eCo and PM56xx PLCs.
\begin{tabular}{|c|c|c|c|}
\hline Schema & PM50xx & PM56xx & Description \\
\hline \begin{tabular}{l}
Default \\
ヶ Chapter 6.8.2.2.1.8.2 \\
"Communication schema \\
"Default"" on page 4438
\end{tabular} & X & X & Balanced priority for communication via communication modules (CMs) and onboard Ethernet communication. \\
\hline \begin{tabular}{l}
Communication modules \\
\({ }^{4}\) Chapter 6.8.2.2.1.8.3 \\
"Communication \\
schema "Communication \\
modules" " on page 4439
\end{tabular} & - & X & Priority and high performance for communication module (CM) based communication via sync tasks. Lower priority for onboard Ethernet and I/O bus. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|l|l|}
\hline Schema & PM50xx & PM56xx & Description \\
\hline \begin{tabular}{l} 
Onboard Ethernet \\
e Chapter 6.8.2.2.1.8.4 \\
"Communication schema \\
"Onboard Ethernet"' \\
on page 4441
\end{tabular} & X & X & \begin{tabular}{l} 
Priority and high performance for com- \\
munication module (CM) based com- \\
munication via sync tasks. Lower pri- \\
ority for onboard Ethernet and I/O bus.
\end{tabular} \\
\hline \begin{tabular}{l} 
Onboard EtherCAT \\
( Chapter 6.8.2.2.1.8.5 \\
"Communication schema \\
"Onboard EtherCAT" \\
since "SystemFW" 3.4.1" \\
on page 4442
\end{tabular} & X & & X \\
\hline
\end{tabular}

Changing communication schema might have significant impact on the behavior of the user application and should therefore only be done if,
- a feature is configured which relies on a specific schema (e.g., onboard EtherCAT).
- there's need of optimization of system behavior in terms of performance (decrease of reaction times, increase of throughput etc.).
Note that the increase of flexibility and freedom comes with an increase of responsibility because IEC task configuration and IEC user code might block important system functionality for too long. The following provided information should help to identify appropriate settings to prevent runtime errors like timeout exceptions or communication problems.

Priority classifi- The functionality prioritized by each communication schema is grouped into 3 different classes:
cations and their purpose
- High priority real-time functionality
- IEC tasks with real-time priority
- Low priority non-real-time functionality


Functionality with strict timing requirements, e.g., drivers with hardware access or hardware watchdog monitoring failures might lead to exception errors, loss of communication with hardware and other problems which are non-recoverable and might cause a stop of the user application.
Functionality of this class can't be influenced by activity of user defined IEC tasks. As a result of this, related functionality can interrupt or delay IEC tasks and therefore increase IEC task jitter and the duration of IEC task cycles.

IEC tasks with This is the priority range of user defined real-time IEC tasks. Priorities 0 to 15 are real-time real-time priority priorities (IEC priority 16 is mapped to non-real-time priority) and can be used to define an IEC task's priority in Automation Builder.
Task in this range can be interrupted or delayed by high priority functionality. It's not possible for IEC user code to suppress high priority activity, thus increasing safety and reducing the risk of negative side effects caused by bugs in IEC user code or libraries or misconfiguration of IEC tasks.

\section*{Low priority \\ non-real-time \\ functionality}

The low priority range is intended for non-real-time functionality which doesn't have any specific timing requirements and which is supposed to execute its activity without delaying or interrupting the user application and its related functionality.
Activity assigned to this range can be suppressed by high priority and real-time IEC tasks. Therefore might show high jitter, sporadic activity or large reaction times, depending on the overall system load.
To avoid suppression of the tasks of this group, the system reserves a short time slot to process low-priority functionality if the CPU load reaches nearly \(100 \%\). Under normal circumstances (below an average system load of 80\%), this mechanism shouldn't be engaged. Although this time slot could cause high priority tasks to jitter slightly ( \(\pm 100\) microseconds), it ensures that the system remains responsive and will always be able to run fundamental jobs which must be executed regularly. Note that tasks of low priority still play an important role and need to be scheduled regularly to meet their individual timing requirements.

\section*{Ensure that the average CPU load doesn't exceed 80\% to ensure that there's} enough time for the system to process low-priority functionality.
The functionality might not be important for the individual application, but it is still necessary to keep alive other functionality like the Automation Builder login or access to diagnosis data.
Once the CPU load exceeds \(80 \%\), it will be difficult for the system to correctly schedule tasks.

\section*{Communication schema "Default"}

Communication schema "Default" is the default schema selected when creating a new AC500 V3 application. It is the recommended setting to start with and suitable for most applications.

The schema leads to a well balanced task priority setup which lays focus on stability and low risk of negative side effects caused by high IEC task load or hardware activity.

The highest IEC task priority 0 should be used for high prior functions in the PLC with short execution time.
The communication modules interrupts (CM5xx), the I/O bus, the IEC scheduler observer task and the CAA event task are processed with higher priority than all user IEC tasks.
Ethernet runs with non-real-time priority to avoid denial-of-service scenarios and system overload caused by devices located in the same network.

The default setting of the priorities is suitable for most applications and should be used unless there are special requirements or performance issues.

Features and In this schema, the range of IEC task priorities is only shared with tasks related to CM598-CN their priorities (CAN communication module) and its CAN2A/2B IO POUs and common POU background job handling. The priorities 9 and 15 can still be used though.

Note that tasks above IEC task priority 9 might have negative side effects on CAN communication in terms of reaction time and throughput, while IEC tasks with priority lower than 9 might be interrupted or delayed by an active CAN communication.

POU background jobs are assigned the lowest IEC real-time priority to avoid negative side effects on IEC tasks and thus on process data synchronization or user code. POUs usually are related to tasks of lower priority and shall not have an impact on higher priority tasks.


\section*{Communication schema "Communication modules"}

This communication schema is only available for PM56xx series AC500 V3
PLCs supporting communication modules.

The communication schema "Communication modules" is intended for users of a communication module which can be configured to synchronize its stack activity with the user application by a dedicated (external event triggered IEC task) which is mapped to an event ("CouplerEventX", \(X=\) slot number of the communication module).
The bus synchronized IEC task is not scheduled by the PLC runtime system but the stack of the communication module. As a result of this, it's necessary for the PLC to quickly react on the related event of the communication module. The schema ensures that negative side-effects on the bus synchronized IEC task are reduced as much as possible by adjusting priorities of high priority functionality.

Features and The following figure shows the changes the schema applies to the available features of the their priorities PLC. The primary goal of this schema is to reduce the number of tasks which are capable of interrupting or delaying the bus synchronized IEC task, while providing as much flexibility as possible. By making use of the priority range of IEC tasks, users can decide whether existing IEC tasks will influence the corresponding features.
- Use IEC priority 0 for the bus synchronized IEC task and a priority lower than 0 . For all other features to avoid negative side-effects on the bus synchronized IEC task.
- Use IEC priorities 1 to 5 for functionality which is more important than high priority features like IO bus or CAN.
- To ensure that a high priority feature is not influenced by any IEC user code (besides of the bus synchronized IEC task), use an IEC task priority which is lower than the priority of the corresponding feature.

It is not forbidden to make use of IEC task priorities which are assigned to other PLC functionality.


\section*{NOTICE!}

Be careful when using this schema to avoid unwanted side-effects on features besides of the protocol realized by the corresponding communication module, caused by violation of individual timing constraints or requirements!
I/O bus must be able to refresh all connected S500 I/O devices within 20 ms . Otherwise, an exception will be raised which will lead to a stop of the application. Use PLC shell command io-bus desc to dump information on the bus timing. The output contains information on the IO bus cycle time:
--- I/O bus information -
Baud rate [baud]: 1714286
Min. cycle time [us]: 1037
Max. cycle time [us]: 10936
Last cycle time [us]: 1567
y Chapter 6.4.1.21.2.8.11 "Tab 'PLC Shell"' on page 2441
The maximum cycle time must be below 20 ms to ensure that the system runs stable and won't raise a timeout exception error.

If the cycle time reaches the limit, try to reduce the amount of code executed by IEC user tasks of higher priority than I/O bus or adjust task priorities if possible.

\section*{Communication schema "Onboard Ethernet"}

This schema is intended for users who make use of Ethernet protocols which rely on a lot of parallel network connections, like Modbus TCP or IEC 61850.

\section*{Features and} their priorities

As shown in the diagram below, Ethernet related tasks will be assigned a priority within IEC task priority range to reduce reaction times and increase throughput to meet the requirements of protocols like Modbus TCP or IEC 61850. By making use of IEC task priority 8, users can decide whether their IEC user tasks are capable of interrupting or delaying Ethernet communication or not, depending on the individual requirements.


\section*{Communication schema "Onboard EtherCAT" since "SystemFW" 3.4.1}

Schema "Onboard EtherCAT" is intended for users who intend to make use of the EtherCAT master protocol, configured at one of the available onboard Ethernet ports of the corresponding AC500 V3 PLC.

Due to the special requirements of EtherCAT, it's necessary to modify the configuration of system components related to onboard Ethernet, and thus provide a low-latency Ethernet port. In addition to that, priorities of various functionality are adjusted to reduce negative side-effects on the EtherCAT IEC task and achieve best possible performance.

Features and their priorities

The following diagram shows the changes the schema applies to the available features of the PLC. The primary goal of this schema is to reduce the number of tasks which are capable of interrupting or delaying the Ethernet IEC task, while providing as much flexibility as possible. By making use of the priority range of IEC tasks, users can decide whether existing IEC tasks will influence the corresponding features.
- Use IEC priority 0 for the EtherCAT IEC task and a priority lower than 0 for all other features to avoid negative side-effects on EtherCAT.
- Use IEC priorities 1 to 5 for functionality which is more important than high priority features like IO bus or onboard I/O.
- To ensure that a high priority feature isn't influenced by any IEC user code (besides of EtherCAT), use an IEC task priority which is lower than the priority of the corresponding feature.


It is not forbidden to make use of IEC task priorities which are assigned to other PLC functionality.


\section*{NOTICE!}

Be careful when using this schema to avoid unwanted side-effects on features other than EtherCAT, caused by violation of individual timing constraints or requirements!

I/O bus must be able to refresh all connected S500 I/O devices within 20 ms . Otherwise, an exception will be raised which will lead to a stop of the application. Use PLC shell command io-bus desc to dump information on the bus timing. The output contains information on the IO bus cycle time:
--- I/O bus information -
Baud rate [baud]: 1714286
Min. cycle time [us]: 1037
Max. cycle time [us]: 10936
Last cycle time [us]: 1567
药 Chapter 6.4.1.21.2.8.11 "Tab 'PLC Shell"" on page 2441
The maximum cycle time must be well below 20 ms to ensure that the system runs stable and won't raise a timeout exception error.
If the cycle time reaches the limit, try to reduce the amount of code executed by IEC user tasks of higher priority than I/O bus or adjust task priorities if possible.

Low latency Besides the adjustment of task priorities, schema "Onboard EtherCAT" also changes the low port level Ethernet driver configuration to offer a dedicated low latency port to the user.
Due to technical constraints, the port number of the low latency port depends on the type of PLC which is used:
\begin{tabular}{|l|l|l|}
\hline PLC type & ETH1 & ETH2 \\
\hline PM50xx eCo V3 & Low latency & Maintenance and visualization \\
\hline PM56xx V3 & Maintenance and visualization & Low latency \\
\hline
\end{tabular}

The other port remains on low priority to minimize the impact on the activity of the v protocol. It is intended for login (Automation Builder), visualization services or similar (non-real-time) activities.

When configuring eCo V3 CPU "Onboard EtherCAT" parameters, the ETH1 interface is set to the IP address 0.0.0.0. After switching to a different communication scheme than "Onboard EtherCAT", IP configuration of the Ethernet port used for Onboard EtherCAT remains invalidated (IP address 0.0.0.0). It needs to be re-applied by the user should the port be required by the new application.
The "Onboard EtherCAT" functionality is only usable/visible for the eCo V3 CPU from SystemFW 3.7.0.

\subsection*{6.8.2.2 . Setting standard configuration}

If the target setting configuration is changed, standard configuration can be restored:
1. Open CODESYS.
2. In the "Resources" tab, double-click "PLC Configuration".
3. Select "Menu Extras \(\rightarrow\) Standard Configuration".

\subsection*{6.8.2.3 Real-time clock and battery}

\subsection*{6.8.2.3.1 Real-time clock}

Notes on realtime clock

The real-time clock is an optional function for AC500-eCo V3 Basic processor modules (e.g. PM5012-x-ETH) require a TA5131-RTC. All other AC500-eCo V3 processor modules have an integrated real-time clock.

The real-time clock operates as a PC clock. It saves date and time to a DWORD in DT format (DATE AND TIME FORMAT), i.e., in seconds passed since the start time: 1 January 1970 at 00:00.

For AC500-eCo V3, Basic CPU with TA5131-RTC buffers the real-time clock for 7 days, and Standard/Pro CPU buffers the integrated real-time clock for 20 days. When the CPU is not powered over the buffering time, the real-time clock data will be cleared.
If a battery is connected and full, the real-time clock continues to run even if the control voltage is switched off.

If no battery is inserted or the battery is empty, the real-time clocks starts with the value 0 (=1970-01-01, 00:00:00).

When switching on the control voltage, the system clock of the operating system is set to the value of the real-time clock.

Real-time clock The PLC browser/PLC shell commands date and time are used to set the real-time clock. with PLC browser

The commands date <ENTER> or time <ENTER> display the current date and time of the real-time clock.

The command: date yyyy-mm-dd<ENTER> (year-month-day) sets the date.
The command: time hh-mm-ss<ENTER> (hours-minutes-seconds) sets the time.

\section*{Example:}

The real-time clock should be set to 22 February 2005, 16:50.
1. Enter the date:
```

date 2005-02-22<ENTER>
=> Display:date 2005-02-22 Clock set to 2005-02-22 08:01:07

```
The time remains unchanged.
2. Enter the time:
time 16:50<ENTER>
\(\Rightarrow\) Display: time 16:50 Clock set to 2005-02-22 16:50:00

Real-time clock The following function blocks located in the folder "Realtime clock" of the system library with user pro- \(\quad A B B \_E x t U t i l s \_A C 500\).lib can be used to set and display the real-time clock (RTC) with help gram of the user program:
\begin{tabular}{|l|l|}
\hline Function block & Function \\
\hline CLOCK (V3) "Library Manager \(\rightarrow\) ABB-AC500 & Sets and displays the real-time clock with \\
\(\rightarrow\) Use Cases \(\rightarrow\) AC500 Utils & values for year, month, day, hours, minutes \\
\(\rightarrow\) PM<Version> (ABB) \(\rightarrow\) Function Blocks & and seconds. \\
\(\rightarrow\) Realtime clock" & Also the day of week is indicated (Mo=1, \\
& Tue=2, Wed=3, Thu=4, Fr=5, Sa=6, Su=0). \\
& \begin{tabular}{l} 
Note: The week of day cannot be set. It \\
\\
\\
\\
\\
\\
\\
is given by the real-time clock. The input \\
DAY_SET is ignored.
\end{tabular} \\
\hline \begin{tabular}{l} 
CLOCK_DT (V3) "Library Manager \\
\(\rightarrow\) ABB-AC500 \(\rightarrow\) Use Cases \(\rightarrow\) AC500 Utils \\
\(\rightarrow\) PM<Version> (ABB) \(\rightarrow\) Function Blocks \\
\(\rightarrow\) Realtime clock"
\end{tabular} & \begin{tabular}{l} 
Sets and displays the real-time clock in DT \\
format, for example DT\#2005-02-17-17:15:00.
\end{tabular} \\
\hline
\end{tabular}

Reference for function blocks, functions, structures etc. \({ }^{\Downarrow}\) Chapter 6.5.14 "Reference, function blocks" on page 4086

\subsection*{6.8.2.3.2 AC500 battery}

The AC500 battery buffers the following data in case of "control voltage off":
- Retentive variables in SRAM (VAR_RETAIN..END_VAR) « Chapter 6.8.2.1 "Handling of remanent variables for AC500 V3 products" on page 4412
- Date and time of the real-time clock

Further information:
- \({ }^{4}\) Chapter 6.9.3 "Diagnosis messages" on page 4655

To prevent data loss when using the AC500 battery, the battery status should be periodically monitored by the user program.

Battery status The battery status can be monitored either with the help of a user program on the PLC or in Automation Builder.

In the PLC shell of Automation Builder the command "BATT" can be used. \({ }^{\circ}\) Chapter 6.9.2.4.4 "PLC shell commands" on page 4647. The following is output:
\begin{tabular}{|l|l|}
\hline 0 & Battery empty \\
\hline 20 & Remaining battery charge below \(20 \%\) \\
\hline 100 & Battery charge OK \\
\hline
\end{tabular}

In the user program, the battery status can be checked with the function BATT which is available in the folder "Battery" of the system library ABB_ExtUtils_AC500.lib ("Library Manager \(\rightarrow\) ABB-AC500 \(\rightarrow\) Use Cases"). The following is output:
\begin{tabular}{|l|l|}
\hline 0 & Battery empty \\
\hline 20 & \begin{tabular}{l} 
Remaining battery charge below \(20 \%\), battery \\
must be replaced
\end{tabular} \\
\hline 100 & Battery charge OK \\
\hline
\end{tabular}

On the device, the battery status can be checked with the function keys of a processor module.
\(>\) Chapter 6.8.2.5 "Display, LEDs and function keys on the front panel" on page 4452
y Chapter 6.8.2.5.5.3 "VAL - reading out state values" on page 4459

\subsection*{6.8.2.3.3 AC500-eCo V3 data buffering}

The AC500-eCo V3 buffers the following data in case of "control voltage off":
- Retentive variables in FLASH (VAR_RETAIN..END_VAR) \(\left.{ }^{*}\right\rangle\) Chapter 6.8.2.1 "Handling of remanent variables for AC500 V3 products" on page 4412
- Date and time of the real-time clock are using an integrated gold-capacitor with a lower retention time as a battery.


The AC500-eCo V3 has no battery but stores the remanent data in flash or the real-time clock using a gold-capacitor, there is no battery or gold-capacitor status or survey.
In case of "control voltage off", the real-time clock is buffered for about 7 days for Basic CPU with TA5131-RTC and about 20 days for Standard or Pro CPUs at \(40^{\circ} \mathrm{C}\) using temperature.

\subsection*{6.8.2.4 AC500-eCo V3 processor module, LEDs, RUN/STOP switch on front panel}

\subsection*{6.8.2.4.1 Features}


Fig. 358: Example: PM5072-T-2ETH
Micro memory card slot
5 LEDs to display the states of the processor module (Power, Error, Run, MC, MOD1) RUN button
RJ45 female connector for Ethernet1 connection
RJ45 female connector for Ethernet2 connection (available for PM50x2-T-2ETH)
3-pin terminal block for power supply 24 V DC 2 holes for screw mounting
Option board slot cover for option board slot (the number of available slots varies according to the CPU type)
9 Cable fixing
10 13-pin terminal block for onboard I/Os
1 12-pin terminal block for onboard I/Os (not available on PM5012-x-ETH)
1212 LEDs to display the states of the signals
1310 LEDs to display the states of the signals
14 Cable fixing accessory TA5301-CFA on the top of the housing (optional)

The processor module is shown with pluggable terminal blocks. These terminal blocks must be ordered separately.

The cable fixing accessory on the top of the housing is optional.
Please use TA5301-CFA cable fixing accessory to provide strain relief.
It can also be used for AC500-eCo I/O modules.

\section*{The PM50x2 processor modules are supplied with option board slot covers as standard.}

There are various TA51xx option boards for the processor modules that can be ordered separately.

Which and how many option boards can be plugged, depends on the respective processor module.

\subsection*{6.8.2.4.2 State LEDs and operating elements}

RUN/STOP The processor modules, PM50xx series, have a RUN/STOP button. By pressing the RUN/STOP button button, the processor modules switch between RUN mode and STOP mode. By long-pressing RUN/STOP button during the processor module power on phase, the processor module will be in MOD1.

State LEDs The processor modules PM50xx indicate their states of operation via 5 LEDs located on the upper left side of the processor module.
\begin{tabular}{|c|c|c|c|c|c|}
\hline LED & State & Color & LED = ON & LED = OFF & LED flashing \\
\hline PWR & Power supply & Green & Power supply present & Power supply missing & - \\
\hline MC & Micro memory card indication & Yellow & Micro memory card is in the socket & Micro memory card is not in the socket & Micro memory card is in read/ write state: any file on card is opened, means activity on card \\
\hline ERR & Error indication & Red & An error occurred & \begin{tabular}{l}
No errors or only warnings encountered (E4 errors). \\
The LED behavior for the error classes 2 to 4 is configurable.
\end{tabular} & Fast flashing (4 Hz ) displays together with the RUN LED a currently running firmware-upgrade or writing data to the FlashEPROM. Slow flashing ( 1 Hz ) alone displays shutdown of Request To Send. Medium flashing ( 2 Hz ) alone displays at start of PLC if reboot after watchdog. \\
\hline MOD1 & Mode 1 indication & Yellow & Processor module is in mode 1 state & Processor module is not in mode 1 state & - \\
\hline
\end{tabular}
\begin{tabular}{|l|l|l|l|l|l|}
\hline LED & State & Color & LED = ON & LED = OFF & LED flashing \\
\hline RUN & RUN/STOP state & Green & \begin{tabular}{l} 
Processor \\
module is in state \\
RUN
\end{tabular} & \begin{tabular}{l} 
Processor \\
module is in state \\
STOP
\end{tabular} & \begin{tabular}{l} 
Fast flashing (4 \\
Hz): \\
The processor \\
module is \\
reading/writing \\
data from/to the \\
memory card. \\
If the ERR-LED \\
is also flashing, \\
data is being \\
written to the \\
Flash-EPROM.
\end{tabular} \\
& & & & & \\
\hline
\end{tabular}

User configurable LEDs

The AC500-eCo V3 processor module also provides 2 LEDs below the state LEDs which can be used by user and driven by an application.
The LEDs can be used into a project and controlled using special function blocks which are contained in the PM AC500 library. The POU is PmLedSet located in folder LED control.

I/O LEDs

The processor module provides up to 10 LEDs (PM5012-x-ETH), 20 LEDs (PM5032-R-ETH, PM5052-R-ETH), or 22 LEDs (PM5032-T-ETH, PM5052-T-ETH, PM5072-T-2ETH(W), PM5082-T-2ETH) to display the states of the inputs and outputs.
\begin{tabular}{|l|l|l|l|l|l|}
\hline \begin{tabular}{l} 
Processor \\
module
\end{tabular} & LED & State & Color & LED = ON & LED = OFF \\
\hline \multirow{3}{*}{ PM5012-x-ETH } & I0 ... I5 & Digital input & Yellow & Input is ON & Input is OFF \\
\cline { 2 - 6 } & O0 ... O3 & \begin{tabular}{l} 
Transistor \\
output
\end{tabular} & Yellow & Output is ON & Output is OFF \\
\hline
\end{tabular}
\begin{tabular}{|l|l|l|l|l|l|}
\hline \begin{tabular}{l} 
Processor \\
module
\end{tabular} & LED & State & Color & LED = ON & LED = OFF \\
\hline & NO0 ... NO3 & Relay output & Yellow & Output is ON & Output is OFF \\
\hline \multirow{5}{*}{\begin{tabular}{l} 
PM5032-x-ETH \\
PM5052-x-ETH
\end{tabular}} & IO ... I11 & Digital input & Yellow & Input is ON & Input is OFF \\
\cline { 2 - 6 } & O0 ... O7 & \begin{tabular}{l} 
Transistor \\
output
\end{tabular} & Yellow & Output is ON & Output is OFF \\
\cline { 2 - 6 } & NO0 ... NO5 & Relay output & Yellow & Output is ON & Output is OFF \\
\cline { 2 - 6 } & C12, C13 & \begin{tabular}{l} 
Digital configu- \\
rable input/ \\
output
\end{tabular} & Yellow & \begin{tabular}{l} 
Input/Output \\
is ON
\end{tabular} & \begin{tabular}{l} 
Input/Output is \\
OFF
\end{tabular} \\
\hline \begin{tabular}{l} 
PM5072- \\
T-2ETH(W) \\
PM5082-T-2ETH
\end{tabular} & IO ... I11 & Digital input & Yellow & Input is ON & Input is OFF \\
\cline { 2 - 6 } & O0 \(\ldots\) & \begin{tabular}{l} 
Transistor \\
output
\end{tabular} & Yellow & Output is ON & Output is OFF \\
\cline { 2 - 6 } & C12, C13 & \begin{tabular}{l} 
Digital configu- \\
rable input/ \\
output
\end{tabular} & Yellow & \begin{tabular}{l} 
Input/Output \\
is ON
\end{tabular} & \begin{tabular}{l} 
Input/Output is \\
OFF
\end{tabular} \\
\hline
\end{tabular}

\section*{Ethernet state LEDs}

Table 810: State LEDs at Ethernet connector
\begin{tabular}{|l|l|l|l|l|}
\hline LED & Color & OFF & ON & Flashing \\
\hline Activity & Yellow & No activity & --- & Activity \\
\hline Link & Green & No link & Link & --- \\
\hline
\end{tabular}

\subsection*{6.8.2.5 Display, LEDs and function keys on the front panel}

\subsection*{6.8.2.5.1 Overview}


The display of a processor module is equipped with a background-lighted 7-segment display. This display consists of 6 digits for plain text or error codes.

Some functionalities may be not yet supported by the product. Please refer to the release notes of the product at time of release.

Display indicators
- A black square ( \(\square\) ) denotes the state/working activity of the corresponding object on the left/right side of the display. The black square flashes according to the device's activity, e.g. during data exchange on ETH1, ETH2, COM1, etc.

\section*{MC activity}

For the activity of the memory card the black square ( is shown as long as a file is open on memory card.
- A black triangle ( ) points to the selected item/interface on the left/right side of the display to be configured or read. Further, it acts as a cursor for the count up/count down function keys.


The indicators point to the following items on the left side of the display:
\begin{tabular}{|l|l|l|}
\hline No. & On the left side & Description \\
\hline 1 & MC (memory card) & Refers to the memory card state. \\
\hline 2 & SYS (system) & Refers to the system state. \\
\hline 3 & BATT (battery) & Refers to the battery state. \\
\hline 4 & I/O bus & Refers to I/O bus connection. \\
\hline
\end{tabular}

The indicators point to the following items on the right side of the display:
\begin{tabular}{|l|l|l|}
\hline No. & On the right side & Description \\
\hline 5 & ETH1 & Refers to the first Ethernet interface. \\
\hline 6 & ETH2 & Refers to the second Ethernet interface. \\
\hline 7 & COM1 & Refers to COM1 interface. \\
\hline 8 & CAN & Refers to CAN interface. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline 9 & Function keys on front panel \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|}
\hline Processor module & Display variant & Description \\
\hline PM56xx-2ETH &  & Display of a processor module with support for 2 Ethernet interfaces, CAN and COM1. \\
\hline
\end{tabular}

\subsection*{6.8.2.5.2 Text outputs of the display}
\begin{tabular}{|c|c|}
\hline Display & Description \\
\hline  & Display on system start (power on). \\
\hline  & PLC is in boot mode. \\
\hline  & \begin{tabular}{l}
Is shown on startup after boot, when a wrong DisplayFW is detected, e.g. an old version. \\
Update display with DisplayFW 4.1 (or higher). \\
๕ Chapter 6.3.1.4.3 "Installation and update of the AC500 V3 firmware" on page 1422
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|r|l|l|}
\hline Display \\
\hline
\end{tabular}

\subsection*{6.8.2.5.3 Startup procedure of the PLC}

Startup procedure of a new PLC from factory
\begin{tabular}{|c|c|c|}
\hline State & Display & Description \\
\hline 0 &  & Display on system start (power on). \\
\hline 1 &  & PLC is in boot mode. \\
\hline 2 & (rity & PLC is in initialization mode. \\
\hline 3 &  & \begin{tabular}{l}
No system firmware (SystemFW) available. \\
Start update firmware. \\
PLC is waiting for a firmware download via Automation Builder or memory card. \\
を Chapter 6.3.1.4.3 "Installation and update of the AC500 V3 firmware" on page 1422
\end{tabular} \\
\hline
\end{tabular}

\section*{Startup procedure of a PLC with system firmware}

The startup procedure depends on the selected PLC mode.
\begin{tabular}{|c|c|c|}
\hline PLC mod e & Display & Startup behavior \\
\hline 00 & svs Mad Tiom & The user program will be loaded and run. PLC changes to mode „RUN". \\
\hline 01 &  & User program will not be loaded/run. PLC stay in mode „STOP". \\
\hline 02 &  & Reserved for further development (currently like mode 00). \\
\hline
\end{tabular}

Mode 01 can be activated via function key CFG, or by pressing function key RUN during startup of PLC until mode 01 is shown on display \& Further information on page 4453.

If an available update or a hotfix is displayed in the Automation Builder, it can be installed as described in the "Automation Builder update notification" chapter. Hy Chapter 4.8.3 "Execution of a hotfix" on page 151
\begin{tabular}{|c|c|c|}
\hline State & Display & Description \\
\hline 0 &  & Display on system start (power on). \\
\hline 1 &  & PLC is in boot mode. \& Further information on page 4453 \\
\hline 2 & (rites & PLC is in initialization mode. \(\leftrightarrows\) Further information on page 4453 \\
\hline 3 & (ions & \begin{tabular}{l}
PLC is in STOP mode. \(\Leftrightarrow\) Further information on page 4453 \\
Same as state Stop in Automation Builder.
\end{tabular} \\
\hline 4 &  & \begin{tabular}{l}
PLC is in RUN mode. \(\Leftrightarrow\) Further information on page 4453 \\
Switch into RUN mode is only possible if a valid boot project is available in the flash memory. « Chapter 6.3.1.4.5.2 "Firmware and/or application installation and update" on page 1427
\end{tabular} \\
\hline 5 &  & \begin{tabular}{l}
Only in RUN mode and as of SystemFW V3.2.0 \\
Reminder: demo license \\
PLC runs in „Demo mode", since at least one feature license is missing. \\
Will be displayed for 5 minutes at every license check \\
If „Demo time" expires, PLC will go to "Stop".
\end{tabular} \\
\hline 6 & \[
\mathfrak{c}
\] & \begin{tabular}{l}
Only in RUN mode and as of SystemFW V3.2.0 \\
10 minutes step reminder: license was removed \\
PLC runs in „Grace mode", since at least one feature license which has been available disappeard. PLC is waiting for this license. \\
Will be displayed for 5 minutes \\
If „Grace time" expires, PLC will go to „Stop".
\end{tabular} \\
\hline
\end{tabular}

\subsection*{6.8.2.5.4 Description of LEDs}

The LEDs below the display indicate the state of the processor module:
\begin{tabular}{|l|l|l|l|l|l|}
\hline LED & State & Color & LED = ON & LED = OFF & LED flashes \\
\hline \begin{tabular}{l} 
Power LED \\
(PWR)
\end{tabular} & \begin{tabular}{l} 
Denotes the power \\
supply state of the \\
processor module
\end{tabular} & Green & \begin{tabular}{l} 
Voltage is present \\
(24 V DC)
\end{tabular} & \begin{tabular}{l} 
Voltage is \\
missing
\end{tabular} & - \\
\hline \begin{tabular}{l} 
Run LED \\
(RUN)
\end{tabular} & \begin{tabular}{l} 
Denotes the \\
activity state of the \\
processor module
\end{tabular} & Green & \begin{tabular}{l} 
Processor module \\
is in RUN mode
\end{tabular} & \begin{tabular}{l} 
Processor \\
module is in \\
STOP mode
\end{tabular} & \begin{tabular}{l} 
If the LED flashes fast (4 \\
Hz) a firmware update is \\
finished with no errors. \\
If the Run LED flashes fast
\end{tabular} \\
\(\left(\begin{array}{lll}4 \text { Hz), alternating with a } \\
\text { flashing Run LED the firm- } \\
\text { ware is updated. }\end{array}\right.\) \\
\hline \begin{tabular}{l} 
To enforce boot mode 1, \\
keep the RUN function \\
key pressed during the \\
boot procedure. In this \\
case, the Run LED flashes \\
slowly (1 Hz). A subsequent
\end{tabular} \\
project download (from \\
within Automation Builder) \\
cancels the blinking.
\end{tabular}\(\left|-\begin{array}{l}\text { If the Error LED flashes } \\
\text { slowly (1 Hz) a firmware } \\
\text { update from the memory } \\
\text { card is finished with errors. }\end{array}\right|\)

A running processor module is indicated with the state RUN on the display, a deactivated processor module is indicated with the state STOP. In both cases the display's backlight is off.

\subsection*{6.8.2.5.5 Description of the function keys}

\section*{Overview}

The processor module can be operated manually using the function keys on the front panel:
\begin{tabular}{|l|l|l|}
\hline Function key & Description & Description \\
\hline RUN & Run & \begin{tabular}{l} 
Toggles between RUN and STOP mode. Switching \\
into RUN mode is only possible if an error free \\
project has been created and downloaded with \\
Automation Builder.
\end{tabular} \\
\hline VAL & Value & \begin{tabular}{l} 
Shows different state values of the processor \\
module.
\end{tabular} \\
\hline ESC & Escape & \begin{tabular}{l} 
Quits the current menu, submenu or function without \\
saving.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|l|}
\hline Function key & Description & Description \\
\hline OK & OK/Acknowledgement & \begin{tabular}{l} 
Acknowledges the current value or selects a menu/ \\
submenu. Changes that have been sent to the pro- \\
cessor module successfully are confirmed with donE \\
on the display.
\end{tabular} \\
\hline DIAG & Diagnostic & Allows evaluation of error messages in detail. \\
\hline CFG & Configuration & \begin{tabular}{l} 
Show/set IP configuration, PLC startup mode and \\
Ethernet address. \\
Enters submenus.
\end{tabular} \\
\hline Sount up/navigate in & \begin{tabular}{l} 
Press the function key repeatedly in order to \\
increase the value each time by 1, or navigate in \\
submenu to previous entry. \\
submenu
\end{tabular} \\
Keep the function key pressed in order to count up \\
fast.
\end{tabular}

Backlight is switched on for about 20 seconds by pressing any function key.

\section*{RUN - start and stop PLC}

\section*{Function key} RUN
\begin{tabular}{|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{State} & \multirow[t]{2}{*}{Description menu level 0} & \multicolumn{3}{|l|}{Result on pressing one of the function keys} \\
\hline & & RUN & \[
\mathrm{OK}
\] & ESC \\
\hline 0 & , 5Lロア & \begin{tabular}{l}
Press short: State 1 is displayed. \\
Press long (>5 sec): State 2 is displayed.
\end{tabular} & \multirow[t]{2}{*}{No action} & \multirow[t]{2}{*}{No action} \\
\hline 1 & PLC only in state RUN if a correct project is in RAM of PLC. & \begin{tabular}{l}
State 0 is displayed. \\
STOP - \\
same as \\
Online stop \\
in \\
Automation \\
Builder \\
(Stop, no init \\
of variables)
\end{tabular} & & \\
\hline 2 & rESEL & \[
\begin{aligned}
& \text { RUN } \\
& \text { LED=ON }
\end{aligned}
\] & \begin{tabular}{l}
Perform RESET same as Online reset in Automation Builder (stop and init variables) \\
State 0 is displayed.
\end{tabular} & \begin{tabular}{l}
No RESET \\
State 0 is displayed.
\end{tabular} \\
\hline
\end{tabular}

\section*{VAL - reading out state values}

Values with SystemFW 3.1.x and DisplayFW 3.0
The following settings of the processor module can be read out by pressing the function key VAL repeatedly:
1. Displays time of the processor module (hh.mm.ss).
2. Displays date of the processor module (yy.mm.dd).
3. Displays state of battery (ub \(100=100 \%\), ub \(020=20 \%\) or ub \(000=\) empty).
4. Displays version of display firmware (e.g. d 3.0 r (= display version 3.0 release)).
5. Displays version of CPU firmware (e.g. C 3.1.0r (= CPU version 3.1.0 release)).
6. Displays CPU type.
7. Displays default text (RUN/STOP).

Values with SystemFW >=V3.2.0 and DisplayFW >=V4.1
Function key By pressing function key VAL state 1 is displayed.
VAL
\begin{tabular}{|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Stat
|e} & \multirow[t]{2}{*}{Description menu VAL} & \multicolumn{4}{|l|}{Result on pressing one of the function keys} \\
\hline & & CFG &  &  & (ESC OK \\
\hline 1 & \begin{tabular}{l}
160001 \\
Time of the processor module (hh.mm.ss).
\end{tabular} & \multirow[t]{5}{*}{No action} & State 2 is displayed. & State 6 is displayed. & \multirow[t]{5}{*}{Go back to main menue RUN/STOP.} \\
\hline 2 & \begin{tabular}{l}
170625 \\
Date of the processor module (yy.mm.dd).
\end{tabular} & & State 3 is displayed. & State 1 is displayed. & \\
\hline 3 & State of battery (ub \(100=100 \%\), ub 020 = 20\% or ub \(000=\) empty). & & State 4 is displayed. & State 2 is displayed. & \\
\hline 4 & Version of display firmware (e.g. d 4.1 r (= display version 4.1 release). & & State 5 is displayed. & State 3 is displayed. & \\
\hline 5 & Version of CPU firmware (e.g. C 3.2.0r (= CPU version 3.2.0 release). & & State 6 is displayed. & State 4 is displayed. & \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{\begin{tabular}{l}
Stat \\
e
\end{tabular}} & \multirow[t]{2}{*}{Description menu VAL} & \multicolumn{5}{|l|}{Result on pressing one of the function keys} \\
\hline & & CFG & \(\square\) &  & ESC & OK \\
\hline 6 & \(\square\) & & State 1 is displayed. & State 5 is displayed. & & \\
\hline
\end{tabular}

\section*{DIAG - reading out diagnosis messages}

Table 811: Example - no diagnosis message in state list
\begin{tabular}{|l|l|l|l|l|l|l|l|}
\hline State & Display & \multicolumn{4}{|l|}{ Result on pressing one of the function keys } \\
\cline { 3 - 6 } & & DIAG & & - & - & \\
\hline 0 & \begin{tabular}{l} 
The processor \\
module is in \\
RUN/STOP \\
mode.
\end{tabular} & \begin{tabular}{l} 
State 1 is \\
displayed.
\end{tabular} & - & - & \\
\hline 1 & & No action & No action & \begin{tabular}{l} 
Return into RUN/STOP \\
mode.
\end{tabular} \\
\hline
\end{tabular}

Table 812: Example - diagnosis messages in state list
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{State} & \multirow[t]{2}{*}{Display} & \multicolumn{5}{|l|}{Result on pressing one of the function keys} \\
\hline & & DIAG &  &  & ESC &  \\
\hline 0 & The processor module is in RUN/STOP mode. & State 1 is displayed. & - & - & - & - \\
\hline 1 & Number of diagnosis messages; here 4 & & Go to first/ next diagnosis message in state list (e.g., state 2). & Go to last/ previous diagnosis message in state list. & Return into RUN/STOP mode. & Return into RUN/STOP mode. \\
\hline 2 & F.E4 bRt
Diagnosis mes-
sage example:
Error battery
empty or
missing.
Toggling
between state 2
and 3. & Selects displayed diagnosis message and shows details. (4) Table 8 13 "Exam ple - error battery empty or missing" on page 4 & Go to first/ next diagnosis message in state list. & Go to last/ previous diagnosis message in state list. & Return into RUN/STOP mode. & Acknowledge and return into RUN/STOP mode. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|l|l|l|l|l|}
\hline State & Display & \multicolumn{4}{l|}{ Result on pressing one of the function keys } \\
\cline { 3 - 6 } & & DIAG & & & & \\
\hline 3 & \begin{tabular}{l} 
ESC \\
\hline
\end{tabular} & \begin{tabular}{l} 
Error ID \\
example \\
Toggling \\
between state 2 \\
and 3
\end{tabular} & & & & \\
\hline
\end{tabular}

Table 813: Example - error battery empty or missing
\begin{tabular}{|l|l|l|l|l|l|l|l|}
\hline State & Display & Result on pressing one of the function keys \\
\hline 0 & & & & & & \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{State} & \multirow[t]{2}{*}{Display} & \multicolumn{5}{|l|}{Result on pressing one of the function keys} \\
\hline & & DIAG &  &  & ESC & \[
\mathrm{OK}
\] \\
\hline 5 &  & & State 6 is displayed. & State 4 is displayed. & \begin{tabular}{l}
State 0 is displayed. \\
Return to diagnosis state list.
\end{tabular} & State 0 is displayed Return to diagnosis state list. \\
\hline 6 & ain : \(14 \quad 0 \mathrm{E}\)
Detail 4
Additional infor-
mation 0: none & & State 1 is displayed. & State 5 is displayed. & \begin{tabular}{l}
State 0 is displayed. \\
Return to diagnosis state list.
\end{tabular} & \begin{tabular}{l}
State 0 is displayed. \\
Return to diagnosis state list.
\end{tabular} \\
\hline
\end{tabular}

\section*{CFG - configuration}

\section*{Configuration with SystemFW V3.1.x and DisplayFW V3.0}

Function key CFG main menu with ETH1/ETH2 mode:"Two separate interfaces"

๕ Chapter 6.3.2.9.3 "Switch functionality of Ethernet interfaces ETH1/ETH2" on page 1520
Navigation starts with the processor module being in RUN/STOP mode (State 0). By pressing one of the 3 function keys a certain action is triggered. The result of this action is described in the result columns of the tables.
\begin{tabular}{|c|c|c|c|c|}
\hline State & Description - main menu 1 & \multicolumn{3}{|l|}{Result on pressing one of the function keys} \\
\hline & & CFG & ESC & \[
\mathrm{OK}
\] \\
\hline 0 & The processor module is in RUN/STOP mode. & State 1 is displayed. & Remains in RUN/ STOP mode. & Remains in RUN/ STOP mode. \\
\hline 1 & IPELH & State 2 is displayed. & Return into RUN/ STOP mode. & Refers to submenu 1. \\
\hline 2 & \begin{tabular}{l}
… \(\mathrm{AdrO00}{ }^{\circ}\) \\
Change the values with the Count up/Count down function keys.
\end{tabular} & State 3 is displayed. & Return into RUN/ STOP mode. & Shows DONE, your settings are saved. Return into RUN/STOP mode. \\
\hline 3 & : PELHZ & State 4 is displayed. & Return into RUN/ STOP mode. & Refers to submenu 1. \\
\hline 4 & \begin{tabular}{l}
쁠.Rdr000 \\
Change the values with the Count up/Count down function keys.
\end{tabular} & State 5 is displayed. & Return into RUN/ STOP mode. & Shows DONE, your settings are saved. Return into RUN/STOP mode. \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|}
\hline State & Description - main menu 1 & \multicolumn{3}{|l|}{Result on pressing one of the function keys} \\
\hline 5 & \begin{tabular}{l}
Change the values with the Count up/Count down function keys. \\
をy) Further information on page 4455
\end{tabular} & State 1 is displayed. & Return into RUN/ STOP mode. & Shows DONE, your settings are saved. Return into RUN/STOP mode. \\
\hline
\end{tabular}

Function key CFG main menu with ETH1/ETH2 mode: "Switch functionality ETH1-ETH2"
« Chapter 6.3.2.9.3 "Switch functionality of Ethernet interfaces ETH1/ETH2" on page 1520 Navigation starts with the processor module being in RUN/STOP mode (State 0). By pressing one of the 3 function keys a certain action is triggered. The result of this action is described in the result columns of the tables.
\begin{tabular}{|c|c|c|c|c|}
\hline State & Description - main menu 2 & \multicolumn{3}{|l|}{Result on pressing one of the function keys} \\
\hline & & CFG & ESC &  \\
\hline 0 & The processor module is in RUN/STOP mode. & State 1 is displayed. & Remains in RUN/ STOP mode. & Remains in RUN/ STOP mode. \\
\hline 1 &  & State 2 is displayed. & Return into RUN/ STOP mode. & Refers to submenu 1. \\
\hline 2 & Change the values with the Count up/Count down function keys. & State 3 is displayed. & Return into RUN/ STOP mode. & Your settings are saved. State 2 is displayed. \\
\hline 3 & \begin{tabular}{l}
Change the values with the Count up/Count down function keys. \\
を) Further information on page 4455
\end{tabular} & State 1 is displayed. & Return into RUN/ STOP mode. & Shows DONE, your settings are saved. Return into RUN/STOP mode. \\
\hline
\end{tabular}

Function key CFG submenu IPETH1 or IPETH2; DHCP not active
\begin{tabular}{|l|l|l|l|l|}
\hline \begin{tabular}{l} 
Stat \\
e
\end{tabular} & \multicolumn{2}{|l|}{ Description - submenu 1 } & \multicolumn{2}{l|}{ Result on pressing one of the function keys } \\
\hline 1.1 & & CFG & ESC & OK \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|}
\hline Stat & Description - submenu 1 & \multicolumn{3}{|l|}{Result on pressing one of the function keys} \\
\hline 1.2 & IP Configuration (address, subnet mask, gateway) & State 1.3 is displayed. & Aborts the menu unchanged. Return to state 1.1. & State 3.2 is displayed. \\
\hline 1.3 & Reset to production data (default settings). & State 1.4 is displayed. & Aborts the menu unchanged. Return to state 1.1 & \begin{tabular}{l}
Activate RESET to default by pressing OK twice. \\
Shows DONE, your settings are saved. Return into RUN/STOP mode.
\end{tabular} \\
\hline 1.4 & \begin{tabular}{l}
Activate DHCP \\
Sets a DHCP address.
\end{tabular} & State 1.2 is displayed. & Aborts the menu unchanged. Return to state 1.1. & \begin{tabular}{l}
Activate DHCP to default by pressing OK twice. \\
Shows DONE, your settings are saved. Return into RUN/STOP mode.
\end{tabular} \\
\hline
\end{tabular}

Function key CFG submenu IPETH1 or IPETH2; DHCP active
\begin{tabular}{|c|c|c|c|c|}
\hline Stat & Description - submenu 2 & \multicolumn{3}{|l|}{Result on pressing one of the function keys} \\
\hline & & CFG & ESC & \[
\mathrm{OK}
\] \\
\hline 2.1 & OELH
IPETH1 or IPETH2 & State 2 is displayed. & Aborts the menu unchanged. Return to state 0. & State 2.2 is displayed. \\
\hline 2.2 & DHCP active & State 2.3 is displayed. & Aborts the menu unchanged. Return to state 2.1. & -- \\
\hline 2.3 & IP Configuration (address, subnet mask, gateway) & State 2.4 is displayed. & Aborts the menu unchanged. Return to state 2.1. & State 3.2 is displayed. \\
\hline 2.4 & \begin{tabular}{l}
\(\qquad\) \\
ETH2
COM1 \\
Reset to production data (default settings)
\end{tabular} & -- & Aborts the menu unchanged. Return to state 2.1 & \begin{tabular}{l}
Activate RESET to default by pressing OK twice. \\
Shows DONE, your settings are saved. Return into RUN/STOP mode.
\end{tabular} \\
\hline
\end{tabular}

\section*{Function key CFG submenu STATIC}
\begin{tabular}{|c|c|c|c|c|}
\hline St & Description - submenu 3 & \multicolumn{3}{|l|}{Result on pressing one of the function keys} \\
\hline & & CFG & ESC & \[
\mathrm{OK}
\] \\
\hline 3.1 & IP Configuration (address, subnet mask, gateway) & State 2.4 is displayed. & \begin{tabular}{l}
Aborts the menu unchanged. \\
Return to state 1.1 (submenu IPETH1 or IPETH2)
\end{tabular} & State 3.2 is displayed. \\
\hline 3.2 & \begin{tabular}{l}
IP address A1-A4 \\
\(\star\) Number is blinking if value has changed and is not yet sent to CPU.
\end{tabular} & State 3.3 is displayed. & \begin{tabular}{l}
Aborts the menu unchanged. \\
Return to state 1.1 (submenu IPETH1 or IPETH2)
\end{tabular} & \begin{tabular}{l}
Sends changed values to CPU and go to default menue RUN/STOP. \\
Displays: \\
DONE \\
New settings stored in CPU. \\
or: \\
FAIL \\
Failed to write new settings to CPU.
\end{tabular} \\
\hline 3.3 & \begin{tabular}{l}
Subnet mask N1-N4 \\
\(\star\) Number is blinking if value has changed and is not yet sent to CPU.
\end{tabular} & State 3.4 is displayed. & \begin{tabular}{l}
Aborts the menu unchanged. \\
Return to state 1.1 (submenu IPETH1 or IPETH2)
\end{tabular} & \begin{tabular}{l}
Sends changed values to CPU and go to default menue RUN/STOP. \\
Displays: \\
DONE \\
New settings stored in CPU. \\
or: \\
FAIL \\
Failed to write new settings to CPU.
\end{tabular} \\
\hline 3.4 & \begin{tabular}{l}
 \\
Gateway G1-G4 \\
* Number is blinking if value has changed and is not yet sent to CPU.
\end{tabular} & State 3.2 is displayed again. & \begin{tabular}{l}
Aborts the menu unchanged. \\
Return to state 1.1 (submenu IPETH1 or IPETH2) \\
Aborts the menu unchanged. Return to State 1.
\end{tabular} & \begin{tabular}{l}
Sends changed values to CPU and go to default menue RUN/STOP. \\
Displays: \\
DONE \\
New settings stored in CPU. \\
or: \\
FAIL \\
Failed to write new settings to CPU.
\end{tabular} \\
\hline
\end{tabular}

\section*{Function key CFG submenu ADR}
\begin{tabular}{|c|c|c|c|c|}
\hline Stat & Description - submenu 4 & \multicolumn{3}{|l|}{Result on pressing one of the function keys} \\
\hline & & CFG & ESC & \[
\mathrm{OK}
\] \\
\hline 4.1 &  & State 4.2 is displayed. & \begin{tabular}{l}
Aborts the menu unchanged. \\
Return to state 1
\end{tabular} & \begin{tabular}{l}
DHCP not active: State 1.2 is displayed \\
DHCP active: State 2.2 is displayed
\end{tabular} \\
\hline 4.2 & \begin{tabular}{l}
. Adran \(^{*} 000^{\circ}=\) \\
Change the values with the Count up/Count down function keys starting with current value. \\
\(\star\) Number is blinking if value has changed and is not yet sent to CPU.
\end{tabular} & State 2.3 is displayed. & \begin{tabular}{l}
Aborts the menu unchanged. \\
Return to state 4.1.
\end{tabular} & \begin{tabular}{l}
Sends changed values to CPU and go to default menue RUN/STOP. \\
Displays: \\
DONE \\
New settings stored in CPU. or: \\
FAIL \\
Failed to write new settings to CPU.
\end{tabular} \\
\hline 4.3 & \begin{tabular}{l}
. Rdr**55 \(\qquad\) \\
Subnet mask N1-N4 \\
\(\star\) Number is blinking if value has changed and is not yet sent to CPU.
\end{tabular} & State 4.1 is displayed. & \begin{tabular}{l}
Aborts the menu unchanged. \\
Return to state 4.1
\end{tabular} & \begin{tabular}{l}
Sends changed values to CPU and go to default menue RUN/STOP. \\
Displays: \\
DONE \\
New settings stored in CPU. \\
or: \\
FAIL \\
Failed to write new settings to CPU.
\end{tabular} \\
\hline
\end{tabular}

\section*{Configuration with SystemFW >=V3.5.0 and DisplayFW >=V4.1} CFG menu level 1, with ETH1/ ETH2 mode:
"Two separate interfaces"

Function key \(\Leftrightarrow\) Chapter 6.3.2.9.3 "Switch functionality of Ethernet interfaces ETH1/ETH2" on page 1520
Navigation starts with the processor module being in RUN/STOP mode (State 0). By pressing one of the 5 function keys a certain action is triggered. The result of this action is described in the result columns of the tables.
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{State} & \multirow[t]{2}{*}{Description - CFG menu level 1} & \multicolumn{5}{|l|}{Result on pressing one of the function keys} \\
\hline & & CFG &  &  & ESC & \[
\mathrm{OK}
\] \\
\hline 0 & The processor module is in RUN/ STOP mode. & State 1 is displayed. & & & Remains in RUN/STOP mode. & Remains in RUN/STOP mode. \\
\hline 1 & \begin{tabular}{l}
\(\qquad\) \\
Switch is OFF
\end{tabular} & Refers to submenu level 2. © "Function key CFG submenu show/set PLC ID" on page 447 & \begin{tabular}{l}
State 2 \\
is dis- \\
played if \\
KNX \\
func- \\
tionality \\
active. \\
State 3 \\
is dis- \\
played if \\
KNX \\
func- \\
tionality \\
is inac- \\
tive.
\end{tabular} & State 6 is displayed. & Return into RUN/STOP mode. & \\
\hline 2 & \begin{tabular}{l}
 \\
KNX program button (appears if functionality is active).
\end{tabular} & & State 3 is displayed. & State 1 is displayed. & Return into RUN/STOP mode. & \\
\hline 3 & … PELH & \begin{tabular}{l}
Refers to submenu level 2 (3) "Function key CFG menu level 2 \\
(IPETH1 or IPETH2); " on page 447
\end{tabular} & State 4 is displayed. & State 2 is displayed. & Return into RUN/STOP mode. & Only active if no changes in CFG menu. Return into RUN/STOP mode. \\
\hline 4 & \[
\text { : } 1 \text { PELHZ }
\] & Refers to submenu level 2 (3) "Function key CFG menu level 2 (IPETH1 or IPETH2); " on page 4470 & State 5 is displayed. & State 3 is displayed. & Return into RUN/STOP mode. & \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{State} & \multirow[t]{2}{*}{Description - CFG menu level 1} & \multicolumn{5}{|l|}{Result on pressing one of the function keys} \\
\hline & & CFG &  &  & ESC & \[
\mathrm{OK}
\] \\
\hline 5 & \begin{tabular}{l}
 \\
Note: COM1 mode RS-232 (default) or RS-485 can only be shown but not changed. This is a PLC boot parameter \({ }^{\circ}\) Chapter 6.3.2.14.1 "Setting up a serial interface" on page 1587 and must be set in AB © Chapter 6.3.2.14.1.3 "Configuration" on page 1587. Mode is activated in PLC boot process.
\end{tabular} & & State 6 is displayed. & State 4 is displayed. & Return into RUN/STOP mode. & Return into RUN/STOP mode. \\
\hline 6 & \% Mad 70 & Refers to submenu set startup mode of PLC. * "Function key CFG submenu show/set startup mode of PLC" on page 4475 & State 1 is displayed. & State 5 is displayed. & Return into RUN/STOP mode. & \\
\hline
\end{tabular}

Function key
CFG menu level
1, with ETH1/
ETH2 mode:
"Switch
functionality,
ETH1-ETH2"

② Chapter 6.3.2.9.3 "Switch functionality of Ethernet interfaces ETH1/ETH2" on page 1520
Navigation starts with the processor module being in RUN/STOP mode (State 0). By pressing one of the 5 function keys a certain action is triggered. The result of this action is described in the result columns of the tables.
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{State} & \multirow[t]{2}{*}{Description - CFG menu level 1} & \multicolumn{5}{|l|}{Result on pressing one of the function keys} \\
\hline & & CFG &  &  & ESC & OK \\
\hline 0 & The processor module is in RUN/ STOP mode. & State 1 is displayed. & & & Remains in RUN/STOP mode. & Remains in RUN/STOP mode. \\
\hline 1 &  & & \begin{tabular}{l}
State 2 is \\
displayed \\
if KNX \\
function- \\
ality is \\
active. \\
State 3 is \\
displayed \\
if KNX \\
function- \\
ality is \\
inactive.
\end{tabular} & State 5 is displayed. & Return into RUN/STOP mode. & Return into RUN/STOP mode. \\
\hline 2 & \begin{tabular}{l}
P. Pbut 0 \\
KNX program button (appears if functionality is active).
\end{tabular} & & State 3 is displayed. & State 1 is displayed. & Return into RUN/STOP mode. & Return into RUN/STOP mode. \\
\hline 3 & : PELH I & Refers to submenu level 2. (y) "Function key CFG menu level 2 (IPETH1 or IPETH2); " on page 44 & State 4 is displayed. & State 2 is displayed. & Return into RUN/STOP mode. & \begin{tabular}{l}
Only active if no changes in CFG menu. \\
Return into RUN/STOP mode.
\end{tabular} \\
\hline 4 & \begin{tabular}{l} 
\%. 5 2J己. \\
\hline Note: COM1 mode \\
RS-232 (default) or \\
RS-485 can only \\
be shown but not \\
changed. This is \\
a PLC boot param- \\
eter \& Chapter \\
6.3.2.14.1 "Setting \\
up a serial inter- \\
face" on page 1587 \\
and must be set \\
in AB \& Chapter \\
6.3.2.14.1.3 \\
"Configuration" \\
on page 1587. Mode \\
is activated in PLC \\
boot process.
\end{tabular} & & State 5 is displayed. & State 3 is displayed. & Return into RUN/STOP mode. & Return into RUN/STOP mode. \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{State} & \multirow[t]{2}{*}{Description - CFG menu level 1} & \multicolumn{4}{|l|}{Result on pressing one of the function keys} \\
\hline & & CFG &  & ESC & OK \\
\hline 5 & …7ad 70 & \begin{tabular}{l|l|}
\hline Refers to & State 1 is \\
submenu & dis- \\
set startup & played. \\
mode of & \\
PLC. & \\
¿ "Func- & \\
tion key & \\
CFG sub- & \\
menu & \\
show/set & \\
startup & \\
mode of & \\
PLC" & \\
on page 4475 \\
\hline
\end{tabular} & State 4 is displayed. & Return into RUN/STOP mode. & Return into RUN/STOP mode. \\
\hline
\end{tabular}

Function key CFG menu level 2 (IPETH1 or IPETH2);
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{State} & \multirow[t]{2}{*}{Description - CFG menu level 2} & \multicolumn{5}{|l|}{Result on pressing one of the function keys} \\
\hline & & CFG &  &  & ESC & OK \\
\hline 0 & \begin{tabular}{l}
:1 PELH \\
IPETH1 or IPETH2
\end{tabular} & State 1 is displayed. & & & Return into RUN/STOP mode. & Return into RUN/ STOP mode. \\
\hline 1 & IP Configuration
(address, ubnet
mask, gateway). & Refers to submenu StAtIC. ③ "Function key CFG menu level 3, show/set STATIC" on page 447 & \begin{tabular}{l}
State 2 is displayed. \\
1
\end{tabular} & State 4 is displayed. & \begin{tabular}{l}
Case 1, no submenu is entered: Return into RUN/STOP mode. \\
Case 2, no changes: State 0 is displayed.
\end{tabular} & \begin{tabular}{l}
Send all changes to CPU. \\
State 5 is displayed.
\end{tabular} \\
\hline 2 & \({ }_{*}^{*}\). \(\mathrm{nodHLP}{ }^{\text {a }}\) & Refers to submenu set dHCP "Function key CFG menu level 2 Show/set DHCP" on page 447 & State 3 is displayed. & State 1 is displayed. & & \\
\hline 3 &  & Refers to submenu set ld. ③ "Function key CFG menu level 2 Show/set Id" on page 447 & \begin{tabular}{l}
State 4 is displayed. \\
3
\end{tabular} & State 2 is displayed. & State 1 is displayed. & \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{State} & \multirow[t]{2}{*}{Description - CFG menu level 2} & \multicolumn{5}{|l|}{Result on pressing one of the function keys} \\
\hline & & CFG & \(\square\) & ( & ESC & OK \\
\hline 4 & \begin{tabular}{l}
:rE5Et \\
Reset to production data (default settings).
\end{tabular} & Refers to submenu set \(r E S E t\). (4) "Function key CFG submenu show/set RESET" on page \(44 \backslash\) & \begin{tabular}{l}
State 1 is displayed. \\
4
\end{tabular} & State 3 is displayed. & & \begin{tabular}{l}
Send all changes to CPU. \\
State 5 is displayed.
\end{tabular} \\
\hline 5 & \({ }^{\text {a }}\) : APPL ( \({ }^{\text {a }}\) & & & & \begin{tabular}{l}
Remain all changes. \\
State 1 is displayed.
\end{tabular} & State 6 is displayed. \\
\hline \multirow[t]{2}{*}{6} & : donE & \multicolumn{5}{|l|}{\begin{tabular}{l}
Changes applied \\
donE is displayed for 2 sec. then return into RUN/STOP mode.
\end{tabular}} \\
\hline & ․FRILEd & \multicolumn{5}{|l|}{\begin{tabular}{l}
Changes failed \\
FAILEd is displayed for 2 sec. then return into RUN/STOP mode.
\end{tabular}} \\
\hline
\end{tabular}

Function key CFG menu level 3, show/set STATIC
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{State} & \multirow[t]{2}{*}{Description - submenu STATIC} & \multicolumn{5}{|l|}{Result on pressing one of the function keys} \\
\hline & & CFG &  &  & ESC &  \\
\hline 0 & \begin{tabular}{l}
\[
\text { . } 5 \text { SALI C }
\] \\
IP Configuration (address, subnet mask, gateway).
\end{tabular} & State 1 is displayed. & & & Aborts the menu unchanged. Return to IPETH1 or IPETH2 & \begin{tabular}{l}
No changes: return to IPETH1 or IPETH2 \\
Changes: (StAtIC blinks) state 5 of previous table is displayed.
\end{tabular} \\
\hline 1 & \begin{tabular}{l}
IP address A1-A4 \\
If submenu is entered: Number is blinking if value has changed and is not yet sent to CPU.
\end{tabular} & Refers to submenu of A1-A4 & Count down A1A4 & Count up A4-A1 & Aborts the menu unchanged. Return to CFG menu level 2 StAtIC. & \begin{tabular}{l}
Take over new values, but don't send to CPU. \\
Return to CFG menu level 2 StAtIC.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{State} & \multirow[t]{2}{*}{Description - submenu STATIC} & \multicolumn{5}{|l|}{Result on pressing one of the function keys} \\
\hline & & CFG &  &  & ESC & \[
\mathrm{OK}
\] \\
\hline 2 & \begin{tabular}{l}
 \\
Subnet mask N1-N4 \\
If submenu is entered: Number is blinking if value has changed and is not yet sent to CPU.
\end{tabular} & Refers to submenu of N1-N4 & Count down N1N4 & \[
\begin{array}{|l|}
\hline \text { Count up } \\
\text { N4-N1 }
\end{array}
\] & \begin{tabular}{l}
Aborts the menu unchanged. \\
Return to CFG menu level 2 StAtIC.
\end{tabular} & \begin{tabular}{l}
Take over new values, but don't send to CPU. \\
Return to CFG menu level 2 StAtIC
\end{tabular} \\
\hline 3 & \begin{tabular}{l}
 \\
Gateway G1-G4 \\
If submenu is entered: Number is blinking if value has changed and is not yet sent to CPU
\end{tabular} & Refers to submenu G1-G4 & Count down G1G4 & \[
\begin{aligned}
& \text { Count up } \\
& \text { G4-G1 }
\end{aligned}
\] & \begin{tabular}{l}
Aborts the menu unchanged. \\
Return to CFG menu level 2 StAtIC.
\end{tabular} & \begin{tabular}{l}
Take over new values, but don't send to CPU. \\
Return to CFG menu level 2 StAtIC.
\end{tabular} \\
\hline
\end{tabular}

Function key CFG menu level 4, Example: I/P address A4
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{State} & \multirow[t]{2}{*}{\[
\begin{aligned}
& \text { Description - CFG } \\
& \text { menu level } 4
\end{aligned}
\]} & \multicolumn{5}{|l|}{Result on pressing one of the function keys} \\
\hline & & CFG &  &  & ESC & OK \\
\hline 0 & \begin{tabular}{l}
\[
\text { . }{ }^{*} .84010
\] \\
A4 is blinking if submenu is entered.
\end{tabular} & No action & \begin{tabular}{l}
Count down value. \\
Value is blinking if changed.
\end{tabular} & \begin{tabular}{l}
Count up value. \\
Value is blinking if changed.
\end{tabular} & \begin{tabular}{l}
Aborts the menu unchanged. \\
Shows unchanged value.
\end{tabular} & \begin{tabular}{l}
Take over new value, but don't send to CPU. \\
Go back to menu level 3 (here subnet mask N1).
\end{tabular} \\
\hline
\end{tabular}

Function key CFG menu level 2 Show/set DHCP


Function key CFG menu level 2 Show/set Id
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{\begin{tabular}{l}
Stat \\
e
\end{tabular}} & \multirow[t]{2}{*}{Description show/set Id} & \multicolumn{5}{|l|}{Result on pressing one of the function keys} \\
\hline & & CFG &  &  &  &  \\
\hline 0 &  & & Count down value: 255 ... 000, starting with current value. & Count up value: 000 ... 255, starting with current value. & Case 2, no changes: stop blinking, return to menu. 4y "Function key CFG menu level 2 (IPETH1 or IPETH2); " on page 447 & Go toCFG menu level 2 (IPETH1 or IPETH2) state 5-6. \\
\hline 1 & \begin{tabular}{l}
IP address A1-A4 \\
\(\star\) Number is blinking if value has changed and is not yet sent to CPU.
\end{tabular} & & Count down value: 255 ... 000, starting with current value. & Count up value: 000 ... 255, starting with current value. & Discard the changes. Stop blinking. Show previous value. & Go toCFG menu level 2 (IPETH1 or IPETH2) state 5-6. \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Stat} & \multirow[t]{2}{*}{Description show/set Id} & \multicolumn{5}{|l|}{Result on pressing one of the function keys} \\
\hline & & CFG &  &  & ESC & OK \\
\hline 2 & \({ }^{*}{ }^{\star}\), \(1255^{\circ}\) & & \begin{tabular}{l}
Count down value: \\
255 ... 000, starting with current value.
\end{tabular} & Count up value: 000 ... 255, starting with current value. & \begin{tabular}{l}
Discard the changes. \\
Stop \\
blinking. \\
Show previous value.
\end{tabular} & Go toCFG menu level 2 (IPETH1 or IPETH2) state 5-6. \\
\hline
\end{tabular}


Function key CFG submenu show/set startup mode of PLC
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Stat
|e} & \multirow[t]{2}{*}{Description - submenu show/set startup mode of PLC} & \multicolumn{5}{|l|}{Result on pressing one of the function keys} \\
\hline & & CFG &  &  & ESC &  \\
\hline Start & Finod 00 & & Count down value: 02 ... 0 , starting with current value. & Count up value: 00 ... 02, starting with current value. & \begin{tabular}{l}
Aborts the menu. \\
Go back to main menue RUN/STOP.
\end{tabular} & \begin{tabular}{l}
Sends changed values to CPU. \\
Displays 2
\end{tabular} \\
\hline 1 & \begin{tabular}{l}
… Mod 01 \\
Text is blinking if value has changed and is not yet sent to CPU.
\end{tabular} & & Count down value: 02 ... 00, starting with current value. & Count up value: 00 ... 02, starting with current value. & \begin{tabular}{l}
Aborts the menu. \\
Go back to main menue RUN/STOP.
\end{tabular} & \begin{tabular}{l}
sec. donEor FAILEd. \\
Go back to main menue RUN/STOP.
\end{tabular} \\
\hline 2 & Text is blinking if value has changed and is not yet sent to CPU. & & Count down value: 02 ... 00, starting with current value. & Count up value: 00 ... 02, starting with current value. & \begin{tabular}{l}
Aborts the menu. \\
Go back to main menue RUN/STOP.
\end{tabular} & \\
\hline
\end{tabular}

Function key CFG submenu show/set PLC ID
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Stat e} & \multirow[t]{2}{*}{Description - submenu show/set PLC ID} & \multicolumn{5}{|l|}{Result on pressing one of the function keys} \\
\hline & & CFG &  &  & (ESC) &  \\
\hline 0 & Switch is OFF & State 1 is displaye d & & & Return into RUN/STOP mode. & \\
\hline 1 &  & & State 2 is displayed & State 2 is displayed & Return into RUN/STOP mode. & \\
\hline 2 & \begin{tabular}{l}
Text PLC is blinking if edit mode with keys up/down is enabled. \\
\(\star\) Number is blinking if value has changed and is not yet sent to CPU.
\end{tabular} & & Count down value: \(255 \rightarrow 0\), starting with current value. & Count up value: \(000 \rightarrow 255\), starting with current value. & \begin{tabular}{l}
Aborts the menu. \\
Go back to main menue RUN/STOP.
\end{tabular} & Takes over new value, if changed. \\
\hline
\end{tabular}

\subsection*{6.8.2.5.6 Enable flashing of display}

Blink functionality

As of SystemFW 3.1.0 and AB 2.1.0 the blink functionality is implemented. "Blink" means activate flashing of backlight of AC500 display in Automation Builder via "IP-Configuration" tool.

Wink function- As of SystemFW 3.1.0 and AB 2.0.0 the wink functionality is implemented. "Wink" means ality activate flashing of backlight of AC500 display in Automation Builder via communication settings wink functionality.

\subsection*{6.8.2.5.7 Function blocks}

PmErrLedSet This function block switches the ERR-LED ON and OFF.

PmDispSetText With this function block a text can be displayed on the CPU.
«y Chapter 6.5.14 "Reference, function blocks" on page 4086

\subsection*{6.8.2.6 Onboard technologies}

\subsection*{6.8.2.6.1 Ethernet}

Ethernet protocols and ports for AC500-eCo V3 processor modules
General
Supported as of Automation Builder V 2.4.1
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline Description & \begin{tabular}{l}
\(\geq\) CPU \\
firmware
\end{tabular} & \[
\begin{aligned}
& \text { PM5012- } \\
& \text { x-ETH }
\end{aligned}
\] & \[
\begin{aligned}
& \text { PM5032- } \\
& \text { x-ETH }
\end{aligned}
\] & \[
\begin{aligned}
& \text { PM5052- } \\
& \text { x-ETH }
\end{aligned}
\] & \[
\begin{gathered}
\text { PM5072- } \\
\text { T-2ETH(W } \\
\text { ) }
\end{gathered}
\] & \begin{tabular}{l}
\(\geq\) CPU \\
firmware
\end{tabular} & \[
\begin{gathered}
\text { PM5082- } \\
\text { T-2ETH } \\
\left.{ }^{3}\right)
\end{gathered}
\] \\
\hline ABB netConfig & V3.4.1 & x & x & x & X & V3.6.2 & x \\
\hline Online access with driver 3S TCP/IP BIkDrvTcp & V3.4.1 & X & X & X & X & V3.6.2 & X \\
\hline Modbus TCP server & V3.4.1 & X & X & x & X & V3.6.2 & x \\
\hline Modbus TCP client with POU ModTcpMast & V3.4.1 & X & X & X & X & V3.6.2 & X \\
\hline UDP out of user program with library netBaseService.lib & V3.4.1 & X & X & X & X & V3.6.2 & x \\
\hline UDP data exchange, Network variables & V3.4.1 & x & x & x & x & V3.6.2 & x \\
\hline TCP/IP out of user program with library netBaseService.lib & V3.4.1 & X & X & X & X & V3.6.2 & X \\
\hline Web server on PLC with web visualization & V3.4.1 & & X & X & X & V3.6.2 & X \\
\hline IEC60870-5-104 control station incl. \(2^{\text {nd }}\) connection and \(2^{\text {nd }}\) port & - & & & & & & \\
\hline IEC60870-5-104 substation incl. \(2^{\text {nd }}\) port & V3.4.1 & & & & X & V3.6.2 & X \\
\hline FTP server & V3.4.1 & & X & X & X & V3.6.2 & x \\
\hline CODESYS network variables & V3.4.1 & X & x & x & X & V3.6.2 & x \\
\hline OPC DA server & V3.4.1 & X & X & X & X & V3.6.2 & X \\
\hline OPC UA server & V3.4.1 & & X & X & X & V3.6.2 & X \\
\hline ICMP - ping out of user project with POU ETHx_ICMP_PING & V3.4.1 & X & X & X & X & V3.6.2 & X \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline Description & \begin{tabular}{l}
\(\geq\) CPU \\
firmware
\end{tabular} & \[
\begin{gathered}
\text { PM5012- } \\
\text { X-ETH }
\end{gathered}
\] & \[
\begin{gathered}
\text { PM5032- } \\
\text { X-ETH }
\end{gathered}
\] & \[
\begin{gathered}
\text { PM5052- } \\
\text { x-ETH }
\end{gathered}
\] & \[
\begin{gathered}
\text { PM5072- } \\
\text { T-2ETH(W } \\
\text { ) }
\end{gathered}
\] & \begin{tabular}{l}
\(\geq\) CPU \\
firmware
\end{tabular} & \[
\begin{gathered}
\text { PM5082- } \\
\text { T-2ETH } \\
3)
\end{gathered}
\] \\
\hline DHCP client & V3.4.1 & X & X & X & X & V3.6.2 & X \\
\hline SNTP (Simple Network Time Protocol) client system solution \# Chapter 6.3.4.4.2 "(S)NTP client configuration" on page 1718 & V3.4.1 & X & X & X & X & V3.6.2 & X \\
\hline SNTP (Simple Network Time Protocol) server system solution (4) Chapter 6.3.4.4.3 "(S)NTP server configuration" on page 1720 & V3.4.1 & X & X & X & X & V3.6.2 & X \\
\hline Maximum number of Input/ output allowed variable on Ethernet for the protocol & V3.4.1 & & \[
\begin{gathered}
1 \mathrm{kB} / 1 \\
\mathrm{kB}
\end{gathered}
\] & \[
\begin{gathered}
1 \mathrm{kB} / 1 \\
\mathrm{kB}
\end{gathered}
\] & \(2 \mathrm{kB} / 2 \mathrm{kB}\) & V3.6.2 & \[
\begin{gathered}
2 \mathrm{kB} / 2 \\
\mathrm{kB}
\end{gathered}
\] \\
\hline IEC 61850 (MMS server, GOOSE) \({ }^{2}\) ) & V3.4.1 & & & & X & V3.6.2 & X \\
\hline EthernetIP Scanner \({ }^{1,2}\) ) & \begin{tabular}{l}
AB 2.4.1/ \\
FW 3.4.1
\end{tabular} & & X & X & X & V3.6.2 & X \\
\hline EthernetIP Adapter \({ }^{1,2}\) ) & AB 2.4.1/ FW 3.4.1 & & X & X & X & V3.6.2 & X \\
\hline \begin{tabular}{l}
EtherCAT Master \\
*) EtherCAT configurator
\end{tabular} & V3.6.1 & & & & X & V3.6.2 & X \\
\hline KNX - Building communication \({ }^{2}\) ) & V3.4.1 & & & X & X & V3.6.2 & X \\
\hline BACnet-BC - Infrastructure communication \({ }^{2}\) ) & V3.3.1 & & & & X & V3.6.2 & X \\
\hline \begin{tabular}{l}
HTTPS - secure web server on PLC with CODESYS web visualization \\
« Chapter 6.3.4.7.3.2 "Secure web server" on page 1727
\end{tabular} & V3.4.1 & & X & X & X & V3.6.2 & X \\
\hline WebVisu for data visualisation on webserver HTML5 & V3.4.1 & & X & X & X & V3.6.2 & X \\
\hline \[
\begin{aligned}
& \hline \text { FTPS - secure FTP } \\
& \text { Chapter 6.3.4.7.3.3 "Secure } \\
& \text { FTP" on page 1728 }
\end{aligned}
\] & V3.4.1 & & X & X & X & V3.6.2 & X \\
\hline Secure online access with driver 3S TCP/IP BlkDrvTcp & V3.4.1 & X & X & X & X & V3.6.2 & X \\
\hline ICMP - ping out of user project with POU ETHx_ICMP_PING or EthlcmpPing (PLCopen style) & V3.4.1 & X & X & X & X & V3.6.2 & X \\
\hline Modbus TCP client (master) with ModTcpMast (PLCopen style) & V3.4.1 & X & X & X & X & V3.6.2 & x \\
\hline \multicolumn{8}{|l|}{\begin{tabular}{l}
Remarks: \\
\({ }^{1}\) ): in preparation \\
\({ }^{2}\) ): feature is licensed \\
\({ }^{3}\) ): from \(\geq A B 2.6 .1\) and CPU FW \(\geq 3.6 .2\)
\end{tabular}} \\
\hline
\end{tabular}

\section*{Default open Ethernet ports of PM50xx-x-xETH}

After startup without a PLC project the AC500-eCo V3 PM50xx-x-xETH contains the following Ethernet ports and sockets:
\begin{tabular}{|l|l|}
\hline Protocol & Port \\
\hline ABB NetConfig \({ }^{1}\) ) & UDP 24576 \\
\hline Online access with driver 3S Tcp/lp BIkDrvTcp (no scan) & TCP 11740 \\
\hline OPC UA server \({ }^{2}\) ) & TCP 4840 \\
\hline \begin{tabular}{l} 
Remarks: \\
1): The port 24576 for ABB NetConfig protocol can be disabled via PLC configuration by \\
deleting the protocol node from configuration tree of Ethernet interfaces ETH1 and ETH2. \\
\({ }^{2}\) ): The port 4840 for OPC UA server is closed by default as of System FW V3.1.0.
\end{tabular} \\
\hline
\end{tabular}

All other ports are closed by default.

Overview of protocols, sockets and ports
\begin{tabular}{|l|l|l|}
\hline Protocol & Port & Sockets \\
\hline ABB netConfig & 24576 & 1 permanent socket per interface \\
\hline \begin{tabular}{l} 
3S gateway client (e.g. CODESYS) to \\
gateway server
\end{tabular} & 1217 & 1 permanent socket \\
\hline \begin{tabular}{l} 
Online access with driver 3S block driver \\
TCP/IP (no scan)
\end{tabular} & 11740 & \begin{tabular}{l}
1 socket per connection + 1 \\
listen
\end{tabular} \\
\hline Modbus TCP server & \begin{tabular}{l}
502 or config- \\
urable
\end{tabular} & \begin{tabular}{l}
1 socket listen + 1 socket per \\
server connection, number of \\
server connections is configu- \\
rable in AB
\end{tabular} \\
\hline Modbus TCP client with POU ModTcpMast & Random & \begin{tabular}{l}
1 socket per connection with \\
POU ModTcpMast
\end{tabular} \\
\hline \begin{tabular}{l} 
UDP out of user program with library \\
SysLibSockets.lib
\end{tabular} & \(1 \ldots 65535\) & 1 socket per connection \\
\hline \begin{tabular}{l} 
TCP/IP out of user program with library \\
SysLibSockets.lib
\end{tabular} & \(1 \ldots 65535\) & 1 socket per connection \\
\hline Web server on PLC with web visualization & 80 & 1 listen and 1 per connection \\
\hline SNTP client & 123 & 1 permanent socket \\
\hline IEC60870-5-104 substation & 2404 & 1 per connection \\
\hline FTP server & \begin{tabular}{l} 
Command \\
port \(=21\) \\
Data active \\
mode \(=20\) \\
Data passive \\
mode \\
random
\end{tabular} & 1 per session, max. 4 allowed \\
\hline OPC DA server (default 3S block driver) & \begin{tabular}{l} 
UDP \(=1740\) \\
or \\
TCP/IP \\
\(=11740\)
\end{tabular} & 1 socket per connection \\
\hline & (UDP broadcast) \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|}
\hline Protocol & Port & Sockets \\
\hline OPC UA server & 4840 & 1 permanent socket \\
\hline ICMP - ping out of user project with POU ETHx_ICMP_PING DHCP & none & No socket \\
\hline DHCP & 67 & 1 socket during startup \\
\hline SNTP (Simple Network Time Protocol) client system solution ※ Chapter 6.3.4.4.2 "(S)NTP client configuration" on page 1718 & 123 & 1 permanent socket \\
\hline \begin{tabular}{l}
SNTP (Simple Network Time Protocol) server system solution \\
』 Chapter 6.3.4.4.3 "(S)NTP server configuration" on page 1720
\end{tabular} & 123 & 1 permanent socket \\
\hline \begin{tabular}{l}
HTTPS - secure web server on PLC with CODESYS web visualization \\
y Chapter 6.3.4.7.3.2 "Secure web server" on page 1727 \\
Not for PM5012-x-ETH!
\end{tabular} & 443 & 1 listen and 1 per connection \\
\hline \begin{tabular}{l}
FTPS - secure FTP \\
\# Chapter 6.3.4.7.3.3 "Secure FTP" on page 1728 \\
Not for PM5012-x-ETH!
\end{tabular} & \begin{tabular}{l}
Command port = 21 \\
Data active mode \(=20\) \\
Data passive mode = random
\end{tabular} & 1 per session, max. 4 allowed \\
\hline Secure online access with driver 3S TCP/IP BlkDrvTcp & 11740 & 1 socket per connection +1 listen \\
\hline ICMP - ping out of user project with POU ETHx_ICMP_PING or EthlcmpPing (PLCopen style) & None & No socket \\
\hline Modbus TCP client (master) with POU ModTcpMast (PLCopen style) & Random & 1 socket per connection with POU ModTcpMast \\
\hline
\end{tabular}

\section*{Limitation of connections per protocol}

AC500-eCo V3 processor modules
\begin{tabular}{|l|c|c|c|c|c|c|c|c|}
\hline Protocol & \begin{tabular}{c}
\(\geq\) CPU \\
firmware
\end{tabular} & \begin{tabular}{c} 
PM5012- \\
x-ETH
\end{tabular} & \begin{tabular}{c} 
PM5032- \\
x-ETH
\end{tabular} & \begin{tabular}{c} 
PM5052- \\
(-ETH
\end{tabular} & \begin{tabular}{c} 
PM5072- \\
T-2ETH(W \\
)
\end{tabular} & \begin{tabular}{c}
\(\geq\) CPU \\
firmware
\end{tabular} & \begin{tabular}{c} 
PM5082- \\
T-2ETH \\
2)
\end{tabular} \\
\hline \begin{tabular}{l} 
Modbus TCP server (e.g. for \\
SCADA access)
\end{tabular} & 3.4 .1 & 3 & 8 & 10 & 15 & 3.6 .2 & 15 \\
\hline \begin{tabular}{l} 
Modbus TCP client with POU \\
ModTcpMast (PLCopen style)
\end{tabular} & 3.4 .1 & 8 & 13 & 20 & 30 & 3.6 .2 & 30 \\
\hline \begin{tabular}{l} 
IEC60870-5-104 control station \\
incl. 2nd connection and 2nd port
\end{tabular} & & & & & & & \\
\hline \begin{tabular}{l} 
IEC60870-5-104 substation incl. \\
\(2^{\text {nd }}\) port
\end{tabular} & 3.4 .1 & & & & 5 & 3.6 .2 & 5 \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline Protocol & \[
\begin{gathered}
\geq \text { CPU } \\
\text { firmware }
\end{gathered}
\] & \[
\begin{gathered}
\text { PM5012- } \\
\text { x-ETH }
\end{gathered}
\] & \[
\begin{array}{|c}
\text { PM5032- } \\
\text { x-ETH }
\end{array}
\] & \[
\begin{gathered}
\text { PM5052- } \\
\text { x-ETH }
\end{gathered}
\] & \[
\begin{array}{|c}
\hline \text { PM5072- } \\
\mathrm{T}-2 \mathrm{ETH}(\mathrm{~W} \\
)
\end{array}
\] & \[
\begin{gathered}
\geq \text { CPU } \\
\text { firmware }
\end{gathered}
\] & \[
\begin{array}{|c}
\hline \begin{array}{c}
\text { PM5082- } \\
\text { T-2ETH } \\
\\
\text { 2 }
\end{array} \\
\hline
\end{array}
\] \\
\hline \begin{tabular}{l}
IEC60870-5-104: No. of free tags \\
+ additional license for extension \({ }^{1}\) )
\end{tabular} & 3.4.1 & & & & 1.000 & 3.6.2 & 1.000 \\
\hline FTP server & 3.4.1 & & 2 & 2 & 2 & 3.6.2 & 2 \\
\hline Online access with driver 3S TCP/IP BlkDrvTcp & 3.4.1 & 4 & 4 & 4 & 6 & 3.6.2 & 6 \\
\hline OPC DA server (number of connections) & 3.4.1 & 4 & 4 & 4 & 6 & 3.6.2 & 6 \\
\hline OPC UA server (number of connections) & 3.4.1 & & 5 & 5 & 10 & 3.6.2 & 10 \\
\hline \begin{tabular}{l}
No. of free tags \\
+ additional license for extension \({ }^{1}\) )
\end{tabular} & 3.4.1 & & 125 & 250 & 1.000 & 3.6.2 & 3.000 \\
\hline min sampling rate (limit) & 3.4.1 & & 1000 ms & 1000 ms & 500 ms & 3.6.2 & 500 ms \\
\hline Secure online access with driver 3S TCP/IP BlkDrvTcp & 3.4.1 & 4 & 4 & 4 & 6 & 3.6.2 & 6 \\
\hline FTPS - secure FTP server & 3.4.1 & & 2 & 2 & 2 & 3.6.2 & 2 \\
\hline HTTPS - Integrated webserver - number of connections & 3.4.1 & & 2 & 2 & 4 & 3.6.2 & 4 \\
\hline \begin{tabular}{l}
Ethernet/IP \\
Maximum number of Input/ Output
\end{tabular} & 3.4.1 & & \[
\begin{array}{|c|}
\hline 0.5 \\
\mathrm{kB} / 0.5 \mathrm{kB}
\end{array}
\] & \[
\begin{array}{|c|}
\hline 0.5 \\
\mathrm{kB} / 0.5 \mathrm{kB}
\end{array}
\] & \[
\begin{gathered}
0.5 \mathrm{kB} / 0.5 \\
\mathrm{kB}
\end{gathered}
\] & 3.6.2 & \[
\begin{gathered}
0.5 \\
\mathrm{kB} / 0.5 \mathrm{kB}
\end{gathered}
\] \\
\hline \multicolumn{8}{|l|}{Remarks:
1): in preparation
2): from \(\geq A B\) 2.6.1 and CPU FW \(\geq 3.6 .2\)} \\
\hline
\end{tabular}

\section*{Ethernet protocols and ports for AC500 V3 products}

\section*{General}

The communication module CM5640-2ETH acts as a port extender of the used AC500 V3 CPU.
It is not offloading the CPU for the protocols.
- the stacks are still executed in CPU.
- the performance is slightly lower or the load is slightly higher than directly from the main CPU, because each communication module and the communication module bus must also be handled.

Each CM5640-2ETH provides:
- 2 additional independent onboard Ethernet interfaces controlled by the CPU and not switched.
- all Ethernet ports, wherever located - on the communication module or on the CPU - must be in different subnets.
Addressing is not done via slot numbers but via IP addresses.
The Ethernet limitations of the CPU used together with CM5640-2ETH apply to all onboard and additional interfaces.

All interfaces of the onboard and CM5640-2ETH modules need to be in different subnets. Overlapping of subnets must be avoided. The selection of the used interface (ETH port) is done automatically according the match of the IP-address of the server and the subnet of the interfaces.
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline Description &  & \[
\begin{gathered}
\text { PM563 } \\
0-2 E T \\
H
\end{gathered}
\] & \[
\begin{gathered}
\hline \text { PM565 } \\
0-2 E T \\
H
\end{gathered}
\] & \[
\begin{gathered}
\text { PM567 } \\
0-2 \mathrm{ET} \\
\mathrm{H}
\end{gathered}
\] & \[
\begin{gathered}
\text { PM567 } \\
5-2 E T \\
H
\end{gathered}
\] & \[
\begin{gathered}
\geq \text { CPU } \\
\text { firm- } \\
\text { ware }
\end{gathered}
\] & \[
\begin{aligned}
& \text { CM564 } \\
& \text { 0-2ETH }
\end{aligned}
\] \\
\hline ABB netConfig & V3.0.0 & X & X & X & X & V3.6.0 & X \\
\hline Online access with driver 3S TCP/IP BIkDrvTcp & V3.0.0 & x & x & x & x & V3.6.0 & X \\
\hline Modbus TCP server & V3.0.3 & X & X & X & X & V3.6.0 & X \\
\hline Modbus TCP client with POU ModTcpMast & V3.0.1 & X & X & X & X & V3.6.0 & X \\
\hline UDP out of user program with library netBaseService.lib & V3.0.0 & X & X & X & X & V3.6.0 & X \\
\hline UDP data exchange, Network variables & V3.0.0 & X & X & x & x & V3.6.0 & X \\
\hline TCP/IP out of user program with library netBaseService.lib & V3.0.0 & X & X & X & X & V3.6.0 & X \\
\hline Web server on PLC with web visualization & V3.0.0 & x & x & X & x & V3.6.0 & X \\
\hline \begin{tabular}{l}
NTP/SNTP ((Simple) Network Time Protocol) client with 3S licenced store package SNTPService.package. \\
Library container: SNTPService
\end{tabular} & V3.0.0 & X & X & X & X & V3.6.0 & X \\
\hline IEC60870-5-104 control station incl. \(2^{\text {nd }}\) connection and \(2^{\text {nd }}\) port & V3.0.0 & X & X & X & x & V3.6.0 & X \\
\hline IEC60870-5-104 substation incl. \(2^{\text {nd }}\) port & V3.0.0 & X & X & X & X & V3.6.0 & X \\
\hline FTP server & V3.0.0 & X & X & X & X & V3.6.0 & X \\
\hline CODESYS network variables & V3.0.0 & X & X & x & X & V3.6.0 & X \\
\hline OPC DA server & V3.0.0 & X & X & X & X & V3.6.0 & X \\
\hline OPC UA server & V3.0.0 & X & X & X & X & V3.6.0 & X \\
\hline ICMP - ping out of user project with POU ETHx_ICMP_PING & V3.0.0 & X & X & X & X & V3.6.0 & X \\
\hline DHCP client & V3.1.0 & X & X & X & X & V3.6.0 & X \\
\hline \begin{tabular}{l}
NTP/SNTP ((Simple) Network Time Protocol) client system solution \\
を Chapter 6.3.4.4.2 "(S)NTP client configuration" on page 1718
\end{tabular} & V3.1.0 & X & X & X & X & V3.6.0 & X \\
\hline \begin{tabular}{l}
NTP/SNTP ((Simple) Network Time Protocol) server system solution \\
(4) Chapter 6.3.4.4.3 "(S)NTP server configuration" on page 1720
\end{tabular} & V3.1.0 & X & X & X & X & V3.6.0 & X \\
\hline Maximum number of Input/output allowed variable on Ethernet for the protocol & V3.4.0 & \[
\begin{gathered}
2 \mathrm{kB} / 2 \\
\mathrm{kB}
\end{gathered}
\] & \[
\begin{gathered}
4 \mathrm{kB} / 4 \\
\mathrm{kB}
\end{gathered}
\] & \[
\begin{gathered}
5 \mathrm{kB} / 5 \\
\mathrm{kB}
\end{gathered}
\] & \[
\begin{gathered}
5 \mathrm{kB} / 5 \\
\mathrm{kB}
\end{gathered}
\] & V3.6.0 & Depen ds on used CPU \\
\hline IEC 61850 (MMS server, GOOSE) \({ }^{2}\) ) & V3.1.0 & X & X & X & X & V3.6.0 & X \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline Description &  & \[
\begin{gathered}
\text { PM563 } \\
0-2 E T \\
\text { H }
\end{gathered}
\] & \[
\begin{gathered}
\hline \text { PM565 } \\
0-2 E T \\
H
\end{gathered}
\] & \[
\begin{gathered}
\text { PM567 } \\
\text { O-2ET } \\
H
\end{gathered}
\] & \[
\begin{gathered}
\text { PM567 } \\
5-2 E T \\
H
\end{gathered}
\] & \[
\begin{gathered}
\geq \text { CPU } \\
\text { firm- } \\
\text { ware }
\end{gathered}
\] & \[
\begin{aligned}
& \text { CM564 } \\
& \text { 0-2ETH }
\end{aligned}
\] \\
\hline EthernetIP Scanner \({ }^{1,2}\) ) & \begin{tabular}{l}
AB \\
2.4.1/ \\
FW \\
3.4.1
\end{tabular} & X & X & X & X & \[
\begin{gathered}
\mathrm{AB} \\
2.6 .0 / \\
\mathrm{FW} \\
3.6 .0
\end{gathered}
\] & \\
\hline EthernetIP Adapter \({ }^{1,2}\) ) & \[
\begin{array}{|l}
\text { AB } \\
2.4 .1 / \\
\text { FW } \\
3.4 .1
\end{array}
\] & x & X & x & x & \[
\begin{gathered}
\mathrm{AB} \\
2.6 .0 / \\
\mathrm{FW} \\
3.6 .0
\end{gathered}
\] & X \\
\hline KNX - Building communication \({ }^{2}\) ) & V3.2.x & X & X & X & X & V3.6.0 & X \\
\hline BACnet-BC - Infrastructure communication \({ }^{2}\) ) & V3.3.1 & X & X & X & X & V3.6.0 & X \\
\hline \begin{tabular}{l}
HTTPS - secure web server on PLC with CODESYS web visualization \\
\({ }^{4}\) ) Chapter 6.3.4.7.3.2 "Secure web server" on page 1727
\end{tabular} & V3.1.0 & X & X & X & X & V3.6.0 & X \\
\hline WebVisu for data visualisation on web server HTML5 & V3.0.0 & X & X & X & X & V3.6.0 & X \\
\hline \begin{tabular}{|lll|}
\hline FTPS - secure FTP \\
\begin{tabular}{l} 
4 Chapter \\
on page 1728
\end{tabular} & 6.3.4.7.3.3 "Secure FTP"
\end{tabular} & V3.1.0 & X & X & X & X & V3.6.0 & X \\
\hline Secure online access with driver 3S UDP BlkDrvUdp & V3.1.0 & X & X & X & X & V3.6.0 & X \\
\hline Secure online access with driver 3S TCP/IP BlkDrvTcp & V3.1.0 & X & x & X & X & V3.6.0 & X \\
\hline ICMP - ping out of user project with POU ETHx_ICMP_PING or EthlcmpPing (PLCopen style) & V3.1.0 & X & X & X & X & V3.6.0 & x \\
\hline Modbus TCP client (master) with ModTcpMast (PLCopen style) & V3.1.0 & X & X & X & X & V3.6.0 & X \\
\hline \begin{tabular}{l}
Remarks: \\
\({ }^{1}\) ): in preparation \\
\({ }^{2}\) ): feature is licensed
\end{tabular} & & & & & & & \\
\hline
\end{tabular}

\section*{Default open Ethernet ports of PM56xx-2ETH}

After startup without a PLC project the PM56xx-2ETH contains the following Ethernet ports and sockets:
\begin{tabular}{|l|l|}
\hline Protocol & Port \\
\hline ABB NetConfig \({ }^{1}\) ) & UDP 24576 \\
\hline Online access with driver 3S Tcp/lp BlkDrvTcp (no scan) & TCP 11740 \\
\hline OPC UA server \({ }^{2}\) ) & TCP 4840 \\
\hline
\end{tabular}

\section*{Remarks:}
\({ }^{1}\) ): The port 24576 for ABB NetConfig protocol can be disabled via PLC configuration by deleting the protocol node from configuration tree of Ethernet interfaces ETH1 and ETH2.
\({ }^{2}\) ): The port 4840 for OPC UA server is closed by default as of SystemFW V3.6.0.

All other ports are closed by default.

Overview of protocols, sockets and ports
\begin{tabular}{|c|c|c|}
\hline Protocol & Port & Sockets \\
\hline ABB netConfig & 24576 & 1 permanent socket per interface \\
\hline 3S gateway client (e.g. CODESYS) to gateway server & 1217 & 1 permanent socket \\
\hline Online access with driver 3S block driver TCP/IP (no scan) & 11740 & 1 socket per connection + 1 listen \\
\hline Modbus TCP server & 502 or configurable & 1 socket listen +1 socket per server connection, number of server connections is configurable in AB \\
\hline Modbus TCP client with POU ModTcpMast & Random & 1 socket per connection with POU ModTcpMast \\
\hline UDP out of user program with library SysLibSockets.lib & 1 ... 65535 & 1 socket per connection \\
\hline TCP/IP out of user program with library SysLibSockets.lib & 1 ... 65535 & 1 socket per connection \\
\hline Web server on PLC with web visualization & 80 & 1 listen and 1 per connection \\
\hline NTP/SNTP client & 123 & 1 permanent socket \\
\hline IEC60870-5-104 control station & Random & 1 per connection \\
\hline IEC60870-5-104 substation & 2404 & 1 per connection \\
\hline FTP server & \begin{tabular}{l}
Command port \(=21\) \\
Data active mode \(=20\) \\
Data passive mode = random
\end{tabular} & 1 per session, max. 4 allowed \\
\hline CODESYS network variables & 1202 & (UDP broadcast) \\
\hline OPC DA server (default 3S block driver) & \[
\begin{aligned}
& \text { UDP = } 1740 \\
& \text { or } \\
& \text { TCP/IP } \\
& =11740
\end{aligned}
\] & 1 socket per connection \\
\hline OPC UA server & 4840 & 1 permanent socket \\
\hline ICMP - ping out of user project with POU ETHx_ICMP_PING DHCP & none & No socket \\
\hline DHCP & 67 & 1 socket during startup \\
\hline \begin{tabular}{l}
NTP/SNTP ((Simple) Network Time Protocol) client system solution \\
② Chapter 6.3.4.4.2 "(S)NTP client configuration" on page 1718
\end{tabular} & 123 & 1 permanent socket \\
\hline \begin{tabular}{l}
NTP/SNTP ((Simple) Network Time Protocol) server system solution \\
\# Chapter 6.3.4.4.3 "(S)NTP server configuration" on page 1720
\end{tabular} & 123 & 1 permanent socket \\
\hline
\end{tabular}
\begin{tabular}{|l|l|l|}
\hline Protocol & Port & Sockets \\
\hline \begin{tabular}{l} 
HTTPS - secure web server on PLC with \\
CODESYS web visualization \\
』y Chapter 6.3.4.7.3.2 "Secure web server" \\
on page 1727
\end{tabular} & 443 & 1 listen and 1 per connection \\
\hline \begin{tabular}{l} 
FTPS - secure FTP \\
乡 Chapter 6.3.4.7.3.3 "Secure FTP" \\
on page 1728
\end{tabular} & \begin{tabular}{l} 
Command \\
port = 21 \\
Data active \\
mode = 20 \\
Data passive \\
mode = \\
random
\end{tabular} & 1 per session, max. 4 allowed \\
\hline \begin{tabular}{l} 
Secure online access with driver 3S \\
TCP/IP BIkDrvTcp
\end{tabular} & 11740 & \begin{tabular}{l}
1 socket per connection + 1 \\
listen
\end{tabular} \\
\hline \begin{tabular}{l} 
ICMP - ping out of user project with \\
POU ETHx_ICMP_PING or EthlcmpPing \\
(PLCopen style)
\end{tabular} & None & No socket \\
\hline \begin{tabular}{l} 
Modbus TCP client (master) with POU \\
ModTcpMast (PLCopen style)
\end{tabular} & Random & \begin{tabular}{l}
1 socket per connection with \\
POU ModTcpMast
\end{tabular} \\
\hline
\end{tabular}

\section*{Limitation of connections per protocol}

The limitation for each CPU apply to all onboard Ethernet interfaces either directly on the CPU itself or also on all CM5640-2ETH interfaces that are added to this CPU.
\begin{tabular}{|c|c|c|c|c|c|}
\hline Protocol & \[
\begin{aligned}
& \text { PM5630 } \\
& \text {-2ETH }
\end{aligned}
\] & \[
\begin{array}{|l|}
\hline \text { PM565 } \\
\text { 0-2ETH } \\
\hline
\end{array}
\] & \[
\begin{aligned}
& \text { PM5670- } \\
& \text { 2ETH }
\end{aligned}
\] & \[
\begin{aligned}
& \text { PM5675- } \\
& \text { 2ETH }
\end{aligned}
\] & \[
\begin{aligned}
& \geq \text { CPU } \\
& \text { firm- } \\
& \text { ware }
\end{aligned}
\] \\
\hline Modbus TCP server (e.g. for SCADA access) & \[
\begin{aligned}
& 30 \\
& 40 \\
& 15
\end{aligned}
\] & \[
\begin{aligned}
& \hline 100 \\
& 40 \\
& 25
\end{aligned}
\] & \[
\begin{aligned}
& 100 \\
& 40 \\
& 50
\end{aligned}
\] & \[
\begin{aligned}
& \hline 100 \\
& 40 \\
& 50
\end{aligned}
\] & \[
\begin{aligned}
& 3.0 .3 \\
& 3.1 .0 \\
& 3.1 .3
\end{aligned}
\] \\
\hline Modbus TCP client with POU ModTcpMast (PLCopen style) & \[
\begin{aligned}
& 30 \\
& 30
\end{aligned}
\] & \[
\begin{array}{|l|}
\hline 100 \\
50
\end{array}
\] & \[
\begin{aligned}
& 100 \\
& 120
\end{aligned}
\] & \[
\begin{array}{|l|}
\hline 100 \\
120
\end{array}
\] & \[
\begin{array}{|l|}
\hline 3.1 .0 \\
3.1 .3
\end{array}
\] \\
\hline IEC60870-5-104 control station incl. \(2^{\text {nd }}\) connection and \(2^{\text {nd }}\) port & \[
\begin{aligned}
& 10 \\
& 5
\end{aligned}
\] & \[
\begin{aligned}
& 10 \\
& 10
\end{aligned}
\] & \[
\begin{aligned}
& 10 \\
& 20
\end{aligned}
\] & \[
\begin{aligned}
& 10 \\
& 20
\end{aligned}
\] & \[
\begin{aligned}
& 3.1 .0 \\
& 3.4 .0
\end{aligned}
\] \\
\hline IEC60870-5-104 substation incl. \(2^{\text {nd }}\) port & \[
\begin{aligned}
& 10 \\
& 5
\end{aligned}
\] & \[
\begin{aligned}
& 10 \\
& 10
\end{aligned}
\] & \[
\begin{aligned}
& 10 \\
& 20
\end{aligned}
\] & \[
\begin{array}{|l|}
\hline 10 \\
20
\end{array}
\] & \[
\begin{array}{|l|}
\hline 3.1 .0 \\
3.4 .0
\end{array}
\] \\
\hline \begin{tabular}{l}
IEC60870-5-104: No. of free tags \\
+ additional license for extension \({ }^{1}\) )
\end{tabular} & 1.000 & 5.000 & 10.000 & 10.000 & 3.4.0 \\
\hline FTP server & 4 & 4 & 4 & 4 & 3.1 .0 \\
\hline Online access with driver 3S TCP/IP BlkDrvTcp & \[
\begin{aligned}
& \mathrm{n} / \mathrm{a} \\
& 8
\end{aligned}
\] & \[
\begin{array}{|l}
\hline 4 \\
8
\end{array}
\] & \[
\begin{aligned}
& \mathrm{n} / \mathrm{a} \\
& 8
\end{aligned}
\] & \[
\begin{array}{|l|}
\hline \mathrm{n} / \mathrm{a} \\
8
\end{array}
\] & \[
\begin{array}{|l|}
\hline 3.0 .0 \\
3.1 .0
\end{array}
\] \\
\hline OPC DA server (number of connections) & \[
\begin{aligned}
& \mathrm{n} / \mathrm{a} \\
& 8
\end{aligned}
\] & \[
4
\] & \[
\begin{aligned}
& \mathrm{n} / \mathrm{a} \\
& 8
\end{aligned}
\] & \[
\begin{array}{|l|}
\hline \mathrm{n} / \mathrm{a} \\
8
\end{array}
\] & \[
\begin{array}{|l|}
\hline 3.0 .0 \\
3.1 .0
\end{array}
\] \\
\hline OPC UA server (number of connections) & \[
\begin{aligned}
& 50 \\
& 10
\end{aligned}
\] & \[
\begin{aligned}
& 50 \\
& 20
\end{aligned}
\] & \[
\begin{aligned}
& 50 \\
& 50
\end{aligned}
\] & \[
\begin{aligned}
& 50 \\
& 50
\end{aligned}
\] & \[
\begin{array}{|l|}
\hline 3.1 .0 \\
3.4 .0
\end{array}
\] \\
\hline
\end{tabular}
\begin{tabular}{|l|l|l|l|l|l|l|}
\hline Protocol & \begin{tabular}{l} 
PM5630 \\
-2ETH
\end{tabular} & \begin{tabular}{l} 
PM565 \\
0-2ETH
\end{tabular} & \begin{tabular}{l} 
PM5670- \\
2ETH
\end{tabular} & \begin{tabular}{l} 
PM5675- \\
2ETH
\end{tabular} & \begin{tabular}{l}
\(\geq\) CPU \\
firm- \\
ware
\end{tabular} \\
\hline \begin{tabular}{l} 
No. of free tags \\
+ additional license for extension \({ }^{1}\) )
\end{tabular} & 1.000 & 5.000 & 30.000 & 30.000 & 3.4 .0 \\
\hline & 500 ms & 100 ms & 50 ms & 50 ms & 3.4 .0 \\
\hline \begin{tabular}{l} 
Secure online access with driver 3S \\
TCP/IP BIkDrvTcp
\end{tabular} & 8 & 8 & 8 & 8 & 3.1 .0 \\
\hline FTPS - secure FTP server & 4 & 4 & 4 & 4 & 3.1 .0 \\
\hline \begin{tabular}{l} 
HTTPS - Integrated webserver - number \\
of connections
\end{tabular} & 4 & 8 & 12 & 12 & 3.4 .0 \\
\hline \begin{tabular}{l} 
Ethernet/IP \\
Maximum number of Input/Output
\end{tabular} & \begin{tabular}{l}
0.5 \\
\(\mathrm{kB} / 0.5\) \\
kB
\end{tabular} & \begin{tabular}{l}
0.5 \\
\(\mathrm{kB} / 0.5\) \\
kB
\end{tabular} & \begin{tabular}{l}
0.5 \\
\(\mathrm{kB} / 0.5\) \\
kB
\end{tabular} & \begin{tabular}{l}
0.5 \\
\(\mathrm{kB} / 0.5 \mathrm{kB}\)
\end{tabular} & 3.4 .1 \\
\hline Remars: & & 8 & & \\
\hline
\end{tabular}

Remarks:
\({ }^{1}\) ): in preparation

The PLC types PM5630-2ETH, PM5670-2ETH and PM5675-2ETH are available as of SystemFW 3.1.0.

\section*{Default Ethernet configuration}
\begin{tabular}{|l|l|l|l|}
\hline Module & IP Address & Netmask & Comment \\
\hline PM5xx2-x-ETH & ETH: 192.168.0.10 & 255.255 .255 .0 & \\
\hline PM50x2-T-2ETH & ETH1: 192.168.0.10 & 255.255 .255 .0 & \begin{tabular}{l} 
The Ethernet ports \\
must be configured \\
in different sub net- \\
works.
\end{tabular} \\
\hline PM56xx-2ETH & ETH1: 192.168.0.10 & 255.255 .255 .0 & \begin{tabular}{l} 
The Ethernet ports \\
must be configured \\
in different sub net- \\
works.
\end{tabular} \\
\hline CM5640-2ETH & ETH2: 192.168.1.10 & & \begin{tabular}{l} 
The Ethernet ports \\
must be configured \\
in different sub net- \\
works.
\end{tabular} \\
& \begin{tabular}{l}
192.168 .11 .10, \\
192.168 .12 .10
\end{tabular} & 255.255 .255 .0 & \\
\hline
\end{tabular}
* Changing the IP addresses using the Automation Builder
\& Changing the IP addresses using the CFG key

Online access
3S TCP/IP block driver. This driver requires at least 2 sockets:
- 1x driver "BIkDrvTcp" on port 11740
- \(1 x\) listen on port 11740 if PLC has established online connection
```

Online access can be established from:

- Automation Builder command 'Login' « Chapter 6.4.1.21.3.7.2 "Command 'Login'" on page 2624
- CODESYS OPC DA server
- Panel CP600 series

```

Each established connection needs one socket. In addition one socket on port 11740 is listening.
1. Startup the PLC. \(\Rightarrow\) One socket on port 11740 (listen).
2. Login from Automation Builder via driver "BIKDrvTcp".
\(\Rightarrow 2\) sockets on port 11740 ( 1 x online, 1 x listen)
3. Additional login out of OPC server with the same driver. \(\Rightarrow 3\) sockets on port 11740 ( \(2 x\) online, \(1 x\) listen)
4. Additional connect CP600 via driver "BlkDrvTcp".
\(\Rightarrow 4\) sockets on port 11740 ( \(3 x\) online, \(1 x\) listen)

\section*{SNTP client and server}

As of version 3.1.0 the SystemFW provides a SNTP Protocol implementation which can be used for time synchronization of PLC clock. It can be used as SNTP Client or / and SNTP Server. But only one instance of each can be executed at the same time on one PLC .

> The SNTP server is listening only on the Ethernet interface, which the protocol is configured on. It is not possible to have an SNTP server on several Ethernet interfaces.

To read diagnosis information from the SNTP protocol within an IEC application the function block PmSntpInfo can be used. This Function block is part of the library ABB_Pm_AC500.lib. It can also be used to determine the synchronization state of the PLC clock.

\section*{Using network variables in AC500 V3}

When using network variables via UDP broadcast, the default broadcast address is set to 255.255.255.255.

This will not work on PLCs with multiple Ethernet interfaces, because of undecidable routing.
Set the broadcast address to a matching subnet broadcast address, depending on which interface should be used to send the variables into the network.

Example - ETH1 with IP 192.168.0.10 netmask 255.255.255.0
- ETH2 with IP 192.168.1.10 netmask 255.255.255.0

If you want the network variables to be broadcast on ETH1, use broadcast address 192.168.0.255.

\subsection*{6.8.2.6.2 Onboard CAN configuration}

AC500 V3 PLCs provide the following methods for CAN integration:
- Onboard CAN interface
- CANopen master-slave arrangement (with CM598-CN as a master device)

Table 814: Differences in supported protocols
\begin{tabular}{|l|l|l|}
\hline & Onboard CAN & CM598-CN \\
\hline CANopen Manager & X & X \\
\hline CAN 2A/2B & X & X \\
\hline J1939 & X & \\
\hline
\end{tabular}

Onboard CAN interface is not available on AC500-eCo V3!

Supported pro- Onboard CAN interface supports the following protocols
tocols
- CANopen Manager: Connection of CI581 and CI582 without additional I/O modules
- CAN 2A/2B
- J1939

Configuration in Automation Builder is described in chapter 'CANopen' \({ }^{\Leftrightarrow}\) Chapter 6.3.2.11.1.1 "CM598-CAN - CANopen Manager communication module" on page 1521.
Further information can be found in chapter 'CAN onboard' \(\Leftrightarrow\) Chapter 6.3.2.16 "CAN onboard" on page 1594

\subsection*{6.8.2.7 Hot swap}

\subsection*{6.8.2.7.1 Preconditions for using hot swap}

\section*{Hot swap}

\section*{WARNING!}

Risk of explosion or fire in hazardous environments during hot swapping!
Hot swap must not be performed in flammable environments to avoid life-threatening injury and property damage resulting from fire or explosion.

\section*{WARNING!}

Electric shock due to negligent behavior during hot swapping!
To avoid electric shock
- make sure the following conditions apply:
- Digital outputs are not under load.
- Input/output voltages above safety extra low voltage/ protective extra low voltage (SELV/PELV) are switched off.
- Modules are fully interlocked with the terminal unit with both snap-fits engaged before switching on loads or input/output voltage.
- Never touch exposed contacts (dangerous voltages).
- Stay away from electrical contacts to avoid arc discharge.
- Do not operate a mechanical installation improperly.

\section*{NOTICE!}

\section*{Risk of damage to I/O modules!}

Hot swapping is only allowed for I/O modules.
Processor modules and communication interface modules must not be removed or inserted during operation.

H = Hot swap

\section*{Hot swap}

System requirements for hot swapping of I/O modules:
- Types of terminal units that support hot swapping of I/O modules have the appendix TU5xx-H.
- I/O modules as of index FO.

The following I/O bus masters support hot swapping of attached I/O modules:
- Communication interface modules CI5xx as of index F0.
- Processor modules PM56xx-2ETH with firmware version as of V3.2.0.

Hot swap is not supported by AC500-eCo V3 CPU!


The index of the module is in the right corner of the label.

\section*{NOTICE!}

Risk of damage to I/O modules!
Modules with index below F0 can be damaged when inserted or removed from the terminal unit in a powered system.

\section*{NOTICE!}

Risk of damage to I/O modules!
Do not perform hot swapping if any I/O module with firmware version lower than 3.0.14 is part of the I/O configuration.

For min. required device index see table below.
\begin{tabular}{|l|l|}
\hline Device & \begin{tabular}{l} 
Min. required device index for I/O module as of \\
FW Version 3.0.14
\end{tabular} \\
\hline AC522(-XC) & F0 \\
\hline Al523 (-XC) & D2 \\
\hline Al531 & D4 \\
\hline Al531-XC & D2 \\
\hline Al561 & B2 \\
\hline Al562 & B2 \\
\hline Al563 & B3 \\
\hline AO523 (-XC) & D2 \\
\hline AO561 & B2 \\
\hline AX521 (-XC) & D2 \\
\hline AX522 (-XC) & D2 \\
\hline AX561 & B2 \\
\hline CD522 (-XC) & D1 \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline Device & \begin{tabular}{l} 
Min. required device index for I/O module as of \\
FW Version 3.0.14
\end{tabular} \\
\hline DA501 (-XC) & D2 \\
\hline DA502 (-XC) & F0 \\
\hline DC522 (-XC) & D2 \\
\hline DC523 (-XC) & D2 \\
\hline DC532 (-XC) & A2 \\
\hline DC562 & D2 \\
\hline D1524 (-XC) & B2 \\
\hline D1561 & B2 \\
\hline DI562 & B2 \\
\hline D1571 & A1 \\
\hline D1572 & A3 \\
\hline DO524 (-XC) & A2 \\
\hline DO526 & A0 \\
\hline DO526-XC & B2 \\
\hline D0561 & A2 \\
\hline DO562 & B3 \\
\hline DO571 & B2 \\
\hline DO572 & A1 \\
\hline DO573 & D2 \\
\hline DX522 (-XC) & D2 \\
\hline DX531 & B2 \\
\hline DX561 & B3 \\
\hline DX571 & A1 \\
\hline FM562 & \\
\hline
\end{tabular}

\subsection*{6.8.2.7.2 Compatibility of hot swap}

Hot swap is not supported by AC500-eCo V3 CPU!
\begin{tabular}{|l|l|}
\hline & Central I/O on V3 CPU \\
\hline \begin{tabular}{l} 
I/O module on TU5xx-H connected to I/O bus \\
master
\end{tabular} & \begin{tabular}{l} 
AC500 V3 CPU types: \\
PM56xx-2ETH
\end{tabular} \\
\hline Required version of I/O bus master & Firmware as of V3.2.0 \\
\hline \begin{tabular}{l} 
Fieldbus master when used as remote I/O with \\
AC500 V3
\end{tabular} & - \\
\hline \begin{tabular}{l} 
When used as remote I/O on third party \\
controller (PLC or DCS)
\end{tabular} & - \\
\hline
\end{tabular}

\subsection*{6.8.2.7.3 Hot swap behavior}
\begin{tabular}{|l|l|}
\hline Hot swap behavior & Central I/O on V3 CPU \\
\hline \begin{tabular}{l} 
Start-up behavior with unplugged or damaged \\
I/O module on hot swap terminal unit TU5xx-H
\end{tabular} & \begin{tabular}{l} 
System and I/O modules attached to the CPU \\
are starting (except unplugged or damaged \\
module when plugged on hot swap terminal \\
unit). \\
As soon as the correct and operational I/O \\
module is plugged on the terminal unit, the \\
module is configured and ready to start. \\
No specific setting needed.
\end{tabular} \\
\hline \begin{tabular}{l} 
Start-up behavior with wrong I/O module type \\
on any terminal unit
\end{tabular} & \begin{tabular}{l} 
System and I/O modules are not starting
\end{tabular} \\
\hline \begin{tabular}{l} 
Diagnosis of presence of hot swap terminal \\
unit
\end{tabular} & \begin{tabular}{l} 
Diagnosis using PLC browser command "io- \\
bus desc" in Automation Builder V3. \\
The PLC browser then provides an overview \\
of the modules on the I/O bus including the \\
position of hot swap terminal units in the \\
I/O bus. \\
In the application program this
\end{tabular} \\
\hline can be detected with a func- \\
tion block "loModuleHotSwapInfo" \\
(Library: AC500_Io/Function Blocks/I/O-Bus). \\
One instance of function block is needed per \\
terminal unit on the I/O bus. The function \\
block provides five outputs delivering infor- \\
mation about slut number, hot swap capa- \\
bility and plugged/unplugged state of the I/O \\
module
\end{tabular}\(\left|\begin{array}{l}\text { If module is pulled then diagnosis Err 9480 } \\
\text { "Module removed from Hot Swap Terminal } \\
\text { Unit" is generated }\end{array}\right|\)\begin{tabular}{l} 
Diagnosis Err 9480 is automatically \\
acknowledged
\end{tabular}

\subsection*{6.8.2.8 KNX IP integration}

\subsection*{6.8.2.8.1 Introduction}

KNX is a bus system used more on the room and floor level in buildings (e.g. for lighting, shading and local HVAC devices).
The KNX as such doesn't necessarily need a dedicated controller for simple connection of sensor/switch to receiving/actuator devices.
The signals exchanged via the protocol are so called "group addresses" ("objects"), which are downloaded via ETS to all the thereby linked (=grouped) devices.
On the room level it typically has a serial wiring called KNX TP (twisted pair), which then is linked to floor or central building or management level via IP routers. On Ethernet it is called KNXnet/IP abbreviated also as KNX IP.


After the engineering in Automation Builder, the AC500 V3 PLC is a standard KNX device. KNX communication is done via the IP network interface. The PLC is arranged topologically on the area / main line of the KNX IP routers. The PLC communicates with the routers via the KNXnet/IP protocol.

\subsection*{6.8.2.8.2 Engineering workflow}

Both engineering software systems for AC500 V3 PLC (Automation Builder) and KNX (ETS) are directly linked.

A data exchange for the group objects (hereinafter also called communication objects) from the Automation Builder to ETS (via an XML file) and received by a DCA (Device configuration APP) for ETS is available.

The AC500 V3 PLC is integrated into the ETS via a certified KNX "device" with the transferred group objects as configured in Automation Builder and a physical KNX address (transferred via ETS and KNX IP to the AC500 V3 PLC).


Programming and commissioning of the AC500 V3 PLC starts with Automation Builder:

Configuration of the AC500 V3 PLC, its communications, here KNX, and I/O modules and all necessary parameters.
1. Configure programmable KNX controller in Automation Builder by adding group objects to the device.
2. Use group objects as inputs and outputs in the IEC application.
3. Download of the above into the AC500 V3 PLC (via the engineering interface)
4. Export of the group objects for ETS via XML file.

The subsequent linking of the AC500 V3 PLC and the other KNX devices takes place with the vendor independent KNX commissioning software ETS:
5. Install DCA Plugin and the AC500 V3 PLC device description in ETS
6. Connect group objects in ETS and assign group addresses.
7. KNX IP download to AC500 V3 PLC.
8. The physical KNX address of the AC500 V3 PLC must be set before or during download of the KNX configuration.

The programming of the AC500 V3 PLC and the KNX commissioning can be done also by different people at different times and with same or separate engineering PCs. Both projects carry out their own download parts of their respective configurations to the AC500 V3 PLC.
The only data exchange between the two Engineering programs are the details about the KNX group objects defined in the ABB Automation Builder. This is done flexibly via the XML configuration file.

\subsection*{6.8.2.8.3 Prerequisites}
- Basic AC500 and Automation Builder know-how.
- Basic experience and expertise in use of KNX and ETS (engineering software for KNX).
- \(\mathrm{PC}(\mathrm{s})\) with Windows 7 or higher with administrator right(s)
- At least temporary network access to the internet for downloading and installing of:
- Automation Builder as of version AB 2.1.2, (and e.g. example .project)
- ETS5 and the necessary additional files (DCA .etsapp, device description .knxprod) plus possibly an matching example .knxproj) to above Automation Builder .project
- Network access to the local network, were the AC500 V3 PLC and KNX devices are connected.
- PS5604-KNX AC500 runtime license for each dedicated AC500 V3 PLC used in KNX networks.
- The current IP address of the engineering PC(s) where Automation Builder and ETS are located in same Network / masked IP range, as the AC500 V3 PLC to be used.

\subsection*{6.8.2.8.4 General settings and system behavior}

The KNX interface at the AC500 V3 PLC is only active during the PLC is in RUN.
1. Download Automation Builder program.
2. Run PLC.
3. Set physical address or download KNX application Programm via ETS.
\(\Rightarrow\) The bus status can be viewed in Online View of Automation Builder.

KNX communication is only working after download the matching ETS application to the AC500 V3 PLC. Until then, the AC500 V3 PLC KNX communication is deactivated and marked with a warning symbol.


Fig. 359: KNX Interface not ready
However in this state the AC500 V3 PLC can still be switched to the KNX programming mode and the physical KNX address can be programmed. Also the device info can be read by ETS.
If the KNX interface is ready, this can be recognized by the green symbol on the KNX interface in the Automation Builder.


Fig. 360: KNX Interface ready

\subsection*{6.8.2.8.5 Start-up behavior}

\section*{Start/Stop PLC}


KNX bus works only in RUN mode.
If the PLC is in "STOP" mode the KNX bus and the outputs are reset.

To avoid this behavior in "STOP" mode set the following preferences at the PLC_AC500_CPU:
1. Double-click PLC_AC500_V3 < ...> and click PLC Settings.
2. Enable checkbox Update IO while in stop and select in dropdown-menu Behavior for Outputs in Stop "Keep current values".

Warm start / If the PLC is reset also the connected objects will be reset on the KNX bus.

\section*{Cold start}

Power ON/Off
After Power ON the KNX Interface need approximately 1 s to start after the PLC program had started. During this period no inputs will be recognized by the PLC and no outputs will be send to the bus.
6.8.2.8.6 Engineering of KNX in Automation Builder

Creation of KNX group objects


The data exchange with the KNX bus is done via KNX group objects.
1. Double-click node "KNX" in the device tree "click General \(\rightarrow\) click Add".
\(\Rightarrow\) The window Communication object appears.
2. Enter your properties:
- Group Object Number:

The number of the KNX Group Object must match within the controller. It is displayed in the ETS and influences the display order in the ETS and the Automation Builder.
- Type:

Selection of the communication direction.
- Input means that the controller receives values from the KNX bus.
- Output means that the controller sends values to the KNX bus.
- Data Point Type:

Specification of the KNX data point type (DPT) of the Group Object. This determines the memory size, scaling and unit. For further information see the KNX Standard.
- Group Object Name:

The name of the KNX Group Object. It is freely selectable and is displayed in the ETS under the field name.
- Group Object Function:

The name of the function of the Group Object. It is freely selectable and is displayed in the ETS under the field Function.

Based on this selection, the flags of the KNX Group Object are set accordingly in the ETS.
You can use the [Export CSV...] button in the "General" tab menu bar to display the list of KNX group objects in a spreadsheet program such as Excel and edit and extend it flexibly. Then you can import them again via [Import CSV].

After you have created all the required KNX group objects, export them using the [Export to ETS] button. This exported file contains the configuration of the KNX group objects of the AC500 V3 PLC and is imported by ETS for linking to other KNX devices. If you have not yet created project information under main menu "Project \(\rightarrow\) Project Information", the default values will be used during the export.

To use these KNX group objects in your application program, you must assign them with IEC61131-3 variables. This additional abstraction layer of an additional variable allows you to create modular automation programs that are independent of the used bus system or input / output modules.
The assignment is possible either via the parameter page "KNX I/O Mapping" or "//O mapping list". Both editors offer the same function in different representations.
On the KNX I/O Mapping page, the KNX variables are shown hierarchically. Each KNX Group Object consists of several channels with additional information. These differ depending on whether it is an input or an output.
The view is structured as follows:
- Variable:

Enter the name of the IEC 61131-3 variable that you want to assign to this channel (KNX Group Object).
- Mapping

Shows if the channel is already linked
- Channel:

Name of the Channel (Channel name)
- Address:

The memory address under which the information is stored in the memory of the AC500 V3 PLC. Inputs start with \%I and outputs start with \%Q.
- Type:

Specification of the IEC 61131-3 variable type
- Default Value:

The value used after starting the controller.
- At a KNX Group Object input, this value is used by the automation program until a value has been received from the KNX bus.
- At a KNX Group Object output, this value is sent to the bus when the controller is started.
- Unit:

Specification of the KNX data point type (DPT)
- Description:

Note text
A KNX Group Object "input" consists of a status and a control part:
The Channel name of the status part consists of: Object Number + Object Name + Object Function and include the following informations:
- UpdateFlag:

This status flag is set to the value "true" for one cycle as soon as a new KNX telegram has been received. Even if the value of the telegram does not differ from the previous one.
- ValueChanged:

This status flag is set to the value "true" as soon as a new KNX telegram has been received and the value differs from the previous one.
- ValueValid:

This status flag is set to the value "true" as soon as a KNX telegram has been received for the first time after the controller has been started.
- WatchdogTimeout:

As of Automation Builder 2.2.1 it will be possible to define a Watchdog Timeout for each input object. If a timeout occur this flag will be set to the value "true" for one cycle.
- Value:

The current value of the KNX Group Object received from the KNX bus.
The Channel name of the control part consists of: "Control" + Object Number + Object Name + Object Function and include the following control possibilities:
- Reset status flags:

When this flag is set from "false" to "true" by the automation program then the above-mentioned status flags of the KNX Group Object are reset to the value "false".
- Send value read:

When this flag is set from "false" to "true" by the automation program, a ValueRead telegram is sent to the KNX bus. This causes the KNX remote device to send back its current value.
A KNX Group Object "output" is represented as follows:
The Channel name of the Group Object consists of: Object Number + Object Name + Object Function
- Trigger Output:

When this flag is set from "false" to "true" by the automation program, the current value is sent immediately to the KNX bus. The sending conditions that are may be activated in the ETS (send on change and cyclic sending) will be restarted
- Disable Output:

As long this flag is set to "true" by the automation program, the sending conditions send on change and cyclic sending in the ETS are deactivated.
- Value:

The current value of the KNX Group Object that is sent to the KNX bus.
The permanently defined Program LED Status represent the function as known in other KNX devices, showing the status of the programming LED.

\section*{Create an application program}

The KNX variables defined on the KNX I/O Mapping page are available programwide under loConfig_Globals_Mapping.

These you can see if you click in to the programming window and either via right-click select "Input Assistant" or press F2.


\section*{Export XML file}

To exchange the configured KNX group objects the configuration has to be exported via XML file.

If later both projects (from Automation Builder and ETS) are loaded on the PLC, the PLC checks if the two projects have the same source and fit together. This will be done by an automatically calculated Checksum. For calculating the Checksum the following information's from the Project information will be used:
- Company
- Title
- Version
- Timestamp

This Information will also be shown in the ETS after loading the XML file. If the user has not entered any project information some default values will be set.

\subsection*{6.8.2.8.7 Integration of the PLC in KNX}

\section*{Insert controller}
1. Start the ETS and insert the PS5604-KNX AC500 as controller from the ETS device catalog into your ETS project.
2. Assign a physical KNX address to the controller.
\(\Rightarrow\) The controller is placed topologically on the IP Main Area.
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline \multicolumn{8}{|l|}{Fi:ETS5 \({ }^{\text {m }}\) - PS5604-KNX_V2-1-2_Exampleproject} \\
\hline ETS & \multicolumn{2}{|l|}{Edit Workplace} & Commissioning & \multicolumn{2}{|l|}{Diagnostics Apps} & \multicolumn{2}{|l|}{Window} \\
\hline ( & Project & & Undo Redo & \(\square \mathrm{R}\) & Reports & [|El| Workplace * & Catalogs \\
\hline \multicolumn{8}{|l|}{Topology} \\
\hline \multicolumn{8}{|l|}{H Add Areas | X Delete 2 Download * (i) Info * Reset \(\%\) Unload * Print} \\
\hline \multicolumn{8}{|l|}{111 Topology *} \\
\hline \multicolumn{8}{|l|}{\[
\begin{aligned}
& \text { Ey Dyamic Folders } \\
& \text { 」暀 } 1 \text { IP Main Area }
\end{aligned}
\]} \\
\hline \multicolumn{8}{|l|}{1 1.0.1 PS5604-KNX AC500 KNX runtime license} \\
\hline \multicolumn{8}{|l|}{1.1 TP Linie 1
1.1.- SU/S30.640.1 Power supply, \(640 \mathrm{~mA}, \mathrm{MDRC}\)
1.1.0 IPR/S3.1.1 IP-Router,REG
1.1.6 SA/S4.10.1 Switch Actuator,4-fold,10A,MDRC} \\
\hline
\end{tabular}

Import configuration

1. Select the "PS5604-KNX AC500" in the ETS explorer tree and click on "DCA" tab in the editor window.
2. Click on Load Configuration and select the configuration XML file.
\(\Rightarrow\) The KNX group objects defined in AC500 V3 PLC in Automation Builder are displayed in the ETS.

\section*{Connect controller with KNX devices}
1. Right-click on a "PS5604-KNX AC500" group object and assign a KNX group address or drag and drop from group address window.
2. Interlink group objects by assigning the same KNX Group Address.

\section*{Parameters of the device}

The following settings are possible in the ETS parameters of the PS5604-KNX AC500.

Tab General settings
\begin{tabular}{|l|ll} 
General Settings & Default Gateway & \(\boxed{239.0 .0 .0}\) \\
\hline \multirow{3}{*}{ Object \(1 . .3\)} & Telegram rate & Max. 10 telegrams per second \\
& & \\
& Project Title & CODESYS KNX \\
& Application date & \(2018-11-13 T 12: 01: 42.60356\) \\
& Identifier & 252617197 \\
& Version & 0.0 .0 .1 \\
& Application state & undefined \\
& Description & \\
\hline
\end{tabular}
- Default Gateway:

The used KNXnet/IP broadcast address. This must match the KNX system (KNX IP router). It is the default setting that is usually not changed.
- The entry field Default Gateway can contain another IP address for the Multicast communication. The normal Multicast IP address for KNX ist 224.0.23.12.
- If another Multicast IP address is to be used, it can be chosen in the area from 239.0.0.0 to 239.255.255.255. This alternative Multicast IP address can be defined in the input field Default Gateway.
- Telegram rate:

The maximum transmission rate of the AC500 V3 PLC can be limited in order to prevent an excessive bus load and thus to avoid malfunction of the KNX system.
The KNX telegrams are buffered until they have been sent. New values which have been calculated by the automation program in the meantime are updating the cached values. The old cached value is discarded and not sent.
- Project Information:

At this point, the project information of the Automation Builder project is displayed.

\section*{Tab Object 1 .. 3}
1.0.1 PS5604-KNX AC500 KNX runtime license > Object 1 .. 3


For each KNX Group Object of the AC500 V3 PLC an Object entry is displayed in the device parameters. This is named after the number of the KNX Group Object.
For outputs (controller sends to the KNX bus) the KNX transmission conditions can be set:
- Communication direction

Setting of the transmission direction of the object.
- Input (KNX to PLC): The controller receives values from the KNX bus.
- Output (PLC to KNX): The controller sends values to the KNX bus.
- Send condition (only for outputs):

Setting whether the Controller sends a telegram to the KNX bus automatically when the object value is changed. The following options are available
- No automatic sending:

No automatic sending to the KNX bus. This must be done via the program code by the Trigger Output flag.
- Send on change: Every time the object value changes, a telegram is sent to the KNX bus. No matter how minor this change is.
- Send on difference (only for group objects which are not DPT 1.* Boolean): Every time the object value changes, this value is only sent to the KNX bus if it differs from the last sent value at least by the settable difference.
- Sending difference (only if Send on difference is active):

Input of the difference by which the object value must change to be send. You can enter numbers with decimal places.
- Cyclic sending (only for outputs):

Setting whether in addition the object value is sent cyclically repeatedly to the bus. This also happens if this object value has not changed. Two different range of values for the cycle time can be specified.
- Cycle time (only when Cyclic sending is active):

Specification of the cycle time for the cyclic transmission.
Input format:
hour:minute:second
Note: The cycle time of the KNX stack depends on the cycle time of the task that executes the stack. A long task time causes long download times from ETS. Consider the CPU load and cycle times of other processes running on the CPU when selecting a cycle time for the KNX stack.

Regardless of the set transmission conditions, the program code can trigger by the flag Trigger Output a sending of the value to the KNX bus at any time.

By activating the flag Read on Init of the KNX group objects in the right ETS properties panel, the Controller sends a value read query to the connected KNX device at startup. This then responds with its current object value.

In this properties panel you can also select the appropriate subdata point type of the KNX Group Object. This defines the unit of the value in the KNX system. For example DPT 9.001 represents temperature in \({ }^{\circ} \mathrm{C}\).
\begin{tabular}{|c|c|c|}
\hline \multicolumn{2}{|l|}{Eit Properties} & > \\
\hline Settings & Comm. &  \\
\hline \begin{tabular}{l}
Name \\
Room 2 La \\
Descriptio
\end{tabular} & state & \\
\hline \multicolumn{3}{|l|}{Aktor A Status} \\
\hline \multicolumn{3}{|l|}{Priority} \\
\hline Low & & - \\
\hline \multicolumn{3}{|l|}{Flags} \\
\hline  & cation

Init & \\
\hline \multicolumn{3}{|l|}{Data Type} \\
\hline \multicolumn{2}{|l|}{\begin{tabular}{l}
1.001 switch \\
1.002 boolean \\
1.003 enable \\
1.004 ramp \\
1.005 alarm \\
1.006 binary value \\
1.007 step \\
1.008 up/down \\
1.009 open/close
\end{tabular}} & \(\wedge\) \\
\hline Default & & \\
\hline
\end{tabular}

If for example the response of an actuator state is needed for an input "Aktor A Status", this feature can be enabled in the parameter of the Switch Actuator (e.g. 1.1.6 SA/S4.10.1).


The current IP address as well as further information of the AC500 V3 PLC can be read via the ETS Device Info function. For this the physical KNX address is necessary. You can determine the address by the ETS function Programming Mode.
With the ETS function Group Monitor you can analyze the telegrams on the KNX bus. You can also use it to write/read KNX telegrams.

\section*{Download ETS configuration to controller}

The download of the ETS configuration to the AC500 V3 PLC is done via the ETS function "Download" in the menu bar. This download happens via the KNX interface directly to the AC500 V3 PLC.

Best you select in ETS the network interface of the computer as the bus interface. Thus, a fast data exchange is possible and the data is not routed via the KNX TP bus.

At the first download, the physical KNX address of the controller is programmed. To do this, set the AC500 V3 PLC to KNX programming mode.

This can be done either via the display or functions inside the application program of the controller (e.g. connected to a Webvisu like done in the example program).

Via AC500 display

²) Chapter 6.8.2.5.5.5.2 "Configuration with SystemFW >=V3.5.0 and DisplayFW >=V4.1" on page 4466

\section*{Attention!}

The activation of the KNX programming mode via the display only works with Automation Builder as of version 2.2.0.

The KNX configuration of AC500-eCo V3 uses TA5130-KNXPB.


The AC500 V3
PLC must be in RUN mode.
1. Press the \(\mathbf{C F G}\) function key.
\(\Rightarrow\) Switch is OFF (S OFF) is displayed.
2. Press the Arrow Down function key
\(\Rightarrow\) Pbut 0 is displayed.
(Pbut is standing for progamming button, the 0 (or the 1 ) showing the status of the programming LED ( \(0=0 \mathrm{ff} ; 1=\mathrm{ON}\) ))
3. Press the CFG function key.
\(\Rightarrow\) The display shows Pbut 1 flashing.
4. Confirm this with the \(O K\) function key.
\(\Rightarrow\) The display permanently shows Pbut 1. The AC500 V3 PLC is in KNX programming mode.

AC500-eCo V3
via TA5130-
KNXPB
1. Ensure the TA5130-KNXPB is plugged in during power up, PLC is in RUN mode.
2. Ensure that TA5130-KNXPB is configured inside Automation Builder configuration.
3. Push the button on the TA5130-KNXPB.
\(\Rightarrow\) The LED on the option board should turn ON. The AC500-eCo V3 is in KNX programming mode.

The AC500 V3 PLC automatically terminates the KNX programming mode after the programming of the physical KNX address.

Alternatively you can terminate the programming mode with Pbut 0 by pressing the CFG function key.
For AC500-eCo V3: push the button on the TA5130-KNXPB again to terminate the programming mode, the LED should turn OFF.

You can exit the menu at any time with the ESC function key.

Via AC500 appli- Please use the following variable for setting the KNX Program Button: cation

AC500_IoDrvKNX.GVL.IoDrvKNXCopyChannels.ProgramButton
The controller automatically terminates the programming mode after programming the physical address with the ETS.
The AC500 V3 PLC has then besides the Automation Builder configuration also the appropriate ETS configuration and starts its KNX communication.
Download all other linked KNX devices as well as the KNX IP routers. The ETS automatically creates the filter tables of the KNX IP routers so that the KNX telegrams are routed from the KNX TP lines to the IP line of the AC500 Controller.

\subsection*{6.8.2.8.8 Make changes}

Changes can be made in the Automation Builder as well as in the ETS without the need for a change in the other software or the need for a new data exchange.

Only if changes are made to the KNX group objects in the Automation Builder, a data exchange with the ETS is again necessary. Afterwards, a download is required both in the Automation Builder and in the ETS. Only when these two configurations have been downloaded again to the AC500 V3 PLC, the KNX communication is in operation again.
The DCA detects changes to names and numbers of the KNX group objects when importing the configuration file in the ETS and keeps the already made settings and linked Group Addresses of these changed group objects.

\subsection*{6.8.2.8.9 Data conversion}

The KNX standard defines a big-endian byte order while the IEC 61131-3 is based on the little-endian byte order. Therefore, the controller automatically converts the data point types.
However, if you access the bits of the structured KNX data point types (DPT) for time, date (DPT 10.*, DPT 16.*, DPT 19.*) in your program code, you have to note the reverse byte order. Therefore, as of Automation Builder version 2.2.0, corresponding function libraries are available that provide conversion functions for these data point types.


\subsection*{6.8.2.8.10 Further information}
- Example projects can be found in the example folder, "Help \(\rightarrow\) Project examples".

Prerequisite: Install the related library package with Automation Builder Installation Manager - [Modify].
- Programming in IEC61131 is described in the application example AC500 V3 KNX IP Integration.
- Further information can be found on the ABB website: \(\underline{A B B}\) products and services.

\subsection*{6.8.2.9 Communication with Modbus RTU}

\subsection*{6.8.2.9.1 Protocol description}

The Modbus RTU protocol is implemented in the AC500 processor modules. Modbus is a master-slave (client-server) protocol. The client sends a request to the server and receives the responses.

The Modbus operating mode of a serial interface is set in the PLC configuration \({ }^{\Perp}\) Chapter 6.3.2.14.2 "Configuring Modbus RTU on serial interface" on page 1588.

To use the AC500 V3 or AC500-eCo V3 as a Modbus RTU client two possible programming methods can be used:
- ModRtuMast
- ModRtuToken with Read/Write function blocks

\section*{ModRtuMast}

With this method the Modbus client function block ModRtuMast is used to program all read and/or write jobs in a sequence. If more than one ModRtuMast function block is used, e.g. for several sever connections, the sequence must take care that only one of those ModRtuMast function blocks is active at a time.
A summary of the function codes that are supported for Modbus RTU communication with an AC500 PLC is given in the application example Modbus RTU and function block ModRtuMast.

ModRtuToken with Read/Write function blocks

With this method the function block ModRtuToken is used along with ModRtuRead and/or ModRtuWrite and/or ModRtuReadWrite23.

Instead of programming a sequence a simple connection via structure variables between the function blocks is necessary to ensure the sequencial functionality of Modbus RTU.

A sample project for Modbus client and server in order to enable Modbus RTU communication with an AC500 PLC is given in the application example Modbus RTU and function block ModRtuMast.


To use Modbus RTU protocol on an AC500-eCo V3 PLC, the CPU must be equipped with an option board for COMx serial communication TA5141-RS2321, TA5142-RS485 or TA5142-RS485I option board. The type of the option board adapter must be selected according to the type of physical serial interface needed.

According to the CPU type, up to 3 option boards for COMx serial communication can be used \(\stackrel{y}{ } \Leftrightarrow\) Chapter 6.3.2.8.4 "Attach an option board for COMx serial communication" on page 1505.

Modbus RTU can be used for communication between an AC500-eCo V3 PLC and a control panel. The connection principle is described using the CP604 control panel as an example: CP604 connection via Modbus RTU.

Modbus client In this operating mode, the telegram traffic with the server(s) is handled via the function block ModRtuMast.
This function block sends Modbus request telegrams to the server(s) via the set interface and receives Modbus response telegrams from the server(s) via this interface.

The Modbus blocks transferred by the server contain the following information:
- Modbus address of the interrogated server (1 byte)
- Function code that defines the request of the client (1 byte)
- Data to be exchanged ( \(n\) bytes)
- CRC16 control code (2 bytes)

Modbus server In this operating mode, no function block is required for Modbus communication. Sending and receiving Modbus telegrams is performed automatically.

The AC500 CPUs process the following Modbus operation codes:
\begin{tabular}{|l|l|l|}
\hline \multicolumn{2}{|l|}{ Function code } & HEX \\
\multirow{2}{|l|}{ Description } \\
\hline DEC & 01 or 02 & Read n bits \\
\hline 01 or 02 & 03 or 04 & Read n words \\
\hline 03 or 04 & 05 & \begin{tabular}{l} 
Write one bit (encoded in one \\
word)
\end{tabular} \\
\hline 05 & 06 & Write one word \\
\hline 06 & \(0 F\) & \begin{tabular}{l} 
Write n bits (encoded in one \\
byte)
\end{tabular} \\
\hline 15 & 10 & Write n words \\
\hline 16 & 16 & Mask write \\
\hline 22 & 17 & \begin{tabular}{l} 
Read/write multiple words in \\
one telegram
\end{tabular} \\
\hline 23 & & \\
\hline
\end{tabular}

The following restrictions apply to the length of the data to be sent:
\begin{tabular}{|c|c|c|}
\hline \multicolumn{2}{|l|}{Function code} & \multirow[t]{2}{*}{Max. length} \\
\hline DEC & HEX & \\
\hline 01 or 02 & 01 or 02 & 2000 bits \\
\hline 03 or 04 & 03 or 04 & 125 words / 62 double words \\
\hline 05 & 05 & 1 bit \\
\hline 06 & 06 & 1 word \\
\hline 15 & OF & 2000 bits \\
\hline 16 & 10 & 123 words / 61 double words \\
\hline 22 & 16 & Write: 1 word \\
\hline 23 & 17 & \begin{tabular}{l}
Read: 125 words / 62 double words \\
Write: 121 words / 60 double words
\end{tabular} \\
\hline
\end{tabular}

\subsection*{6.8.2.9.2 Technical data}

Setting of the Modbus operating mode and the interface parameters

Table 815: Description of the Modbus protocol
\begin{tabular}{|c|c|}
\hline Parameter & Value \\
\hline Supported standard & * Chapter 6.3.2.14.2 "Configuring Modbus RTU on serial interface" on page 1588 \\
\hline Number of connection points & \begin{tabular}{l}
1 client \\
Max. 1 server with RS-232 interface \\
Max. 31 servers with RS-485
\end{tabular} \\
\hline Protocol & Modbus \\
\hline Operating mode & Client/server \\
\hline Address & Server only \\
\hline Data transmission control & CRC16 \\
\hline Data transmission speed & From 9,600 bits/s to 115,200 bits/s «2 Chapter 6.3.2.14.2 "Configuring Modbus RTU on serial interface" on page 1588 \\
\hline Encoding & \begin{tabular}{l}
1 start bit \\
8 data bits \\
1 or 2 stop bits \\
1 parity bit \\
« Chapter 6.3.2.14.2 "Configuring Modbus \\
RTU on serial interface" on page 1588)
\end{tabular} \\
\hline
\end{tabular}

Table 816: Max. cable length
\begin{tabular}{|l|l|}
\hline Parameter & Value \\
\hline \begin{tabular}{l} 
Max. cable length for RS-485 on COM1 for \\
AC500 CPU
\end{tabular} & 1.200 m at 19.200 baud \\
\hline
\end{tabular}

\subsection*{6.8.2.9.3 Modbus addresses for AC500-eCo V3 processor modules PM50x2}

Modbus A range of maximum 64 kB is allowed for the access via Modbus to the addressable flag area address table (\%M area). Thus, the complete address range 0000hex up to 7FFFhex is available for Modbus.

The availability of the segments depends on the CPU. The size of the \%M area can be found in the technical data of the CPUs and in the target system settings.

Inputs and outputs cannot be directly accessed using Modbus.
Following values apply:
\begin{tabular}{|c|c|c|c|c|c|}
\hline & PM5012-x-ETH & PM5032-x-ETH & PM5052-x-ETH & \begin{tabular}{l}
PM5072- \\
T-2ETH(W)
\end{tabular} & PM5082-T-2ETH \\
\hline Size of the \%M area & 4 kB & 16 kB & 16 kB & 64 kB & 64 kB \\
\hline \multicolumn{6}{|l|}{Modbus address range (Word accesses)} \\
\hline HEX & 0000 ... 07FF & 0000 ... 1FFF & 0000 ... 1FFF & 0000 ... 7FFF & 0000 ... 7FFF \\
\hline DEC & 0000 ... 2047 & 0000 ... 8191 & 0000 ... 8191 & 0000 ... 32767 & 0000 ... 32767 \\
\hline Byte & \begin{tabular}{l}
\%MB0 ... \\
\%MB4097
\end{tabular} & \%MB0 ... \%MB16382 & \%MB0 ... \%MB16382 & \%MB0 ... \%MB65534 & \%MB0 ... \%MB65534 \\
\hline Word & \%MWO ... \%MW2047 & \%MW0 ... \%MW8191 & \%MW0 ... \%MW8191 & \%MWO ... \%MW32767 & \%MW0 ... \%MW32767 \\
\hline
\end{tabular}
6.8.2.9.4 Modbus addresses for AC500 V3 processor modules PM56xx

\section*{Modbus address table}

Table 817: Modbus addresses (word accesses)
\begin{tabular}{|c|c|c|c|c|c|}
\hline \multicolumn{2}{|l|}{Modbus address} & \multirow[t]{2}{*}{\begin{tabular}{|l|l|} 
Byte \\
BYTE
\end{tabular}} & \multirow[t]{2}{*}{Bit (byte-oriented)
BOOL} & \multirow[t]{2}{*}{Word WORD} & \multirow[t]{2}{*}{Double word DWORD} \\
\hline HEX & DEC & & & & \\
\hline \multirow[t]{2}{*}{0000} & \multirow[t]{2}{*}{0} & \%MB0 & \[
\begin{aligned}
& \text { \%MX0.0 ... } \\
& \text { \%MX0. }
\end{aligned}
\] & \multirow[t]{2}{*}{\%MW0} & \multirow[t]{4}{*}{\%MD0} \\
\hline & & \%MB1 & \[
\begin{aligned}
& \text { \%MX1.0 ... } \\
& \text { \%MX1.7 }
\end{aligned}
\] & & \\
\hline \multirow[t]{2}{*}{0001} & \multirow[t]{2}{*}{1} & \%MB2 & \[
\begin{aligned}
& \text { \%MX2.0 ... } \\
& \text { \%MX2. }
\end{aligned}
\] & \multirow[t]{2}{*}{\%MW1} & \\
\hline & & \%MB3 & \[
\begin{aligned}
& \text { \%MX3.0 ... } \\
& \text { \%MX3.7 }
\end{aligned}
\] & & \\
\hline \multirow[t]{2}{*}{0002} & \multirow[t]{2}{*}{2} & \%MB4 & \[
\begin{aligned}
& \text { \%MX4.0 ... } \\
& \text { \%MX4. }
\end{aligned}
\] & \multirow[t]{2}{*}{\%MW2} & \multirow[t]{4}{*}{\%MD1} \\
\hline & & \%MB5 & \[
\begin{aligned}
& \text { \%MX5.0 ... } \\
& \text { \%MX5. }
\end{aligned}
\] & & \\
\hline \multirow[t]{2}{*}{0003} & \multirow[t]{2}{*}{3} & \%MB6 & \[
\begin{aligned}
& \text { \%MX6.0 ... } \\
& \text { \%MX6.7 }
\end{aligned}
\] & \multirow[t]{2}{*}{\%MW3} & \\
\hline & & \%MB7 & \[
\begin{aligned}
& \text { \%MX7.0 ... } \\
& \text { \%MX7.7 }
\end{aligned}
\] & & \\
\hline \multicolumn{6}{|l|}{\(\ldots\)} \\
\hline \multirow[t]{2}{*}{7FFE} & \multirow[t]{2}{*}{32766} & \%MB65532 & \[
\begin{aligned}
& \text { \%MX65532.0 } \\
& \text { \% } \\
& \text { \%MX6532.7 }
\end{aligned}
\] & \multirow[t]{2}{*}{\%MW32766} & \multirow[t]{4}{*}{\%MD16383} \\
\hline & & \%MB65533 & \[
\begin{array}{|l|}
\hline \text { \%MX65533.0 } \\
\ldots \\
\text { \%MX65533.7 } \\
\hline
\end{array}
\] & & \\
\hline \multirow[t]{2}{*}{7FFF} & \multirow[t]{2}{*}{32767} & \%MB65534 & \[
\begin{aligned}
& \text { \%MX65534.0 } \\
& \text { \% } \\
& \text { \%MX65534.7 }
\end{aligned}
\] & \multirow[t]{2}{*}{\%MW32767} & \\
\hline & & \%MB65535 & \%MX65535.0
\%MX65535.7 & & \\
\hline \multirow[t]{2}{*}{8000} & \multirow[t]{2}{*}{32768} & \%MB65536 & \%MX65536.0
\%MX65536.7 & \multirow[t]{2}{*}{\%MW32768} & \multirow[t]{4}{*}{\%MD16384} \\
\hline & & \%MB65537 & \[
\begin{aligned}
& \text { \%MX65537.0 } \\
& \text { \%MX65537.7 }
\end{aligned}
\] & & \\
\hline \multirow[t]{2}{*}{8001} & \multirow[t]{2}{*}{32769} & \%MB65538 & \[
\begin{aligned}
& \text { \%MX65538.0 } \\
& \text { \%MX65538.7 }
\end{aligned}
\] & \multirow[t]{2}{*}{\%MW32769} & \\
\hline & & \%MB65539 & \[
\begin{aligned}
& \text { \%MX65539.0 } \\
& \text { \%MX65539.7 }
\end{aligned}
\] & & \\
\hline 8002 & 32770 & \%MB65540 & \[
\begin{aligned}
& \text { \%MX65540.0 } \\
& \text { \% } \\
& \text { \%X65540.7 }
\end{aligned}
\] & \%MW32770 & \%MD16385 \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|}
\hline \multicolumn{2}{|l|}{Modbus address} & \multirow[t]{2}{*}{\begin{tabular}{l}
Byte \\
BYTE
\end{tabular}} & \multirow[t]{2}{*}{Bit (byte-oriented)
BOOL} & \multirow[t]{2}{*}{Word WORD} & \multirow[t]{2}{*}{Double word DWORD} \\
\hline HEX & DEC & & & & \\
\hline & & \%MB65541 & \[
\begin{aligned}
& \text { \%MX65541.0 } \\
& \ldots \\
& \% M X 65541.7
\end{aligned}
\] & & \\
\hline \multirow[t]{2}{*}{8003} & \multirow[t]{2}{*}{32771} & \%MB65542 & \[
\begin{aligned}
& \text { \%MX65542.0 } \\
& \text { \%MX65542.7 }
\end{aligned}
\] & \multirow[t]{2}{*}{\%MW32771} & \\
\hline & & \%MB65543 & \[
\begin{aligned}
& \text { \%MX65543.0 } \\
& \ldots \\
& \text { \%MX65543.7 }
\end{aligned}
\] & & \\
\hline \multicolumn{6}{|l|}{...} \\
\hline \multirow[t]{2}{*}{FFFE} & \multirow[t]{2}{*}{65534} & \%MB131068 & \[
\begin{aligned}
& \text { \%MX131068. } \\
& 0 \ldots \\
& \% M X 131068 . \\
& 7
\end{aligned}
\] & \multirow[t]{2}{*}{\%MW65534} & \multirow[t]{4}{*}{\%MD32767} \\
\hline & & \%MB131069 & \[
\begin{aligned}
& \text { \%MX131069. } \\
& 0 \ldots \\
& \% M X 131069 . \\
& 7
\end{aligned}
\] & & \\
\hline \multirow[t]{2}{*}{FFFF} & \multirow[t]{2}{*}{65535} & \%MB131070 & \[
\begin{aligned}
& \text { \%MX131070. } \\
& 0 \ldots \\
& \% M X 131070 . \\
& 7
\end{aligned}
\] & \multirow[t]{2}{*}{\%MW65535} & \\
\hline & & \%MB131071 & \[
\begin{aligned}
& \text { \%MX131071. } \\
& 0 \ldots \\
& \% M X 131071 . \\
& 7
\end{aligned}
\] & & \\
\hline
\end{tabular}

Table 818: Address assignment (bit accesses)
\begin{tabular}{|c|c|c|c|c|c|}
\hline \multicolumn{2}{|l|}{Modbus address} & \multirow[t]{2}{*}{Byte BYTE} & \multirow[t]{2}{*}{Bit (byte-oriented) BOOL} & \multirow[t]{2}{*}{Word WORD} & \multirow[t]{2}{*}{Double word DWORD} \\
\hline HEX & DEC & & & & \\
\hline 0000 & 0 & \multirow[t]{8}{*}{\%MB0} & \%MX0.0 & \multirow[t]{14}{*}{\%MW0} & \multirow[t]{14}{*}{\%MD0} \\
\hline 0001 & 1 & & \%MX0.1 & & \\
\hline 0002 & 2 & & \%MX0.2 & & \\
\hline 0003 & 3 & & \%MX0.3 & & \\
\hline 0004 & 4 & & \%MX0.4 & & \\
\hline 0005 & 5 & & \%MX0.5 & & \\
\hline 0006 & 6 & & \%MX0.6 & & \\
\hline 0007 & 7 & & \%MX0.7 & & \\
\hline 0008 & 8 & \multirow[t]{6}{*}{\%MB1} & \%MX1.0 & & \\
\hline 0009 & 9 & & \%MX1.1 & & \\
\hline 000A & 10 & & \%MX1.2 & & \\
\hline 000B & 11 & & \%MX1.3 & & \\
\hline 000C & 12 & & \%MX1.4 & & \\
\hline 000D & 13 & & \%MX1.5 & & \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|}
\hline \multicolumn{2}{|l|}{Modbus address} & \multirow[t]{2}{*}{\begin{tabular}{l}
Byte \\
BYTE
\end{tabular}} & \multirow[t]{2}{*}{Bit (byte-oriented) BOOL} & \multirow[t]{2}{*}{Word WORD} & \multirow[t]{2}{*}{Double word DWORD} \\
\hline HEX & DEC & & & & \\
\hline 000E & 14 & & \%MX1.6 & & \\
\hline 000F & 15 & & \%MX1.7 & & \\
\hline 0010 & 16 & \%MB2 & \%MX2.0 & \%MW1 & \\
\hline 0011 & 17 & & \%MX2.1 & & \\
\hline 0012 & 18 & & \%MX2.2 & & \\
\hline 0013 & 19 & & \%MX2.3 & & \\
\hline 0014 & 20 & & \%MX2.4 & & \\
\hline 0015 & 21 & & \%MX2.5 & & \\
\hline 0016 & 22 & & \%MX2.6 & & \\
\hline 0017 & 23 & & \%MX2.7 & & \\
\hline 0018 & 24 & \%MB3 & \%MX3.0 & & \\
\hline 0019 & 25 & & \%MX3.1 & & \\
\hline 001A & 26 & & \%MX3.2 & & \\
\hline 001B & 27 & & \%MX3.3 & & \\
\hline 001C & 28 & & \%MX3.4 & & \\
\hline 001D & 29 & & \%MX3.5 & & \\
\hline 001E & 30 & & \%MX3.6 & & \\
\hline 001F & 31 & & \%MX3.7 & & \\
\hline 0020 & 32 & \%MB4 & \%MX4.0 & \%MW2 & \%MD1 \\
\hline 0021 & 33 & & \%MX4.1 & & \\
\hline 0022 & 34 & & \%MX4.2 & & \\
\hline ... & ... & ... & ... & ... & \(\ldots\) \\
\hline OFFF & 4095 & \%MB511 & \%MX511.7 & \%MW255 & \%MD127 \\
\hline 1000 & 4096 & \%MB512 & \%MX512.0 & \%MW256 & \%MD128 \\
\hline ... & ... & \(\ldots\) & \(\ldots\) & ... & ... \\
\hline 7FFF & 32767 & \%MB4095 & \%MX4095.7 & \%MW2047 & \%MD1023 \\
\hline 8000 & 32768 & \%MB4096 & \%MX4096.0 & \%MW2048 & \%MD1024 \\
\hline \(\ldots\) & ... & \(\ldots\) & \(\ldots\) & ... & \(\cdots\) \\
\hline FFFF & 65535 & \%MB8191 & \%MX8191.7 & \%MW4095 & \%MD2047 \\
\hline
\end{tabular}

Calculation of the bit variable from the hexadecimal address:

\section*{Formula:}
\begin{tabular}{|l|l|l|l|}
\hline & \multicolumn{3}{|l|}{ Bit variable (BOOL) \(:=\%\) MXBYTE. BIT } \\
\hline & DEC & Decimal address & \\
\hline & where: & BYTE & DEC \(/ 8\) \\
\\
\hline & BIT & DEC mod 8 & (Modulo division) \\
\hline
\end{tabular}
```

Examples: - Address hexadecimal = 16\#2002
DEC := 8194
BYTE := 8194 / 8 := 1024
BIT := 8194 mod 8 := 2
Bit variable: %MX1024.2
- Address hexadecimal = 16\#3016
DEC := 12310
BYTE := 12310 / 8 := 1538,75 -> 1538
BIT := 12310 mod }8:=
Bit variable: %MX1538.6

- Address hexadecimal = 16\#55AA
DEC := 21930
BYTE := 21930 / 8 := 2741,25 -> 2741
BIT := 21930 mod }8:=
Bit variable: %MX2741.2

```

Calculation of the hexadecimal address from the bit variable:
```

Examples: - Bit variable := %MX515.4
DEC := 515 * 8 + 4 := 4124
Address hex := 16\#101C
- Bit variable := %MX3.3
DEC := 3 * 8 + 3 := 27
Address hex := 16\#001B
- Bit variable := %MX6666.2
DEC := 6666 * 8 + 2 := 53330
Address hex := 16\#D052

```

\section*{Peculiarities for accessing Modbus addresses}

Peculiarities for bit access:
- A WORD in the \%M area is assigned to each Modbus address 0000hex .. FFFFhex.
- Bit addresses 0000hex .. FFFFhex are contained in the word range \%MW0 .. \%MW4095

\section*{Areas protect from read/write access by Modbus client}

As described in 'Protocols and special servers' \(\Leftrightarrow\) Chapter 6.3.4.3.2 "Configuration of Modbus TCP/IP server" on page 1714, one write-protected and one read-protected area can be defined. If you try to write to a write-protected area or to read from a read-protected area, an exception response is generated.

\subsection*{6.8.2.9.5 Local data of the Modbus client}

The address of the area from which data are to be read or to which data are to be written is specified in the function block ModRtuMast at input "Data", via the ADR operator.

For the AC500, the following areas can be accessed using the ADR operator:
- Inputs area (\%I area)
- Outputs area (\%Q area)
- Area of non-buffered variables (VAR .. END_VAR or VAR_GLOBAL END_VAR)
- Addressable flag area (also protected areas for \%M area)
- Area of buffered variables (VAR RETAIN .. END_VAR or VAR_GLOBAL RETAIN .. END_VAR)

\subsection*{6.8.2.9.6 Modbus telegrams}

\section*{General}

The sending and receiving of telegrams shown in this section are not visible in the PLC.
However, the complete telegrams can be made visible using a serial data analyzer connected to the connection line between server and client, if required.

The amount of user data depends on the capabilities of the server and the client.
For the following examples, it is assumed that one AC500 Modbus module is used as client and another one is used as server. There may be different properties if modules of other manufacturers are used.

FCT 1 or 2: Read \(\mathbf{n}\) bits
Table 819: Client request
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Server address} & \multirow[t]{2}{*}{Function code} & \multicolumn{2}{|l|}{Server operand address} & \multicolumn{2}{|l|}{Number of bits} & \multicolumn{2}{|l|}{CRC} \\
\hline & & High & Low & High & Low & High & Low \\
\hline
\end{tabular}

Table 820: Server response
\begin{tabular}{|l|l|l|l|l|l|}
\hline \begin{tabular}{l} 
Server \\
address
\end{tabular} & \begin{tabular}{l} 
Function \\
code
\end{tabular} & Number of Bytes & \(\ldots\) Data... & CRC \\
\cline { 5 - 6 } & & & High & Low \\
\hline
\end{tabular}

\section*{FCT 3 or 4: Read \(n\) words}

Table 821: Client request
\begin{tabular}{|l|l|l|l|l|l|l|}
\hline \begin{tabular}{l} 
Server \\
address
\end{tabular} & \begin{tabular}{l} 
Function \\
code
\end{tabular} & \multicolumn{8}{|l|}{\begin{tabular}{l} 
Server operand \\
address
\end{tabular}} & \multicolumn{2}{l|}{ Number of words } & \multicolumn{2}{l|}{ CRC } \\
\cline { 3 - 8 } & & High & Low & High & Low & High \\
\hline
\end{tabular}

Table 822: Server response
\begin{tabular}{|l|l|l|l|l|l|l|}
\hline \begin{tabular}{l} 
Server \\
address
\end{tabular} & \begin{tabular}{l} 
Function \\
code
\end{tabular} & Number of Bytes & \multicolumn{2}{|l|}{ Data } & \multicolumn{1}{l|}{ CRC } \\
\cline { 4 - 6 } & & High & Low & High & Low \\
\hline
\end{tabular}

\section*{FCT 3 or 4: Read \(n\) double words}

The function code "read double word" is not defined in the Modbus RTU standard. This is why the double word is composed of a low word and a high word (depending on the manufacturer) ⓨ Chapter 6.8.2.9.6.3 "FCT 3 or 4: Read \(n\) words" on page 4513.

FCT 5: Write 1 bit
For the function code "write 1 bit", the value of the bit to be written is encoded in one word.
BIT = TRUE -> Data word = FF 00 HEX
BIT = FALSE -> Data word = 0000 HEX

Table 823: Client request
\begin{tabular}{|l|l|l|l|l|l|l|}
\hline \multirow{2}{*}{\begin{tabular}{l} 
Function \\
code
\end{tabular}} & \multicolumn{3}{|l|}{ Server operand address } & Number of words & CRC \\
\cline { 2 - 7 } & High & Low & High & Low & High & Low \\
\hline
\end{tabular}

Table 824: Server response
\begin{tabular}{|l|l|l|l|l|l|l|}
\hline \begin{tabular}{l} 
Function \\
code
\end{tabular} & \multicolumn{8}{|l|}{ Server operand address } & Data & CRC \\
\cline { 2 - 7 } & High & Low & High & Low & High & Low \\
\hline
\end{tabular}

FCT 6: Write 1 word
Table 825: Server request
\begin{tabular}{|l|l|l|l|l|l|l|}
\hline \begin{tabular}{l} 
Server \\
address
\end{tabular} & \begin{tabular}{l} 
Function \\
code
\end{tabular} & \multicolumn{3}{|l|}{\begin{tabular}{l} 
Server operand \\
address
\end{tabular}} & Data & CRC \\
\cline { 3 - 7 } & & High & Low & High & Low & High \\
\hline
\end{tabular}

Table 826: Server response
\begin{tabular}{|l|l|l|l|l|l|l|}
\hline \begin{tabular}{l} 
Server \\
address
\end{tabular} & \begin{tabular}{l} 
Function \\
code
\end{tabular} & \begin{tabular}{l} 
Server operand \\
address
\end{tabular} & \multicolumn{2}{|l|}{ Data } & CRC \\
\cline { 3 - 8 } & High & Low & High & Low & High & Low \\
\hline
\end{tabular}

FCT 15: Write n bits
Table 827: Client request
\begin{tabular}{|l|l|l|l|l|l|l|}
\hline \begin{tabular}{l} 
Server operand \\
address
\end{tabular} & \multicolumn{3}{|l|}{ Number of bits } & \begin{tabular}{l} 
Number of \\
bytes
\end{tabular} & \(\ldots\) Data... & CRC \\
\hline High & Low & High & Low & & & High \\
\hline
\end{tabular}

Table 828: Server response
\begin{tabular}{|l|l|l|l|l|l|l|}
\hline \begin{tabular}{l} 
Server \\
address
\end{tabular} & \begin{tabular}{l} 
Function \\
code
\end{tabular} & \multicolumn{2}{|l|}{\begin{tabular}{l} 
Server operand \\
address
\end{tabular}} & \multicolumn{2}{|l|}{ Number of bits } & \multicolumn{2}{l|}{ CRC } \\
\cline { 2 - 7 } & & High & Low & High & Low & High \\
\hline
\end{tabular}

FCT 16: Write n words
Table 829: Client request
\begin{tabular}{|l|l|l|l|l|l|l|}
\hline \begin{tabular}{l} 
Server operand \\
address
\end{tabular} & \multicolumn{2}{|l|}{ Number of words } & \begin{tabular}{l} 
Number of \\
bytes
\end{tabular} & \(\ldots\). Data... & \multicolumn{2}{|l|}{ CRC } \\
& & & & & \\
\hline High & Low & High & Low & & Low \\
\hline
\end{tabular}

Table 830: Server response
\begin{tabular}{|l|l|l|l|l|l|l|}
\hline \multirow{2}{*}{\begin{tabular}{l} 
Function \\
code
\end{tabular}} & \multicolumn{3}{|l|}{ Server operand address } & \multicolumn{2}{|l|}{ Number of words } & CRC \\
\cline { 2 - 7 } & High & Low & High & Low & High & Low \\
\hline
\end{tabular}

FCT 16: Write \(n\) double words
The function code "write double word" is not defined in the Modbus RTU standard. This is why the double word is composed of a low word and a high word (depending on the manufacturer).

Table 831: Client request
\begin{tabular}{|l|l|l|l|l|l|l|}
\hline \begin{tabular}{l} 
Server operand \\
address
\end{tabular} & \multicolumn{3}{|l|}{ Number of words } & \begin{tabular}{l} 
Number of \\
bytes
\end{tabular} & \(\ldots\) Data... & \multicolumn{1}{l|}{ CRC } \\
\hline High & Low & High & Low & & & High \\
\hline
\end{tabular}

Table 832: Server response
\begin{tabular}{|l|l|l|l|l|l|l|}
\hline \begin{tabular}{l} 
Server \\
address
\end{tabular} & \begin{tabular}{l} 
Function \\
code
\end{tabular} & \begin{tabular}{l} 
Server operand \\
address
\end{tabular} & \multicolumn{2}{|l|}{ Number of words } & \multicolumn{1}{l|}{ CRC } \\
\cline { 3 - 7 } & & High & Low & High & Low & High \\
Low \\
\hline
\end{tabular}

\section*{FCT 22: Mask write register}

Table 833: Client request
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Server address} & \multirow[t]{2}{*}{Function code} & \multicolumn{2}{|l|}{Server operand address} & \multicolumn{2}{|l|}{AND Mask} & \multicolumn{2}{|l|}{OR Mask} & \multicolumn{2}{|l|}{CRC} \\
\hline & & High & Low & High & Low & High & Low & High & Low \\
\hline
\end{tabular}

Table 834: Server response
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Server address} & \multirow[t]{2}{*}{Function code} & \multicolumn{2}{|l|}{Server operand address} & \multicolumn{2}{|l|}{AND Mask} & \multicolumn{2}{|l|}{OR Mask} & \multicolumn{2}{|l|}{CRC} \\
\hline & & High & Low & High & Low & High & Low & High & Low \\
\hline
\end{tabular}

FCT 23: Read/Write n words
Table 835: Client request
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Server addre ss} & \multirow[t]{2}{*}{Function code} & \multicolumn{2}{|l|}{Operand addr. read} & \multicolumn{2}{|l|}{Number of words read} & \multicolumn{2}{|l|}{Operand addr. write} & \multicolumn{2}{|l|}{Number of words write} & \multirow[t]{2}{*}{Numb er of bytes write} & \multirow[t]{2}{*}{\[
\begin{aligned}
& \text {...Dat } \\
& \text { a.... }
\end{aligned}
\]} & \multicolumn{2}{|l|}{CRC} \\
\hline & & High & \[
\begin{aligned}
& \text { Lo } \\
& \text { w }
\end{aligned}
\] & High & \[
\begin{aligned}
& \text { Lo } \\
& \text { w }
\end{aligned}
\] & High & Low & High & Low & & & Hig
h & Lo \\
\hline
\end{tabular}

Table 836: Server response
\begin{tabular}{|l|l|l|l|l|l|}
\hline \begin{tabular}{l} 
Server \\
address
\end{tabular} & \begin{tabular}{l} 
Function \\
code
\end{tabular} & \begin{tabular}{l} 
Number of bytes \\
read
\end{tabular} & \(\ldots\) Data... & \multicolumn{2}{|l|}{ CRC } \\
\cline { 4 - 6 } & & High & Low \\
\hline
\end{tabular}

\section*{Exception response by server}

In operating mode Modbus client, the AC500 does only send requests, if the parameters at the ModRtuMast inputs are logically correct.
Nevertheless, it can happen that a server cannot process the request of the client or that the server cannot interpret the request due to transmission errors or in case it's capabilities are exceeded in any way. In those cases, the server returns an exception response to the client. In order to identify this response as an exception response, the function code returned by the server is a logical OR interconnection of the function code received from the client and the value 80HEX.

Table 837: Server response
\begin{tabular}{|l|l|l|l|l|}
\hline Server address & OR 80HEX & Error code & \multicolumn{2}{|l|}{ CRC } \\
\cline { 4 - 5 } & & & High & Low \\
\hline
\end{tabular}

Possible error codes of the client
\begin{tabular}{|l|l|}
\hline Code & Description \\
\hline 01DEC & \begin{tabular}{l} 
ILLEGAL FUNCTION \\
The server does not support the function requested by the client
\end{tabular} \\
\hline 02DEC & \begin{tabular}{l} 
ILLEGAL DATA ADDRESS \\
Invalid operand address in the server or operand area exceeded
\end{tabular} \\
\hline 03DEC & \begin{tabular}{l} 
ILLEGAL DATA VALUE \\
At least one value is outside the permitted range of values
\end{tabular} \\
\hline 04DEC & \begin{tabular}{l} 
SERVER DEVICE FAILURE \\
An unrecoverable error occurred while the server was attempting to \\
perform the requested action
\end{tabular} \\
\hline 05DEC & \begin{tabular}{l} 
ACKNOWLEDGE \\
Specialized use in conjunction with programming commands. \\
The server has accepted the request and is processing it, but a \\
long duration of time will be required to do so. This response is \\
returned to prevent a timeout error from occurring in the client. \\
The client can next issue a Pooll Program Complete message to \\
determine if processing is completed
\end{tabular} \\
\hline 06DEC & \begin{tabular}{l} 
SERVER DEVICE BUSY \\
Specialized use in conjunction with programming commands.
\end{tabular} \\
\hline 07DEC & \begin{tabular}{l} 
The server is engaged in processing a long-duration program com- \\
mand. The client should retransmit the message later when the \\
server is free.
\end{tabular} \\
\hline \begin{tabular}{l} 
NEGATIVE ACKNOWLEDGE \\
Specialized use in conjunction with programming commands.
\end{tabular} \\
\hline \begin{tabular}{l} 
The server cannot perform the programming functions. Client \\
should request diagnostic or error information from server.
\end{tabular} \\
\hline \begin{tabular}{l} 
MEMORY PARITY ERROR \\
Specialized use in conjunction with function codes 20 and 21 and \\
reference type 6, to indicate that the extended file area failed to \\
pass a consistency check. The server attempted to read record file, \\
but detected a parity error in the memory. The client can retry the \\
request, but service may be required on the server device.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline Code & Description \\
\hline 10DEC & \begin{tabular}{l} 
GATEWAY PATH UNAVAILABLE \\
Specialized use in conjunction with gateways, indicates that the \\
gateway was unable to allocate an internal communication path \\
from the input port to the output port for processing the request. \\
Usually means that the gateway is misconfigured or overloaded.
\end{tabular} \\
\hline 11DEC & \begin{tabular}{l} 
GATEWAY TARGET DEVICE FAILED TO RESPOND \\
Specialized use in conjunction with gateways, indicates that no \\
response was obtained from the target device. Usually means that \\
the device is not present on the network.
\end{tabular} \\
\hline
\end{tabular}
Example \begin{tabular}{l} 
Table 838: Modbus request of the client \\
\begin{tabular}{|l|l|l|}
\hline Function code: & 01 & Read n bits \\
\hline Server operand address: & \(4000 \mathrm{HEX}=16384 \mathrm{DEC}\) & \begin{tabular}{l} 
Area for read access disa- \\
bled in server
\end{tabular} \\
\hline
\end{tabular} \\
\begin{tabular}{|l|l|}
\hline Table 839: Modbus response of the server & \\
\hline Function code: & 81 HEX \\
\hline & 03 \\
\hline
\end{tabular}
\end{tabular}\(.\)\begin{tabular}{l} 
Error code:
\end{tabular}

\subsection*{6.8.2.9.7 Processing bits}

\section*{General}

Some of the Modbus function codes are used to read or write bits (coils, discrete inputs). While a variable of data type WORD can be accessed easily, accessing a stream of bits is complex.
Data type 'BIT' \({ }^{\wedge} \gg\) Chapter 6.4.1.20.5.11 "Data Type 'BIT"' on page 2243 must not be mixed up with data type 'BOOL' \(\Longleftrightarrow\) Chapter 6.4.1.20.5.2 "Data type 'BOOL'" on page 2235. Variables of both types may have values 'TRUE' or 'FALSE'. But while BIT means one single bit only, BOOL requires a byte (8 bit) of memory.

\section*{Modbus client}

When accessing bits in a Server, the local data referred to at Client function blocks input data is always expected to be of format BOOL.
```

VAR
abBoolArray : ARRAY [0..15] OF BOOL;
ModMast : ModTcpMast;
END_VAR
abBoolArray[0] := TRUE;
abBoolArray[15] := FALSE;
ModMast (Execute:= ModMastExecute,
Eth:= ETH1,
IPAdr:= IP_ADR_STRING_IO_DWORD('192.168.0.100'),
UnitID:= 0,
Fct:= 15,
Addr:= 16\#0000,
Nb:= SIZEOF (abBoolArray),
Data:= ADR(abBoolArray));

```

\section*{Modbus server}

\section*{Using the bit offset}

The simplest way to access a certain bit within a larger variable is to directly use the bit offset ( 0 based; \(\Leftrightarrow\) Chapter 6.4.1.20.4.11 "Bit Access in Variables" on page 2229).
```

VAR
awWordVariable AT %%NWO : ARRAY [0..7] OF WORD;
END VAR
awWordVariable[0].0 := TRUE;
awWordVariable[7].15 := FALSE;

```

\section*{Defining symbolic names for the bit offsets}

A more convenient way to access bits e.g. within a word is to define a symbolic name for each single offset « Chapter 6.4.1.20.4.11 "Bit Access in Variables" on page 2229.
```

VAR GLOBAL CONSTANT
(*bit offsets in avWordVariable[0]*)
NameBitOffset_0_00 : INT := 0;
NameBitOffset_0_01 : INT := 1;
(*...*)
NameBitOffset_0_15 : INT := 15;
(*bit offsets in av|ordVariable[7]*)
NameBitOffset_7_00 : INT := 0;
NameBitOffset_7_01 : INT := 1;
(*...*)
NameBitOffset_7_15 : INT := 15;
END VAR

```
VAR
awWordVariable AT sMWO : ARRAY [0..7] OF WORD;
END VAR
awWordVariable[0].NameBitOffset_0_00 := TRUE;
awWordVariable[7].NameBitOffset_7_15 := FALSE;

\section*{Defining a data type}

A further alternative is to define your own data types \& Chapter 6.4.1.21.2.9 "Object 'DUT"' on page 2461 according to the requirements of your particular application * "Symbolic bit access in structure variables" on page 2230.
```

TYPE ControlWordType :
STRUCT
NameBitOffset_00 : BIT;
NameBitOffset_01 : BIT;
NameBitOffset_02 : BIT;
NameBitOffset_03 : BIT;
NameBitOffset_04 : BIT;
NameBitOffset_05 : BIT;
NameBitOffset_06 : BIT;
NameBitOffset_07 : BIT;
NameBitOffset_08 : BIT;
NameBitOffset_09 : BIT;
NameBitOffset_10 : BIT;
NameBitOffset_11 : BIT;
NameBitOffset_12 : BIT;
NameBitOffset_13 : BIT;
NameBitOffset_14 : BIT;
NameBitOffset_15 : BIT;
END_STRUCT
END_TYPE

```

\section*{VAR}
atTypeVariable AT sMw : ARRAY [0..7] OF ControlWordType;
END VAR
atTypeVariable [0].NameBitOffset_00 := TRUE;
atTypeVariable[7].NameBitOffset_15 := FALSE;

\section*{Defining a complex data type}

In case your application requires some more complex data types you can combine data types (DUT; \({ }^{4}\) "Symbolic bit access in structure variables" on page 2230).
```

TYPE ControlWordOType :
STRUCT
NameBitOffset_0_00 : BIT;
NameBitOffset_0_01 : BIT;
NameBitOffset_0_02 : BIT;
NameBitOffset_0_03 : BIT;
NameBitOffset_0_04 : BIT;
NameBitOffset_0_05 : BIT;
NameBitOffset_0_06 : BIT;
NameBitOffset_0_07 : BIT;
NameBitOffset_0_08 : BIT;
NameBitOffset_0_09 : BIT;
NameBitOffset_0_10 : BIT;
NameBitOffset_0_11 : BIT;
NameBitOffset_0_12 : BIT;
NameBitOffset_0_13 : BIT;
NameBitOffset_0_14 : BIT;
NameBitOffset_0_15 : BIT;
END_STRUCT
END_TYPE

```

\section*{ControlWord1 Type}
            TYPE ControlWord1Type :
            STRUCT
                NameBitOffset_1_00 : BIT;
                NameBitOffset_1_01 : BIT;
                NameBitOffset_1_02 : BIT;
                    NameBitOffset_1_03 : BIT;
                    NameBitOffset_1_04 : BIT;
                        NameBitOffset_1_05 : BIT;
                        NameBitOffset_1_06 : BIT;
    NameBitOffset_1_07 : BIT;
    NameBitOffset_1_08 : BIT;
    NameBitOffset_1_09 : BIT;
    NameBitOffset_1_10 : BIT;
    NameBitOffset_1_11 : BIT;
    NameBitOffset_1_12 : BIT;
    NameBitOffset_1_13 : BIT;
    NameBitOffset_1_14 : BIT;
    NameBitOffset_1_15 : BIT;
    END_STRUCT
    END_TYPE
                    TYPE ControlWordListType :
                    STRUCT
                            ControlWordo : ControlWord0Type;
                            ControlWord1 : ControlWord1Type;
                    END_STRUCT
                            END TYPE
VAR
atWordListType AT \(\$\) MWO : ControlWordListType;
END_VAR
atWordListType.ControlWord0.NameBitOffset_0_00 := TRUE;
atWordListType.ControlWord1.NameBitOffset_1_15 := FALSE;

\section*{Pack／unpack BOOL variables}

In case you prefer variables of type BOOL you can use the functions for packing MEM＿Pack＿BitsToByte and unpacking MEM＿UnpackWord of the CAA＿Memory．library，which can be found with the Library Manager \({ }^{\circ} \Longleftrightarrow\) Chapter 6．5．3＂Library Manager functionality＂ on page 3773.

VAR
abBoolVariable ：ARRAY［0．．15］OF BOOL；
wWordVariable0 ：AT s⿳⺈⿴囗十一日⿹勹巳 WORD； wWordVariable1 ：AT sMW1 WORD；
END＿VAR
VAR
Unpack ：Mem．UnpackWord；
END＿VAR
abBoolVariable［0］：＝TRUE；
abBoolVariable［15］：＝FALSE；
wWordVariable0 ：＝Mem．PackBitsToWord（abBoolVariable［0］，abBoolVariable［1］，abBoolVariable［2］，abBoolVariable［3］， abBoolVariable［4］，abBoolVariable［5］，abBoolVariable［6］，abBoolVariable［7］， abBoolVariable［8］，abBoolVariable［9］，abBoolVariable［10］，abBoolVariable［11］， abBoolVariable［12］，abBoolVariable［13］，abBoolVariable［14］，abBoolVariable［15］）；
wWordVariable1 ：＝1；
Unpack（wValue：＝wWordVariable1，
\(x B i t 0 \Rightarrow\) abBoolVariable［0］，xBit1 \(\Rightarrow\) abBoolVariable［1］，xBit2 \(\Rightarrow>\) abBoolVariable［2］，xBit3 \(\Rightarrow>\) abBoolVariable［3］， \(x\) Bit4 \(\Rightarrow\) abBoolVariable［4］，xBit5 \(\Rightarrow\) abBoolVariable［5］，xBit6 \(\Rightarrow\) abBoolVariable［6］，xBit7 \(\Rightarrow\) abBoolVariable［7］， \(x\) xit8 \(\Rightarrow\) abBoolVariable［8］，xBit9 \(\Rightarrow\) abBoolVariable［9］，xBit10 \(\Rightarrow>\) abBoolVariable［10］，xBit11 \(\Rightarrow>\) abBoolVariable［11］， xBit12 \(\Rightarrow\) abBoolVariable［12］，xBit13 \(\Rightarrow\) abBoolVariable［13］，xBit14＝＞abBoolVariable［14］，xBit15＝＞abBoolVariable［15］）；

\section*{6．8．2．9．8 Function block ModRtuMast}

This function block is only required in the operating mode Modbus client．It handles the commu－ nication（transmission of telegrams to the servers and receipt of telegrams from the servers）． The function block can be used for the local serial interfaces of the controller．A separate instance of the function block has to be used for each interface．

ModRtuMast is contained in the library AC500＿ModRtuMast．

\subsection*{6.8.2.10 Communication with Modbus TCP/IP}

\subsection*{6.8.2.10.1 Protocol description}

The Modbus TCP protocol is implemented in the AC500 processor modules. Modbus is a master-slave (client-server) protocol. The client sends a request to the server and receives the responses. Each Ethernet interface can work as Modbus client and server interface in parallel if required.

The Modbus operating mode of an Ethernet interface is set in Modbus on TCP/IP protocol \(\boldsymbol{y}^{2}\) Chapter 6.3.4.3.1 "Modbus on TCP/IP protocol" on page 1713.

Modbus client In this operating mode, the telegram traffic with the server(s) is handled via the function block ModTcpMast, which can be found through the Library Manager \({ }^{\wedge}\), Chapter 6.5.3 "Library Manager functionality" on page 3773. This function block sends Modbus request telegrams to the server(s) via the set interface and receives Modbus response telegrams from the server(s) via this interface.

The Modbus function blocks transferred by the client contain the following information:
- Transaction identifier for synchronization between messages of server and client (2 byte)
- Protocol identifier (0 for Modbus/TCP) (2 byte)
- Length field (Number of bytes in frame) (2 byte)
- Unit identifier (1 byte)
- Function code that defines the request of the client (1 byte)
- Data to be exchanged ( n bytes)

Modbus server In this operating mode, no function block is required for Modbus communication. Sending and receiving Modbus telegrams is performed automatically.
The AC500 CPUs process the following Modbus operation codes:
\begin{tabular}{|l|l|l|}
\hline \multicolumn{2}{|l|}{ Function code } & HEX \\
\cline { 1 - 2 } DEC & 01 or 02 & Description \\
\hline 01 or 02 & 03 or 04 & Read n bits \\
\hline 03 or 04 & 05 & \begin{tabular}{l} 
Read n words \\
Write one bit (encoded in one \\
word \()\)
\end{tabular} \\
\hline 05 & 06 & Write one word \\
\hline 06 & \(0 F\) & \begin{tabular}{l} 
Write n bits (encoded in one \\
byte)
\end{tabular} \\
\hline 15 & 10 & Write n words \\
\hline 16 & 16 & Mask write \\
\hline 22 & 17 & \begin{tabular}{l} 
Read/write multiple words in \\
one telegram
\end{tabular} \\
\hline 23 & & \\
\hline
\end{tabular}

The following restrictions apply to the length of the data to be sent:
\begin{tabular}{|l|l|l|}
\hline Function code & HEX & \multirow{2}{*}{ Max. length } \\
\hline DEC & 01 or 02 & 2000 bits \\
\hline 01 or 02 & 03 or 04 & 125 words / 62 double words \\
\hline 03 or 04 & &
\end{tabular}
\begin{tabular}{|l|l|l|}
\hline Function code & HEX & Max. length \\
\hline DEC & 05 & 1 bit \\
\hline 05 & 06 & 1 word \\
\hline 06 & 0 F & 2000 bits \\
\hline 15 & 10 & 123 words \(/ 61\) double words \\
\hline 16 & 16 & Write: 1 word \\
\hline 22 & 17 & \begin{tabular}{l} 
Read: 125 words \(/ 62\) double \\
words \\
Write: 121 words \(/ 60\) double \\
words
\end{tabular} \\
\hline 23 & & \\
\hline
\end{tabular}

\subsection*{6.8.2.10.2 Technical data}

Configuration of Modbus on TCP/IP is described in the chapter 'Protocols and special servers' ² Chapter 6.3.4.3 "Modbus protocol" on page 1713.

\subsection*{6.8.2.10.3 Modbus addresses for AC500-eCo V3 processor modules PM50xx}

Modbus A range of maximum 64 kB is allowed for the access via Modbus to the addressable flag area address table (\%M area). Thus, the complete address range 0000hex up to 7FFFhex is available for Modbus.

The availability of the segments depends on the CPU. The size of the \%M area can be found in the technical data of the CPUs and in the target system settings.
Inputs and outputs cannot be directly accessed using Modbus.
Following values apply:
\begin{tabular}{|c|c|c|c|c|c|}
\hline & PM5012-x-ETH & PM5032-x-ETH & PM5052-x-ETH & \[
\begin{array}{|l|}
\hline \text { PM5072- } \\
\text { T-2ETH(W) }
\end{array}
\] & PM5082-T-2ETH \\
\hline Size of the \%M area & 4 kB & 16 kB & 16 kB & 64 kB & 64 kB \\
\hline \multicolumn{6}{|l|}{Modbus address range (Word accesses)} \\
\hline HEX & 0000 ... 07FF & 0000 ... 1FFF & 0000 ... 1FFF & 0000 ... 7FFF & 0000 ... 7FFF \\
\hline DEC & 0000 ... 2047 & 0000 ... 8191 & 0000 ... 8191 & 0000 ... 32767 & \(0000 \ldots 32767\) \\
\hline Byte & \[
\begin{aligned}
& \hline \% M B 0 \ldots \\
& \% M B 4097
\end{aligned}
\] & \%MB0 ... \%MB16382 & \%MB0 ... \%MB16382 & \[
\begin{aligned}
& \text { \%MB0 ... } \\
& \% M B 6534
\end{aligned}
\] & \[
\begin{aligned}
& \% M B 0 \ldots \\
& \% M B 65534
\end{aligned}
\] \\
\hline Word & \%MWO .. \%MW2047 & \%MWO ... \%MW8191 & \%MWO .. \%MW8191 & \%MW0 . \%MW32767 & \%MW0 .. \%MW32767 \\
\hline
\end{tabular}
6.8.2.10.4 Modbus addresses for AC500 V3 processor modules PM56xx

\section*{Modbus address table}

Table 840: Modbus addresses (word accesses)
\begin{tabular}{|c|c|c|c|c|c|}
\hline \multicolumn{2}{|l|}{Modbus address} & \multirow[t]{2}{*}{Byte BYTE} & \multirow[t]{2}{*}{Bit (byte-oriented) BOOL} & \multirow[t]{2}{*}{Word WORD} & \multirow[t]{2}{*}{Double word DWORD} \\
\hline HEX & DEC & & & & \\
\hline \multirow[t]{2}{*}{0000} & \multirow[t]{2}{*}{0} & \%MB0 & \[
\begin{aligned}
& \text { \%MX0.0 ... } \\
& \text { \%MX0.7 }
\end{aligned}
\] & \multirow[t]{2}{*}{\%MW0} & \multirow[t]{4}{*}{\%MD0} \\
\hline & & \%MB1 & \[
\begin{aligned}
& \text { \%MX1.0 ... } \\
& \text { \%MX1.7 }
\end{aligned}
\] & & \\
\hline \multirow[t]{2}{*}{0001} & \multirow[t]{2}{*}{1} & \%MB2 & \[
\begin{aligned}
& \text { \%MX2.0 ... } \\
& \text { \%MX2.7 }
\end{aligned}
\] & \multirow[t]{2}{*}{\%MW1} & \\
\hline & & \%MB3 & \[
\begin{aligned}
& \text { \%MX3.0 ... } \\
& \text { \%MX3.7 }
\end{aligned}
\] & & \\
\hline \multirow[t]{2}{*}{0002} & \multirow[t]{2}{*}{2} & \%MB4 & \[
\begin{aligned}
& \text { \%MX4.0 ... } \\
& \text { \%MX4.7 }
\end{aligned}
\] & \multirow[t]{2}{*}{\%MW2} & \multirow[t]{4}{*}{\%MD1} \\
\hline & & \%MB5 & \[
\begin{aligned}
& \text { \%MX5.0 ... } \\
& \text { \%MX5.7 }
\end{aligned}
\] & & \\
\hline \multirow[t]{2}{*}{0003} & \multirow[t]{2}{*}{3} & \%MB6 & \[
\begin{aligned}
& \text { \%MX6.0 ... } \\
& \text { \%MX6.7 }
\end{aligned}
\] & \multirow[t]{2}{*}{\%MW3} & \\
\hline & & \%MB7 & \[
\begin{aligned}
& \text { \%MX7.0 ... } \\
& \text { \%MX7.7 }
\end{aligned}
\] & & \\
\hline \multicolumn{6}{|l|}{\(\ldots\)} \\
\hline \multirow[t]{2}{*}{7FFE} & \multirow[t]{2}{*}{32766} & \%MB65532 & \[
\begin{aligned}
& \text { \%MX65532.0 } \\
& \ldots \\
& \% M X 65532.7
\end{aligned}
\] & \multirow[t]{2}{*}{\%MW32766} & \multirow[t]{4}{*}{\%MD16383} \\
\hline & & \%MB65533 & \[
\begin{aligned}
& \text { \%MX65533.0 } \\
& \text { \% MX65533.7 }
\end{aligned}
\] & & \\
\hline \multirow[t]{2}{*}{7FFF} & \multirow[t]{2}{*}{32767} & \%MB65534 & \[
\begin{aligned}
& \text { \%MX65534.0 } \\
& \text { \% } \\
& \text { \%MX65534.7 }
\end{aligned}
\] & \multirow[t]{2}{*}{\%MW32767} & \\
\hline & & \%MB65535 & \[
\begin{aligned}
& \text { \%MX65535.0 } \\
& \text { \% } \\
& \text { \%MX65535.7 }
\end{aligned}
\] & & \\
\hline \multirow[t]{2}{*}{8000} & \multirow[t]{2}{*}{32768} & \%MB65536 & \[
\begin{aligned}
& \text { \%MX65536.0 } \\
& \ldots \\
& \text { \%MX65536.7 }
\end{aligned}
\] & \multirow[t]{2}{*}{\%MW32768} & \multirow[t]{4}{*}{\%MD16384} \\
\hline & & \%MB65537 & \[
\begin{aligned}
& \text { \%MX65537.0 } \\
& \text { \% } \\
& \text { \%X65537.7 }
\end{aligned}
\] & & \\
\hline \multirow[t]{2}{*}{8001} & \multirow[t]{2}{*}{32769} & \%MB65538 & \[
\begin{aligned}
& \text { \%MX65538.0 } \\
& \text { \% } \\
& \text { \%MX65538.7 }
\end{aligned}
\] & \multirow[t]{2}{*}{\%MW32769} & \\
\hline & & \%MB65539 & \[
\begin{aligned}
& \text { \%MX65539.0 } \\
& \text { \% } \\
& \text { \%MX65539.7 }
\end{aligned}
\] & & \\
\hline 8002 & 32770 & \%MB65540 & \[
\begin{aligned}
& \text { \%MX65540.0 } \\
& \text { \% } \\
& \text { \%MX65540.7 }
\end{aligned}
\] & \%MW32770 & \%MD16385 \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|}
\hline \multicolumn{2}{|l|}{Modbus address} & \multirow[t]{2}{*}{\begin{tabular}{l}
Byte \\
BYTE
\end{tabular}} & \multirow[t]{2}{*}{Bit (byte-oriented)
BOOL} & \multirow[t]{2}{*}{Word WORD} & \multirow[t]{2}{*}{Double word DWORD} \\
\hline HEX & DEC & & & & \\
\hline & & \%MB65541 & \[
\begin{aligned}
& \text { \%MX65541.0 } \\
& \ldots \\
& \% M X 65541.7
\end{aligned}
\] & & \\
\hline \multirow[t]{2}{*}{8003} & \multirow[t]{2}{*}{32771} & \%MB65542 & \[
\begin{aligned}
& \text { \%MX65542.0 } \\
& \text { \%MX65542.7 }
\end{aligned}
\] & \multirow[t]{2}{*}{\%MW32771} & \\
\hline & & \%MB65543 & \[
\begin{aligned}
& \text { \%MX65543.0 } \\
& \ldots \\
& \text { \%MX65543.7 }
\end{aligned}
\] & & \\
\hline \multicolumn{6}{|l|}{...} \\
\hline \multirow[t]{2}{*}{FFFE} & \multirow[t]{2}{*}{65534} & \%MB131068 & \[
\begin{aligned}
& \text { \%MX131068. } \\
& 0 \ldots \\
& \% M X 131068 . \\
& 7
\end{aligned}
\] & \multirow[t]{2}{*}{\%MW65534} & \multirow[t]{4}{*}{\%MD32767} \\
\hline & & \%MB131069 & \[
\begin{aligned}
& \text { \%MX131069. } \\
& 0 \ldots \\
& \% M X 131069 . \\
& 7
\end{aligned}
\] & & \\
\hline \multirow[t]{2}{*}{FFFF} & \multirow[t]{2}{*}{65535} & \%MB131070 & \[
\begin{aligned}
& \text { \%MX131070. } \\
& 0 \ldots \\
& \% M X 131070 . \\
& 7
\end{aligned}
\] & \multirow[t]{2}{*}{\%MW65535} & \\
\hline & & \%MB131071 & \[
\begin{aligned}
& \text { \%MX131071. } \\
& 0 \ldots \\
& \% M X 131071 . \\
& 7
\end{aligned}
\] & & \\
\hline
\end{tabular}

Table 841: Address assignment (bit accesses)
\begin{tabular}{|c|c|c|c|c|c|}
\hline \multicolumn{2}{|l|}{Modbus address} & \multirow[t]{2}{*}{Byte BYTE} & \multirow[t]{2}{*}{Bit (byte-oriented)
BOOL} & \multirow[t]{2}{*}{Word WORD} & \multirow[t]{2}{*}{Double word DWORD} \\
\hline HEX & DEC & & & & \\
\hline 0000 & 0 & \multirow[t]{8}{*}{\%MB0} & \%MX0.0 & \multirow[t]{14}{*}{\%MW0} & \multirow[t]{14}{*}{\%MD0} \\
\hline 0001 & 1 & & \%MX0.1 & & \\
\hline 0002 & 2 & & \%MX0.2 & & \\
\hline 0003 & 3 & & \%MX0.3 & & \\
\hline 0004 & 4 & & \%MX0.4 & & \\
\hline 0005 & 5 & & \%MX0.5 & & \\
\hline 0006 & 6 & & \%MX0.6 & & \\
\hline 0007 & 7 & & \%MX0.7 & & \\
\hline 0008 & 8 & \multirow[t]{6}{*}{\%MB1} & \%MX1.0 & & \\
\hline 0009 & 9 & & \%MX1.1 & & \\
\hline 000A & 10 & & \%MX1.2 & & \\
\hline 000B & 11 & & \%MX1.3 & & \\
\hline 000C & 12 & & \%MX1.4 & & \\
\hline 000D & 13 & & \%MX1.5 & & \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|}
\hline \multicolumn{2}{|l|}{Modbus address} & \multirow[t]{2}{*}{\begin{tabular}{l}
Byte \\
BYTE
\end{tabular}} & \multirow[t]{2}{*}{Bit (byte-oriented) BOOL} & \multirow[t]{2}{*}{Word WORD} & \multirow[t]{2}{*}{Double word DWORD} \\
\hline HEX & DEC & & & & \\
\hline 000E & 14 & & \%MX1.6 & & \\
\hline 000F & 15 & & \%MX1.7 & & \\
\hline 0010 & 16 & \%MB2 & \%MX2.0 & \%MW1 & \\
\hline 0011 & 17 & & \%MX2.1 & & \\
\hline 0012 & 18 & & \%MX2.2 & & \\
\hline 0013 & 19 & & \%MX2.3 & & \\
\hline 0014 & 20 & & \%MX2.4 & & \\
\hline 0015 & 21 & & \%MX2.5 & & \\
\hline 0016 & 22 & & \%MX2.6 & & \\
\hline 0017 & 23 & & \%MX2.7 & & \\
\hline 0018 & 24 & \%MB3 & \%MX3.0 & & \\
\hline 0019 & 25 & & \%MX3.1 & & \\
\hline 001A & 26 & & \%MX3.2 & & \\
\hline 001B & 27 & & \%MX3.3 & & \\
\hline 001C & 28 & & \%MX3.4 & & \\
\hline 001D & 29 & & \%MX3.5 & & \\
\hline 001E & 30 & & \%MX3.6 & & \\
\hline 001F & 31 & & \%MX3.7 & & \\
\hline 0020 & 32 & \%MB4 & \%MX4.0 & \%MW2 & \%MD1 \\
\hline 0021 & 33 & & \%MX4.1 & & \\
\hline 0022 & 34 & & \%MX4.2 & & \\
\hline ... & ... & ... & ... & ... & \(\ldots\) \\
\hline OFFF & 4095 & \%MB511 & \%MX511.7 & \%MW255 & \%MD127 \\
\hline 1000 & 4096 & \%MB512 & \%MX512.0 & \%MW256 & \%MD128 \\
\hline ... & ... & \(\ldots\) & \(\ldots\) & ... & ... \\
\hline 7FFF & 32767 & \%MB4095 & \%MX4095.7 & \%MW2047 & \%MD1023 \\
\hline 8000 & 32768 & \%MB4096 & \%MX4096.0 & \%MW2048 & \%MD1024 \\
\hline \(\ldots\) & ... & \(\ldots\) & \(\ldots\) & ... & \(\cdots\) \\
\hline FFFF & 65535 & \%MB8191 & \%MX8191.7 & \%MW4095 & \%MD2047 \\
\hline
\end{tabular}

Calculation of the bit variable from the hexadecimal address:

\section*{Formula:}
\begin{tabular}{|l|l|l|l|}
\hline & \multicolumn{3}{|l|}{ Bit variable (BOOL) \(:=\%\) MXBYTE. BIT } \\
\hline & DEC & Decimal address & \\
\hline & where: & BYTE & DEC \(/ 8\) \\
\\
\hline & BIT & DEC mod 8 & (Modulo division) \\
\hline
\end{tabular}
```

Examples: - Address hexadecimal = 16\#2002
DEC := 8194
BYTE := 8194 / 8 := 1024
BIT := 8194 mod 8 := 2
Bit variable: %MX1024.2
- Address hexadecimal = 16\#3016
DEC := 12310
BYTE := 12310 / 8 := 1538,75 -> 1538
BIT := 12310 mod }8:=
Bit variable: %MX1538.6
- Address hexadecimal = 16\#55AA
DEC := 21930
BYTE := 21930 / 8 := 2741,25 -> 2741
BIT := 21930 mod }8:=
Bit variable: %MX2741.2

```

Calculation of the hexadecimal address from the bit variable:
```

Examples: - Bit variable := %MX515.4
DEC := 515 * 8 + 4 := 4124
Address hex := 16\#101C
- Bit variable := %MX3.3
DEC := 3 * 8 + 3 := 27
Address hex := 16\#001B
- Bit variable := %MX6666.2
DEC := 6666 * 8 + 2 := 53330
Address hex := 16\#D052

```

\section*{Peculiarities for accessing Modbus addresses}

Peculiarities for bit access:
- A WORD in the \%M area is assigned to each Modbus address 0000hex .. FFFFhex.
- Bit addresses 0000hex .. FFFFhex are contained in the word range \%MW0 .. \%MW4095

\section*{Areas protect from read/write access by Modbus client}

As described in 'Protocols and special servers' \(\Leftrightarrow\) Chapter 6.3.4.3.2 "Configuration of Modbus TCP/IP server" on page 1714, one write-protected and one read-protected area can be defined. If you try to write to a write-protected area or to read from a read-protected area, an exception response is generated.

\subsection*{6.8.2.10.5 Local data of the Modbus client}

The address of the area from which data are to be read or to which data are to be written is specified in the function block ModTcpMast at input "Data", via the ADR operator.

For more information about the function blocks use the Library Manager \({ }^{\$ y}\) Chapter 6.5.3 "Library Manager functionality" on page 3773.

For the AC500, the following areas can be accessed using the ADR operator:
- Inputs area (\%l area)
- Outputs area (\%Q area)
- Area of non-buffered variables (VAR .. END_VAR or VAR_GLOBAL END_VAR)
- Addressable flag area (also protected areas for \%M area)
- Area of buffered variables (VAR RETAIN .. END_VAR or VAR_GLOBAL RETAIN .. END_VAR)

\subsection*{6.8.2.10.6 Modbus telegrams}

For a detailed description of the Modbus TCP telegrams and their elements please see the corresponding specifications on public websites.

\section*{Exception response by server}

In operating mode Modbus client, the AC500 does only send requests, if the parameters at the MODMAST inputs are logically correct. Nevertheless, it can happen that a server cannot process the request of the client or that the server cannot interpret the request due to transmission errors or in case it's capabilities are exceeded in any way. In those cases, the server returns an exception response to the client. In order to identify this response as an exception response, the function code returned by the server is a logical OR interconnection of the function code received from the client and the value 80HEX.

\section*{General telegram description}

Table 842: Server response
\begin{tabular}{|l|l|l|}
\hline \multirow{2}{*}{ Error code } & CRC & Low \\
\cline { 2 - 3 } & High & \multicolumn{1}{|l|}{} \\
\hline
\end{tabular}

Possible error codes of the client
\begin{tabular}{|l|l|}
\hline Code & Description \\
\hline 01DEC & \begin{tabular}{l} 
ILLEGAL FUNCTION \\
The server does not support the function requested by the client
\end{tabular} \\
\hline 02DEC & \begin{tabular}{l} 
ILLEGAL DATA ADDRESS \\
Invalid operand address in the server or operand area exceeded
\end{tabular} \\
\hline 03DEC & \begin{tabular}{l} 
ILLEGAL DATA VALUE \\
At least one value is outside the permitted range of values
\end{tabular} \\
\hline 04DEC & \begin{tabular}{l} 
SERVER DEVICE FAILURE \\
An unrecoverable error occurred while the server was attempting \\
to perform the requested action
\end{tabular} \\
\hline \begin{tabular}{l} 
ACKNOWLEDGE \\
Specialized use in conjunction with programming commands.
\end{tabular} \\
\begin{tabular}{l} 
Ses \\
The server has accepted the request and is processing it, but a \\
long duration of time will be required to do so. This response is \\
returned to prevent a timeout error from occurring in the client. \\
The client can next issue a Poll Program Complete message to \\
determine if processing is completed
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline Code & Description \\
\hline 06DEC & \begin{tabular}{l} 
SERVER DEVICE BUSY \\
Specialized use in conjunction with programming commands. \\
The server is engaged in processing a long-duration program \\
command. The client should retransmit the message later when \\
the server is free.
\end{tabular} \\
\hline 07DEC & \begin{tabular}{l} 
NEGATIVE ACKNOWLEDGE \\
Specialized use in conjunction with programming commands. \\
The server cannot perform the programming functions. Client \\
should request diagnostic or error information from server.
\end{tabular} \\
\hline 08DEC & \begin{tabular}{l} 
MEMORY PARITY ERROR \\
Specialized use in conjunction with function codes 20 and 21 and \\
reference type 6, to indicate that the extended file area failed to \\
pass a consistency check. The server attempted to read record \\
file, but detected a parity error in the memory. The client can retry \\
the request, but service may be required on the server device.
\end{tabular} \\
\hline 09DEC & \begin{tabular}{l} 
UNDEFINED \\
Actually not defined by Modbus specification but might be used by \\
particular servers.
\end{tabular} \\
\hline 10DEC & \begin{tabular}{l} 
GATEWAY PATH UNAVAILABLE \\
Specialized use in conjunction with gateways, indicates that the \\
gateway was unable to allocate an internal communication path \\
from the input port to the output port for processing the request. \\
Usually means that the gateway is misconfigured or overloaded.
\end{tabular} \\
\hline \(11 D E C\) & \begin{tabular}{l} 
GATEWAY TARGET DEVICE FAILED TO RESPOND \\
Specialized use in conjunction with gateways, indicates that no \\
response was obtained from the target device. Usually means that \\
the device is not present on the network.
\end{tabular} \\
\hline
\end{tabular}

\section*{Example}

Table 843: Modbus request of the client
\begin{tabular}{|l|l|l|}
\hline Function code: & 01 & Read \(n\) bits \\
\hline Server operand address: & \(4000 \mathrm{HEX}=16384 \mathrm{DEC}\) & \begin{tabular}{l} 
Area for read access disa- \\
bled in server
\end{tabular} \\
\hline
\end{tabular}

Table 844: Modbus response of the server
\begin{tabular}{|l|l|}
\hline Function code: & 81 HEX \\
\hline Error code: & 03 \\
\hline
\end{tabular}

\subsection*{6.8.2.10.7 Processing bits}

General
Some of the Modbus function codes are used to read or write bits (coils, discrete inputs). While a variable of data type WORD can be accessed easily, accessing a stream of bits is complex.

Data type 'BIT' \({ }^{\wedge} \gg\) Chapter 6.4.1.20.5.11 "Data Type 'BIT"' on page 2243 must not be mixed up with data type 'BOOL' \(\Longleftrightarrow\) Chapter 6.4.1.20.5.2 "Data type 'BOOL'" on page 2235. Variables of both types may have values 'TRUE' or 'FALSE'. But while BIT means one single bit only, BOOL requires a byte ( 8 bit ) of memory.

\section*{Modbus client}

When accessing bits in a Server, the local data referred to at Client function blocks input data is always expected to be of format BOOL.
```

VAR
abBoolArray : ARRAY [0..15] OF BOOL;
ModMast : ModTcpMast;
END VAR
abBoolArray[0] := TRUE;
abBoolArray[15] := FALSE;
ModMast(Execute:= ModMastExecute,
Eth:= ETH1,
IPAdr:= IP_ADR_STRING_IO_DWORD('192.168.0.100'),
UnitID:= 0,
Fct:= 15,
Addr:= 16\#0000,
Nb:= SIZEOF(abBoolArray),
Data:= ADR(abBoolArray));

```

\section*{Modbus server}

\section*{Using the bit offset}

The simplest way to access a certain bit within a larger variable is to directly use the bit offset ( 0 based; \({ }^{〔}\) Chapter 6.4.1.20.4.11 "Bit Access in Variables" on page 2229).
```

VAR
awWordVariable AT \&MWO : ARRAY [0..7] OF WORD;
END VAR
awWordVariable[0].0 := TRUE;
awWordVariable[7].15 := FALSE;

```

\section*{Defining symbolic names for the bit offsets}

A more convenient way to access bits e.g. within a word is to define a symbolic name for each single offset \({ }^{*}\) Chapter 6.4.1.20.4.11 "Bit Access in Variables" on page 2229.
```

VAR_GLOBAL CONSTANT
(*bit offsets in avWordVariable[0]*)
NameBitOffset_0_00 : INT := 0;
NameBitOffset_0_01 : INT := 1;
(*...*)
NameBitOffset_0_15 : INT := 15;
(*bit offsets in avWordVariable[7]*)
NameBitOffset_7_00 : INT := 0;
NameBitOffset_7_01 : INT := 1;
(*...*)
NameBitOffset_7_15 : INT := 15;
END VAR

```
VAR
awWordVariable AT sMWO : ARRAY [0..7] OF WORD;
END_VAR
awWordVariable[0].NameBitOffset_0_00 := TRUE;
awWordVariable[7].NameBitOffset_7_15 := FALSE;

\section*{Defining a data type}

A further alternative is to define your own data types \(\Leftrightarrow\) Chapter 6.4.1.21.2.9 "Object 'DUT"" on page 2461 according to the requirements of your particular application \({ }^{\star}\) "Symbolic bit access in structure variables" on page 2230.
```

TYPE ControlWordType :
STRUCT
NameBitOffset_00 : BIT;
NameBitOffset_01 : BIT;
NameBitOffset_02 : BIT;
NameBitOffset_03 : BIT;
NameBitOffset_04 : BIT;
NameBitOffset_05 : BIT;
NameBitOffset_06 : BIT;
NameBitOffset_07 : BIT;
NameBitOffset_08 : BIT;
NameBitOffset_09 : BIT;
NameBitOffset_10 : BIT;
NameBitOffset_11 : BIT;
NameBitOffset_12 : BIT;
NameBitOffset_13 : BIT;
NameBitOffset_14 : BIT;
NameBitOffset_15 : BIT;
END_STRUCT
END_TYPE

```

\section*{VAR}
atTypeVariable AT \(\$\) MWO : ARRAY [0..7] OF ControlWordType;
END VAR
```

atTypeVariable[0].NameBitOffset_00 := TRUE;
atTypeVariable[7].NameBitOffset_15 := FALSE;

```

\section*{Defining a complex data type}

In case your application requires some more complex data types you can combine data types (DUT; «s "Symbolic bit access in structure variables" on page 2230).
```

TYPE ControlWordOType :
STRUCT
NameBitOffset_0_00 : BIT;
NameBitOffset_0_01 : BIT;
NameBitOffset_0_02 : BIT;
NameBitOffset_0_03 : BIT;
NameBitOffset_0_04 : BIT;
NameBitOffset_0_05 : BIT;
NameBitOffset_0_06 : BIT;
NameBitOffset_0_07 : BIT;
NameBitOffset_0_08 : BIT;
NameBitOffset_0_09 : BIT;
NameBitOffset_0_10 : BIT;
NameBitOffset_0_11 : BIT;
NameBitOffset_0_12 : BIT;
NameBitOffset_0_13 : BIT;
NameBitOffset_0_14 : BIT;
NameBitOffset_0_15 : BIT;
END_STRUCT
END_TYPE

```

ControlWord1 Type
TYPE ControlWord1Type :
STRUCT
NameBitOffset_1_00 : BIT;
NameBitOffset_1_01 : BIT;
NameBitOffset_1_02 : BIT;
NameBitOffset_1_03 : BIT;
NameBitOffset_1_04 : BIT;
NameBitOffset_1_05 : BIT;
NameBitOffset_1_06 : BIT;
NameBitOffset_1_07 : BIT;
NameBitOffset_1_08 : BIT;
NameBitOffset_1_09 : BIT;
NameBitOffset_1_10 : BIT;
NameBitOffset_1_11 : BIT;
NameBitOffset_1_12 : BIT;
NameBitOffset_1_13 : BIT;
NameBitOffset_1_14 : BIT;
NameBitOffset_1_15 : BIT;
END_STRUCT
END_TYPE

TYPE ControlWordListType :
STRUCT
ControlWordo : ControlWordOType;
ControlWord1 : ControlWord1Type;
END_STRUCT
END TYPE
VAR
atWordListType AT \$MWO : ControlWordListType; END_VAR
atWordListType.ControlWord0.NameBitOffset_0_00 := TRUE;
atWordListType.ControlWord1.NameBitOffset_1_15 := FALSE;

\section*{Pack／unpack BOOL variables}

In case you prefer variables of type BOOL you can use the functions for packing MEM＿Pack＿BitsToByte and unpacking MEM＿UnpackWord of the CAA＿Memory．library，which can be found with the Library Manager \({ }^{\star}>\) Chapter 6．5．3＂Library Manager functionality＂ on page 3773.

VAR
abBoolVariable ：ARRAY［0．．15］OF BOOL； wWordVariable0 ：AT s⿳⺈⿴囗十一日⿹勹巳 WORD； wWordVariable1 ：AT sMW1 WORD； END＿VAR

VAR
Unpack ：Mem．UnpackWord；
END＿VAR
abBoolVariable［0］：＝TRUE；
abBoolVariable［15］：＝FALSE
wWordVariable0 ：＝Mem．PackBitsToWord（abBoolVariable［0］，abBoolVariable［1］，abBoolVariable［2］，abBoolVariable［3］， abBoolVariable［4］，abBoolVariable［5］，abBoolVariable［6］，abBoolVariable［7］， abBoolVariable［8］，abBoolVariable［9］，abBoolVariable［10］，abBoolVariable［11］， abBoolVariable［12］，abBoolVariable［13］，abBoolVariable［14］，abBoolVariable［15］）；
wWordVariable1 ：＝1；
Unpack（wValue：＝wWordVariable1，
\(x\) Bit0 \(\Rightarrow\) abBoolVariable［0］，xBit1 \(\Rightarrow\) abBoolVariable［1］，xBit2 \(\Rightarrow\) abBoolVariable［2］，xBit3 \(\Rightarrow\) abBoolVariable［3］， \(x\) Bit4 \(\Rightarrow\) abBoolVariable［4］，xBit5 \(\Rightarrow\) abBoolVariable［5］，xBit6 \(\Rightarrow\) abBoolVariable［6］，xBit7 \(\Rightarrow\) abBoolVariable［7］， \(x\) xit \(8 \Rightarrow\) abBoolVariable［8］，xBit9 \(\Rightarrow\) abBoolVariable［9］，xBit10 \(\Rightarrow\) abBoolVariable［10］，xBit11 \(\Rightarrow\) abBoolVariable［11］， \(x B i t 12 \Rightarrow\) abBoolVariable［12］，xBit13 \(\Rightarrow\) abBoolVariable［13］，xBit14＝＞abBoolVariable［14］，xBit15 \(\Rightarrow\) abBoolVariable［15］）；

\section*{6．8．2．10．8 Function block ModTcpMast}

These function blockis only required for the operating mode Modbus client．It handles the communication（transmission of telegrams to the servers and receipt of telegrams from the servers）．The function block can be used for the Ethernet interfaces of the controller．

ModTcpMast is contained in the library ABB＿ModbusTcp＿AC500．

\section*{6．8．2．11 Communication with PROFINET redundancy}

In a high availability system one of the following redundancy solutions can be configured：
－AC500 High Availability CS31 redundancy（HA－CS31）
－AC500 High Availability Modbus redundancy（HA－Modbus TCP）
„ \(\boldsymbol{y}\) Chapter 6．5．9．1．1＂The AC500 High Availability system＂on page 3859
－PROFINET redundancy based on HA－Modbus TCP without CI52x modules．
Normally，two PROFINET IO controllers CM579－PNIO connect to the devices simultaneously． One controller establishes a primary connection that is identical to a regular PROFINET connec－ tion．The second controller establishes a special＇backup＇connection．This connection doesn＇t contain any valid output data and doesn＇t allow the secondary controller to change any options on the devices．If the primary controller fails，the second parallel controller takes over the tasks of the first．

How to set－up and configure PROFINET redundancy communication based on HA－Modbus TCP（without CI52xs modules）is described in the application example PROFINET redundancy．

Network redun－The Media Redundancy Protocol allows in smaller systems rings of Ethernet switches to over－ dancy come any single failure with a maximum recovery times in the following range： \(10 \mathrm{~ms}, 30 \mathrm{~ms}\) ， 200 ms and 500 ms ．
ⓨ Chapter 6．3．4．1．2．2．7．4＂Network redundancy＂on page 1660

\subsection*{6.8.2.12 Fast counters in AC500 devices}

For AC500 devices the function "fast counter" is available in S500 I/O modules as of firmware version V1.3.

For AC500-eCo V3 devices the function "fast counter" is available in onboard I/Os of PM50x2 modules, according to the CPU type, the fast inputs have different functionality or frequency.

Integrated fast counters are only available for digital I/O modules.
The digital I/O modules on the I/O bus contain two fast counters each.
If the counter is used, it needs up to 2 digital inputs and one digital output.
If the fast counter is deactivated, the inputs and outputs reserved for the counter can be used for other tasks.
\(\Leftrightarrow\) Ehapter 6.3.2.13.9 "Fast counter" on page 1574.
A fast counter is available in the following constellations:
- In digital I/O modules, connected to an AC500 processor module.
- In AC500-eCo V3 processor modules PM50x2 with onboard I/Os
- In CANopen communication interface modules.
- In Modbus, PROFIBUS and PROFINET communication interface modules and in the connected digital I/O modules.
- In digital I/O modules, connected to an EtherCAT communication interface module.

Fast counter integrated in S500 modules

The following table shows the S500 modules which contain a fast counter and which of the digital inputs and outputs are reserved for the counter.
\begin{tabular}{|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Module} & \multicolumn{2}{|l|}{Assigned inputs \({ }^{1}\) )} & Assigned & \multirow[t]{2}{*}{Remarks} \\
\hline & Channel A & Channel B & Channel C \({ }^{2}\) ) or (CF) & \\
\hline DA501 & DC16 & DC17 & DC18 & \multirow[t]{6}{*}{The counter func tion is not available if the modules are mounted on the communication interface modules CI581CN or CI582-CN} \\
\hline DA502 & DC16 & DC17 & \begin{tabular}{l}
DC18 - in mode 1 and mode 2 \\
DOO - in mode 101 and mode \(102{ }^{3}\) )
\end{tabular} & \\
\hline DC522 & C8 & C9 & C10 & \\
\hline DC523 & C16 & C17 & C18 & \\
\hline DC532 & C24 & C25 & C26 & \\
\hline DI524 & 124 & 125 & No hardware output available & \\
\hline DX522 & 10 & 11 & The counter does not activate any relay output & \\
\hline
\end{tabular}
\begin{tabular}{|l|l|l|l|l|}
\hline \multirow{2}{|l|}{ Module } & \multicolumn{2}{|l|}{ Assigned inputs \({ }^{1}\) ) } & \begin{tabular}{l} 
Assigned \\
output
\end{tabular} & Remarks \\
\cline { 2 - 4 } & Channel A & Channel B & \begin{tabular}{l} 
Channel C \({ }^{2}\) ) or \\
(CF)
\end{tabular} & \\
\hline \begin{tabular}{l} 
CI501-PNIO, \\
CI541-DP, \\
CI581-CN, \\
CI521-MODTCP
\end{tabular} & DI0 & DI1 & DO0 & \\
\hline \begin{tabular}{l} 
CI502-PNIO, \\
CI542-DP, \\
CI582-CN; \\
CI522-MODTCP
\end{tabular} & DI8 & DI9 & DO8 & \\
\hline
\end{tabular}
\({ }^{1}\) ) The two hardware inputs (channels \(A\) and \(B\) ) are also and always available within the normal process image, irrespective of the operating mode of the counter.
\({ }^{2}\) ) The hardware output channel C is activated by the fast counter only in the operating modes 1 and 2.
\({ }^{3}\) ) Especially for module DA502: The counter operating mode 101 is the same as mode 1 , but the assigned output is DO0 instead of DC18. Also the counter operating mode 102 is the same as mode 2, but the assigned output is DO0 instead of DC18.

The counter function is performed within the communication interface module and, accordingly, in the digital I/O module(s). It works independently of the user program and is therefore able to respond quickly to external signals. A simultaneous counter operation of several digital I/O modules is possible.
Each module counter can be configured for one out of 10 possible modes. The desired operating mode is selected in the PLC configuration using module parameters. After that, it is activated during the initialization phase (power-on, cold start, warm start).
The data exchange to and from the user program is performed using input and output operands. While integrating a module containing a fast counter in the PLC configuration, the necessary operands are created and reserved immediately. Thus, a counter implementation carried out later on does not cause an address shift.

Features independent of the fast counter operating mode
- The pulses at the fast counters' inputs or the evaluated signals of the traces \(A\) and \(B\) in case of incremental position sensors are counted.
- The counting frequencies of the communication interface modules of PROFINET, PROFIBUS and CANopen are max. 200 kHz (in modes 1 to 6), max. 50 kHz (in mode 7), max. 35 kHz (in mode 9), and max. 20 kHz (in mode 10).
- If the modules DA501, DC522, DC523, DC532 are used, each counting input must be circuited externally in series with a resistor of \(470 \Omega / 1 \mathrm{~W}\), in order to safely avoid influences from the deactivated module outputs to the connected sensors.
- The positive signal edges are counted, if not noted differently.
- By setting the operating mode 0 , the counting function is switched off. In this case, the reserved inputs and outputs can be used for other tasks. Simultaneous use of these terminals for the fast counter and other signals must be avoided.
- The fast counter's actual value is provided as a double word (32 bits).
- The fast counter can count upwards in all operating modes. It counts beginning at the start value (set value) up to the end value (max. from 0 to 4,294,967,295 or hexadecimal from 00000000 to FF FF FF FF. After reaching 4,294,967,295, the counter jumps with the next pulse to 0 . When the counter reaches the programmed end value, the counter output is stored permanently as CF = TRUE (end value reached). Only when the fast counter is set again (set value), CF is reset to FALSE.

Further informa- - Operating modes of the fast counter \(\left.{ }^{\mu}\right\rangle\) Chapter 6.3.2.13.9.1.3 "Operating modes"
tion
on page 1577
- Configuration of the fast counter \({ }^{4} y\) Chapter 6.3.2.13.9 "Fast counter" on page 1574

\subsection*{6.8.2.13 \\ Fast counter in AC500-eCo V3}

For AC500 devices the function "fast counter" is available in S500 I/O modules as of firmware version V1.3.

For AC500-eCo V3 devices the function "fast counter" is available in onboard I/Os of PM50xx.

The AC500-eCo V3 processor modules with onboard I/Os provide some special functionality on the digital inputs or digital outputs. Fast counter, encoder inputs, interrupt inputs or PWM/PTO outputs are available depending on the device used.
The fast counter functionality can be activated within the onboard I/O configuration.
The fast counter can work in pulse/direction mode or A/B track counter mode.
\& Further information about the operating modes of the fast counter
\(\Leftrightarrow\) Further information about the configuration of the fast counter

As AC500-eCo V3 PLCs provide the fast counters via their onboard I/Os not only the correct power distribution to the PLC has to be made, but also the correct wiring to the signal wires. How to connect and use fast counters in AC500-eCo V3 PLCs is described in an application example.

The pulse/direction counter detects the rising edge of the counter input. It will increase or decrease the count value (depending on the direction input) at every rising edge.

The A/B track counter is used to count the signal from an encoder.
The counter can count with quad phases. In the following the behavior of the A/B track counter is described.

\subsection*{6.8.2.14 Onboard I/O on AC500-eCo V3 processor modules}

\subsection*{6.8.2.14.1 Intended purpose}

The AC500-eCo V3 processor modules have onboard I/Os which provide several functionalities. According to the CPU type, the number or the functionality of the onboard I/Os can be different.

Table 845: Numbers and types of the onboard I/Os
\begin{tabular}{|l|l|l|l|}
\hline Processor module & \begin{tabular}{l} 
No. and type of dig- \\
ital inputs
\end{tabular} & \begin{tabular}{l} 
No. and type of dig- \\
ital outputs
\end{tabular} & \begin{tabular}{l} 
No. and type of con- \\
figurable inputs/out- \\
puts
\end{tabular} \\
\hline PM5012-T-ETH & \begin{tabular}{l}
6 \\
24 V DC \\
(one isolation group)
\end{tabular} & \begin{tabular}{l}
0.5 A max., transistor \\
(one isolation group)
\end{tabular} & None \\
\hline PM5012-R-ETH & \begin{tabular}{l}
6 \\
24 V DC \\
(one isolation group)
\end{tabular} & \begin{tabular}{l}
4 \\
(two isolation groups)
\end{tabular} & None \\
\hline PM5032-T-ETH & \begin{tabular}{l}
12 \\
24 V DC \\
(one isolation group)
\end{tabular} & \begin{tabular}{l}
8 \\
(one isolation group) \\
(o.5 A max., transistor
\end{tabular} & \begin{tabular}{l}
24 V DC input or \\
\(0.5 \mathrm{~A} \mathrm{max.} transistor\), \\
output \\
(one isolation group)
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|}
\hline Processor module & No. and type of digital inputs & No. and type of digital outputs & No. and type of configurable inputs/outputs \\
\hline PM5032-R-ETH & \[
\begin{array}{|l}
\hline 12 \\
24 \mathrm{~V} \text { DC } \\
\text { (one isolation group) }
\end{array}
\] & \begin{tabular}{l}
6 \\
2 A max., relay (two isolation groups)
\end{tabular} & \begin{tabular}{l}
2 \\
24 V DC input or \\
0.5 A max., transistor output \\
(one isolation group)
\end{tabular} \\
\hline PM5052-T-ETH & \[
\begin{aligned}
& 12 \\
& 24 \mathrm{~V} \text { DC } \\
& \text { (one isolation group) }
\end{aligned}
\] & 8 0.5 A max., transistor (one isolation group) & \begin{tabular}{l}
2 \\
24 V DC input or \\
0.5 A max., transistor output \\
(one isolation group)
\end{tabular} \\
\hline PM5052-R-ETH & \[
\begin{array}{|l}
\hline 12 \\
24 \mathrm{~V} \text { DC } \\
\text { (one isolation group) }
\end{array}
\] & 6 2 A max., relay (two isolation groups) & \begin{tabular}{l}
2 \\
24 V DC input or \\
0.5 A max., transistor output \\
(one isolation group)
\end{tabular} \\
\hline PM5072-T-2ETH & \[
\begin{aligned}
& 12 \\
& 24 \mathrm{~V} \text { DC } \\
& \text { (one isolation group) }
\end{aligned}
\] & 8 0.5 A max., transistor (one isolation group) & \begin{tabular}{l}
2 \\
24 V DC input or \\
0.5 A max., transistor output \\
(one isolation group)
\end{tabular} \\
\hline PM5072-T-2ETHW & \[
\begin{aligned}
& \hline 12 \\
& 24 \mathrm{~V} \text { DC } \\
& \text { (one isolation group) }
\end{aligned}
\] & 8 0.5 A max., transistor (one isolation group) & \begin{tabular}{l}
2 \\
24 V DC input or \\
0.5 A max., transistor output \\
(one isolation group)
\end{tabular} \\
\hline PM5082-T-2ETH & \[
\begin{aligned}
& 12 \\
& 24 \mathrm{~V} \text { DC } \\
& \text { (one isolation group) }
\end{aligned}
\] & 8 0.5 A max., transistor (one isolation group) & \begin{tabular}{l}
2 \\
24 V DC input or \\
0.5 A max., transistor output \\
(one isolation group)
\end{tabular} \\
\hline
\end{tabular}

\subsection*{6.8.2.14.2 Functionality}
\begin{tabular}{|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Parameter} & \multicolumn{4}{|l|}{Value} \\
\hline & PM5012-T-ETH & PM5012-R-ETH & \begin{tabular}{l}
PM5032-T-ETH \\
PM5052-T-ETH \\
PM5072- \\
T-2ETH(W) \\
PM5082-T-2ETH
\end{tabular} & \begin{tabular}{l}
PM5032-R-ETH \\
PM5052-R-ETH
\end{tabular} \\
\hline Digital inputs & \multicolumn{2}{|l|}{6} & \multicolumn{2}{|l|}{12} \\
\hline \begin{tabular}{l}
Functionality of digital inputs \\
(encoder, fast counter, counter, interrupt)
\end{tabular} & \multicolumn{2}{|l|}{\multirow[t]{3}{*}{\begin{tabular}{l}
6 DI fast input 24 V DC (max. 5 kHz) \\
usable as \\
- 6 DI 24 V DC standard \\
- 2 channel 5 kHz encoder with frequency measurement or \\
- 2 channel 5 kHz encoder with frequency measurement and with touch/reset using standard DI or \\
- 2 fast counter ( 5 kHz ) \\
- 4 DI as interrupt input with 1 dedicated interrupt task and input information
\end{tabular}}} & \multicolumn{2}{|l|}{\begin{tabular}{l}
4 DI fast input 24 V DC (max. 200 kHz) \\
usable as \\
- 4 DI 24 V DC standard or \\
- 4 fast counter ( 100 kHz ) or \\
- \(2 \mathrm{~A} / \mathrm{B}\) encoder \((200 \mathrm{kHz})\) with frequency measurement or \\
- 2 full A/B encoders 0 and 1 (200 kHz ) with frequency measurement and with touch/reset using standard highspeed ( 5 kHz ) DI \\
- 1 full A/B encoder 0 ( 200 kHz ) with frequency measurement and optional with touch/reset using 2 touch/sync inputs with A/B encoder 0
\end{tabular}} \\
\hline & & & \multicolumn{2}{|l|}{\begin{tabular}{l}
4 DI fast input 24 V DC ( 5 kHz ) usable as \\
- 4 DI 24 V DC standard or \\
- 4 DI as interrupt input with 1 dedicated interrupt task and input information \\
- 4 touch/sync inputs with \(A / B\) encoder 0 or 1
\end{tabular}} \\
\hline & & & \multicolumn{2}{|l|}{4 standard DI 24 V DC} \\
\hline Digital outputs & \multicolumn{2}{|l|}{4} & 8 & 6 \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Parameter} & \multicolumn{4}{|l|}{Value} \\
\hline & PM5012-T-ETH & PM5012-R-ETH & \begin{tabular}{l}
PM5032-T-ETH \\
PM5052-T-ETH \\
PM5072- \\
T-2ETH(W) \\
PM5082-T-2ETH
\end{tabular} & PM5032-R-ETH PM5052-R-ETH \\
\hline Functionality of digital outputs & \begin{tabular}{l}
4 fast output DO-T \\
24 V DC/0.5 A (max. 5 kHz ) usable as \\
- 4 DO-T 24 V DC/0.5 A or \\
- 4 PWM Note: The speed must be limited below 100 Hz . The low speed PWM can be used for heating control. \\
- 4 limit switch
\end{tabular} & 4 DO-R
24 V DC /
\(120 / 240 \mathrm{~V}\) AC 2A
in 2 groups & \begin{tabular}{l}
4 fast output DO-T \\
24 V DC (100 kHz) \\
usable as \\
- 4 DO-T 24 V DC/0.5 A \\
- 4 limit/ switch outputs for encoder/ counter or \\
- 4 PWM (30 kHz, \(2 \mu \mathrm{~s}\) accuracy and maximum duty 95 \%) or \\
- 2 PTO (200 kHz) CW/CCW or Pulse/Direction \\
- 4 PTO (PWM) 100... 200 kHz Pulse/Direction using standard output
\end{tabular} & \[
\begin{aligned}
& \hline 6 \text { DO-R } \\
& 24 \text { V DC / } \\
& 120 / 240 \text { V AC 2A } \\
& \text { in } 2 \text { groups }
\end{aligned}
\] \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Parameter} & \multicolumn{4}{|l|}{Value} \\
\hline & PM5012-T-ETH & PM5012-R-ETH & \begin{tabular}{|l|}
\hline PM5032-T-ETH \\
PM5052-T-ETH \\
PM5072- \\
T-2ETH(W) \\
PM5082-T-2ETH \\
\hline
\end{tabular} & PM5032-R-ETH PM5052-R-ETH \\
\hline & & & \begin{tabular}{l}
4 fast output DO-T \\
24 V DC/0.5 A (5 kHz ) (max. 5 kHz ) \\
usable as \\
- 4 DO-T 24 V \\
DC/0.5 A \\
- 4 limit/ switch outputs for encoder/ counter or \\
- 4 PWM Note: The speed must be limited below 100 Hz . The low speed PWM can be used for heating control. \\
- 4 direction outputs together with 4 high speed pulses for up to 4 PTO Pulse/Direction outputs up to 200 kHz
\end{tabular} & \\
\hline Digital inputs/ outputs, configurable & - & - & 2 & 2 \\
\hline Functionality of digital inputs/ outputs, configurable & - & - & \begin{tabular}{l}
2 DC 24 V DC \\
- 2 standard I/Os configurable
\end{tabular} & \begin{tabular}{l}
2 DC 24 V DC usable as \\
- 2 DC standard (DI 24 V DC or DO-T) or \\
- 2 PWM (30 kHz ) or \\
- 1 PTO (200 kHz ) as Pulse/Direction or CW/CCW
\end{tabular} \\
\hline LED displays & \multicolumn{4}{|l|}{For signal states} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|l|l|l|}
\hline Parameter & \multicolumn{4}{|l|}{ Value } \\
\cline { 2 - 5 } & PM5012-T-ETH & PM5012-R-ETH & \begin{tabular}{l} 
PM5032-T-ETH \\
PM5052-T-ETH
\end{tabular} & \begin{tabular}{l} 
PM5032-R-ETH \\
PM5052-R-ETH
\end{tabular} \\
& & & \begin{tabular}{l} 
PM5072- \\
T-2ETH(W) \\
PM5082-T-2ETH
\end{tabular} & \\
\hline \begin{tabular}{l} 
Internal power \\
supply
\end{tabular} & \multicolumn{4}{|l|}{ Via processor module } \\
\hline \begin{tabular}{l} 
External power \\
supply
\end{tabular} & \multicolumn{4}{|l|}{ Via UP and ZP terminal } \\
\hline
\end{tabular}

\subsection*{6.8.2.15 Simple motion}

\subsection*{6.8.2.15.1 Introduction}

The AC500-eCo V3 PLC provide several hardware and software features allowing to realize some motion application.
Specific fast onboard I/O and dedicated software library function blocks (simple motion) are available and can manage up to 2 axis on the CPU.
The simple motion capability is based on a library for the onboard I/O and some motion control blocks allowing point-to-point or velocity control.
All the AC500-eCo V3 PLC from Basic, Standard or Pro type offer dedicated feature according to their performance classes.
\begin{tabular}{|c|c|c|c|c|c|}
\hline \multirow[t]{3}{*}{} & \multicolumn{2}{|c|}{Basic} & \multicolumn{2}{|c|}{Standard} & Pro \\
\hline & \multicolumn{2}{|c|}{PM5012-x-ETH} & \multicolumn{2}{|l|}{PM5032-x-ETH / PM5052-x-ETH} & PM50x2-T-2ETH \\
\hline & Relay outputs & Transistor outputs & Relay outputs & Transistor outputs & Transistor outputs \\
\hline High-speed counter (HSC) & \multicolumn{2}{|c|}{Up to 2 ( 5 kHz )} & \multicolumn{3}{|c|}{Up to 4 ( 100 kHz )} \\
\hline Frequency measurement & \multicolumn{2}{|c|}{Up to 2 ( 5 kHz )} & \multicolumn{3}{|c|}{Up to \(2(200 \mathrm{kHz})\)} \\
\hline A/B encoder & \multicolumn{2}{|l|}{\(1 \mathrm{~A} / \mathrm{B}\) simple encoder ( 5 kHz ) with sync/reset} & \multicolumn{3}{|l|}{Up to \(2 \mathrm{~A} / \mathrm{B}\) encoder 200 kHz with sync/reset inputs} \\
\hline Interrupt inputs & \multicolumn{2}{|c|}{Up to 4} & \multicolumn{3}{|c|}{Up to 4} \\
\hline \multirow[t]{2}{*}{Pulse-train output (PTO)} & \multicolumn{2}{|c|}{\multirow[t]{2}{*}{-}} & \multirow[t]{2}{*}{\begin{tabular}{l}
1 \\
pulse/direction \\
or CW/CCW both mode with 200 kHz
\end{tabular}} & \multicolumn{2}{|l|}{\begin{tabular}{l}
Up to 2 \\
pulse/direction or \\
CW/CCW \\
both mode with 200 kHz
\end{tabular}} \\
\hline & & & & Pulse/direction using fast outpu and standard ou tion on softwa & \begin{tabular}{l}
o 4 \\
th 100... 200 kHz hannels for pulse uts and for direcmotion function ck.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|}
\hline \multirow{4}{*}{} & \multicolumn{2}{|c|}{ Basic } & \multicolumn{2}{c|}{ Standard } & Pro \\
\cline { 2 - 6 } & \multicolumn{2}{|c|}{ PM5012-x-ETH } & PM5032-x-ETH / PM5052-x-ETH & PM50x2-T-2ETH \\
\cline { 2 - 6 } & Relay outputs & \begin{tabular}{c} 
Transistor \\
outputs
\end{tabular} & Relay outputs & \begin{tabular}{c} 
Transistor \\
outputs
\end{tabular} & \begin{tabular}{c} 
Transistor \\
outputs
\end{tabular} \\
\hline \begin{tabular}{c} 
Pulse-width mod- \\
ulation (PWM)
\end{tabular} & - & Up to \(4(100 \mathrm{~Hz})\) & Up to \(2(30 \mathrm{kHz})\) & Up to \(4(30 \mathrm{kHz})\) \\
\hline Limit switches & \multicolumn{3}{|c|}{-} & Up to 2 & Up to 8 \\
\hline
\end{tabular}

\subsection*{6.8.2.15.2 Hardware components for motion control}

Basic CPU - PM5012-R-ETH and PM5012-T-ETH


Fig. 361: Example: PM5012-T-ETH
1 High-speed counter 5 kHz frequency measurement interrupt I/O
2 Pulse-width modulation output

\section*{Standard and Pro CPU - PM5032-x-ETH/PM5052-x-ETH/PM5072-T-2ETH(W)/PM5082-T-2ETH}


Fig. 362: Example: PM5052-T-ETH
1 High-speed counter 100 kHz and \(5 \mathrm{kHz} \mathrm{A} / \mathrm{B}\) encoder 200 kHz interrupt I/O standard inputs
2 Pulse-train output \(100 \mathrm{kHz} / 200 \mathrm{kHz}\) pulse-width modulation 30 kHz limit switch standard outputs
3 Drives, encoder, stepper motor
For PLC with relay outputs, the input features are identical.
The digital configurable inputs or outputs can be used for pulse-train output/pulse-width modulalion functions.

\subsection*{6.8.2.15.3 System technology}

General
The following chapters describe the system technology of the AC500-eCo V3 using motion examples.

The simple motion set of function blocks is standard part of the system libraries for AC500-eCo VB.

\section*{Simple motion}

Velocity, point2pointpositioning

\section*{ECO_V3_ONBOARD.Library supports the onboard 10}

\section*{ESTP...}
```

...ENCODER(2), ..FORWARD_COUNTER(4),

```
..INTERRUPT(4), ..LIMIT_SWITCH(8), ..PWM(8),
..STEPPER(2)

\section*{Use the onboard I/Os as encoder with A and B signals}

\section*{Parameter configuration}

The onboard \(I / O\) accept encoder signal \(A\) and \(B\). When configure the encoder track \(A\), the encoder track \(B\) will be automatically inserted.
The user can configure the following input channel as encoder input.
- "Encoder 0 Track - A": Input channel 4
- "Encoder 0 Track - B": Input channel 5
- "Encoder 1 Track - A": Input channel 6
- "Encoder 1 Track - B": Input channel 7

After configuring the encoder input channel, the user can configure the touch/reset for the respective encoder channel.

More information about this topic can be found in the chapter 'Configuration the I/O channel'. « Chapter 6.3.2.5 "Configure the onboard I/O channel" on page 1483

\section*{Function block}

OBIOEncoder
Counter
\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|c|}{OBIOEncoderCounter} \\
\hline Enable BOOL & BOOL Busy \\
\hline Channel BYTE & BOOL Error \\
\hline Set BOOL & Error_ID ErrorID \\
\hline CounterValueSet UDIVT & UDINT CounterValue \\
\hline EnableLimit BOOL & BOOL ReadyRef \\
\hline LimitValueMax UDIVT & BOOL ReadyTouch \\
\hline Mode BYTE & BOOL NewFrequencyResult \\
\hline EnableRef BOOL & BOOL MinFrequencyWarning \\
\hline EnableTouch BOOL & BOOL DirectionChange \\
\hline MaxFrequency UDIVT & UDINT CounterTouchValue \\
\hline & LREAL CounterFrequency \\
\hline
\end{tabular}

If "Enable" is TRUE, the "OBIOEncoderCounter" instruction increments the counter by one base on the input.

If "Set" bit is TRUE, the "OBIOEncoderCounter" instruction moves the "CounterValueSet" to the "CounterValue".


If "Enable" is TRUE, the "OBIOEncoderCounter" instruction increments the counter by one based on the input.

If "EnableLimit" bit is TRUE, the accumulated value continues incrementing.
After "CounterValue" reaches the "LimitValueMax", the "OBIOEncoderCounter" instruction writes 0 to the "CounterValue".

"Encoder Counter Mode": \(0=\) " \(90^{\circ}\) Mode"

In this encoder counter mode, an increasing count results when input \(B\) is \(90^{\circ}\) ahead of input \(A\). The count is initiated on the rising edge of input \(A\), and the direction of the encoder is clockwise (positive).
The module produces a decreasing count when input \(A\) is \(90^{\circ}\) ahead of input \(B\).
The count is initiated on the falling edge of input \(A\), and the direction is counterclockwise (negative).
By monitoring both the number of pulses and the phase relationships of input \(A\) and \(B\), you can accurately determine the position and direction of the rotation.

"Encoder
Counter Mode": 1 = "pulse/ direction"

In this encoder counter mode, the count increases or decreases based on the state of input B, which can be a random signal.

If input \(B\) is high, the counter will count down.
If input \(B\) is low the counter counts up.
Counting is done on the leading edge of input \(A\).

Count Up


Frequency Value \(\square\)

If "Enable" is TRUE, the "OBIOEncoderCounter" instruction increments the counter by one based on the input.
If "EnableRef" bit is TRUE, the "OBIOEncoderCounter" instruction is ready to receive the touch/ reset input.

If the "Touch/Reset" input is TRUE, the current "CounterValue" will be replaced by the "CounterValueSet".


If "Enable" is TRUE, the "OBIOEncoderCounter" instruction increments the counter by one based on the input.

If "EnableTouch" bit is TRUE, the "OBIOEncoderCounter" instruction is ready to receive the "Touch/Reset" input.

If the "Touch/Reset" input is TRUE, the current "CounterValue" will be captured and written to the "CounterTouchValue".


\section*{Use the onboard I/Os as forward counter}

\section*{Parameter configuration}

The onboard I/O accept pulse input as forward counter.
User can configure the following input channel as forward counter.
- "Forward Counter 0": Input channel 4
- "Forward Counter 1": Input channel 5
- "Forward Counter 2": Input channel 6
- "Forward Counter 3": Input channel 7

More information about this topic can be found in the chapter 'Configuration the I/O channel'. \& Chapter 6.3.2.5 "Configure the onboard I/O channel" on page 1483

\section*{Function block}

\section*{OBIOForward}

\section*{Counter}
\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|c|}{OBIOForwardCounter} \\
\hline Enable BOOL & BOOL Busy \\
\hline Channel BYTE & BOOL Error \\
\hline Set BOOL & Error_ID ErrorID \\
\hline CounterValueSet UDIVT & UDINT CounterValue \\
\hline EnableLimit BOOL & \\
\hline LimitValueMax UDIVT & \\
\hline
\end{tabular}

If "Enable" is TRUE, the "OBIOForwardCounter" instruction increments the counter by one based on the input.
If "Set" bit is TRUE, the "OBIOForwardCounter" instruction moves the "CounterSetValue" to the "CounterValue".


If "Enable" is TRUE, the "OBIOForwardCounter" instruction increments the counter by one based on the input.

If "EnableLimit" bit is TRUE, the accumulated value continues incrementing.
After "CounterValue" reaches the "LimitValueMax", the "OBIOForwardCounter" instruction writes 0 to the "CounterValue".


\section*{Use the onboard I/Os as interrupt input with dedicated interrupt task}

\section*{Parameter configuration}

The onboard I/O input can be configured as interrupt input to trigger the interrupt task.

The user can configure the following input channel as interrupt input.
- "Interrupt input 0" : Input Channel 0
- "Interrupt input 1": Input Channel 1
- "Interrupt input 2" : Input Channel 2
-"Interrupt input 3" : Input Channel 3

More information about this topic can be found in the chapter 'Configuration the I/O channel'. \& Chapter 6.3.2.5 "Configure the onboard I/O channel" on page 1483

After configuring the parameter, the user need to create a new task with the "Type" set to "External" and the "External event" set to "OnBoard_Binary_Input".


\section*{Function block}



The "OBIOInterruptPara" instruction is configured for 4 interrupt inputs.
If "EnableInterrupt" bit is TRUE, the "OBIOInterruptInfo" instruction is ready to receive the interrupt input.

If the interrupt input is TRUE, the interrupt task will be executed.

If the second interrupt is TRUE with the interval less than 10 ms (as set), the execution of the interrupt task will be ignored.
If no interrupt occurred in 50 ms (as set), the interrupt task is executed automatically.


\section*{Use the onboard I/Os as output limit switch}

\section*{Parameter configuration}

The user can configure the following output channel as limit switch.
- "LimitSwitch 0": Output channel 0
- "LimitSwitch 1": Output channel 1
- "LimitSwitch 2": Output channel 2
- "LimitSwitch 3": Output channel 3
- "LimitSwitch 4": Output channel 4
- "LimitSwitch 5": Output channel 5
- "LimitSwitch 6": Output channel 6
- "LimitSwitch 7": Output channel 7

More information about this topic can be found in the chapter 'Configuration the I/O channel'. \({ }_{\Downarrow}{ }^{\mu}\) Chapter 6.3.2.5 "Configure the onboard I/O channel" on page 1483

\section*{Function block}
\begin{tabular}{|c|c|}
\hline OBIOLimitSwitch & \multirow{7}{*}{} \\
\hline Enable BOOL & \\
\hline Channel BYTE & \\
\hline Signal BOOL & \\
\hline UpperLimitOn UDIVT & \\
\hline LowerLimitOn UDIVT & \\
\hline Counter OBIOBaseCounter & \\
\hline
\end{tabular}

If the counter value reaches the "LowerLimitOn" preset, it will write to the limit switch output based on the signal until the "UpperLimitOn" preset is reached.


\section*{Use the onboard I/Os as pulse-train output (PTO)}

100 kHz frequency (max. 2 pulse-train outputs using pulse-train output hardware channels)

\section*{Parameter configuration}

The user can configure the following output channels as pulse-train output (PTO).
- "PTO": Output channel 4
- "PTO": Output channel 5
- "PTO": Output channel 6
- "PTO": Output channel 7

If the user configures the output 4 or 6 as pulse-train output, the output 5 or 7 is automatically configured as pulse-train output.

The input "CwCCw" of the function block "OBIOPulseTrainOutput" determines the output 5 and 7 as "CounterClockWise" or "Direction" if it is set as pulse-train output.

More information about this topic can be found in the chapter 'Configuration the I/O channel'. Chapter 6.3.2.5 "Configure the onboard I/O channel" on page 1483

\section*{Function block}
\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|c|}{OBIOPulseTrainOutput} \\
\hline Enable BOOL & BOOL Busy \\
\hline Channel BYTE & BOOL Error \\
\hline Set BOOL & Error_ID ErrorID \\
\hline CounterValueSet UDIVT & UDINT CounterValue \\
\hline EnableLimit BOOL & \\
\hline LimitValueMax UDIVT & \\
\hline Start BOOL & \\
\hline Halt BOOL & \\
\hline CwCCw BOOL & \\
\hline Invert \(B O O L\) & \\
\hline Direction BOOL & \\
\hline EnableFinesteps BOOL & \\
\hline Frequency \(\angle R E A L\) & \\
\hline NumberOfPulses UDIVT & \\
\hline MaxFrequency UDIVT & \\
\hline CycleTime LREAL & \\
\hline
\end{tabular}

If "Set" bit is TRUE, the instruction moves the "CounterSetValue" to the "CounterValue".


If "EnableLimit" bit is TRUE, the accumulated value continues incrementing.
After "CounterValue" reaches the "LimitValueMax", the instruction writes 0 to the "CounterValue".


If the input "CwCCw" of the "OBIOPulseTrainOutput" is set to FALSE, the pulse-train output channel \(B\) is toggled based on the direction.


If the input "CwCCw" of the "OBIOPulseTrainOutput" is set to TRUE. The pulse-train output channel A will lead by \(90^{\circ}\) or pulse-train output channel \(B\) will lead by \(90^{\circ}\) depending on direction.


200 kHz frequency (max. 2 pulse-train output using pulse-train output hardware channels) and simple motion "OBIOMotionPTO" function block
Parameter configuration
Only the Standard and Pro processor modules can be used with pulse-train outputs. The Basic processor modules PM5012 do not have pulse-train outputs.
The available pulse-train output can be used as pulse-train output with pulse/ direction or pulse-train output with CW/CCW mode when the channels have been configured as pulse-train output.

The user can configure the following output channels as pulse-train output (PTO).
- "PTO": Output channel 4
- "PTO": Output channel 5
- "PTO": Output channel 6
- "PTO": Output channel 7

If the user configures the output 4 as pulse-train output, the output 5 is automatically configured as pulse-train output.
If the user configures the output 6 as pulse-train output, the output 7 is automatically configured as pulse-train output.
The input " CwCCw " of the function block "OBIOPulseTrainOutput" determines the output 5 and 7 as "CounterClockWise" or "Direction" if it is set as pulse-train output.

More information about this topic can be found in the chapter 'Configuration the I/O channel'. \# Chapter 6.3.2.5 "Configure the onboard I/O channel" on page 1483

\section*{Function block}
\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|c|}{OBIOMotionPTO} \\
\hline Enable BOOL & B00L Busy \\
\hline Channel BYTE & B00L Error \\
\hline OpenLoop BOOL & Error_10 Errorid \\
\hline CwCCw BOOL & BOOL InPosition \\
\hline ResetPos BOOL & BOOL InVelocity \\
\hline ExecutePos B00L & DWORD ActFrequency \\
\hline ExecuteVel BOOL & BOOL ActDirection \\
\hline Stop BOOL & \\
\hline RefPosition DIVT & \\
\hline RefFrequency DWORD & \\
\hline RefDirection \(B 00 \mathrm{~L}\) & \\
\hline RefAcceleration TIME & \\
\hline RefDeceleration TIME & \\
\hline MaxFrequency DWORD & \\
\hline MinFrequency DWORD & \\
\hline Cycle INT & \\
\hline ActPosition DIVT & \\
\hline p2p OBIOPoint2Point & \\
\hline
\end{tabular}

If the input "CwCCw" of the "OBIOMotionPTO" is set to FALSE, the pulse-train output channel B is toggled based on the direction.


If the input "CwCCw" of the "OBIOMotionPTO" is set to TRUE. The pulse-train output channel A will lead by \(90^{\circ}\) or pulse-train output channel \(B\) will lead by \(90^{\circ}\) depending on direction.


100 kHz .. 200 kHz frequency (max. 4 pulse-train output using puls-width modulation (PWM) hardware channels) and simple motion "OBIOMotionPWM" function block

\section*{Parameter configuration}

It is possible to have also up to 4 pulse-train output channels only with pulse/direction mode on the AC500-eCo V3 CPU by using the fast outputs O4...O7 configured as pulse-width modulation outputs and using a specific motion function block and standard outputs for direction channel.

Only the Standard and Pro processor modules can be used with pulsetrain output or pulse-width modulation outputs. The Basic processor modules PM5012 do not have pulse-train outputs.

The available software pulse-train outputs can be used as pulse-train output with pulse/direction or pulse-train output with CW/CCW mode when the channels have been configured as PWM outputs.

The user must configure the following output channels as pulse-width modulation outputs and use the "OBIOMotionPWM" function block.
- "PWM": Output channel 4
- "PWM": Output channel 5
- "PWM": Output channel 6
- "PWM": Output channel 7

If the user configures the output 4... 7 as pulse-width modulation using the "OBIOMotionPWM" function block, up to four software pulse-train output can be realized offering then only the pulse/direction mode.
The pulse output will always use the fast output channels \(\mathrm{O} 4 \ldots \mathrm{O}\) and the direction output of the function block can be assigned to any other output e.g. O0...O3 or also outputs from a S500 I/O module.

More information about this topic can be found in the chapter 'Configuration the I/O channel'. \(\Leftrightarrow\) Chapter 6.3.2.5 "Configure the onboard I/O channel" on page 1483

\section*{Function block}
\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|c|}{OBIOMotionPwm} \\
\hline Enable BOOL & BOOL Busy \\
\hline Channel BYTE & BOOL Error \\
\hline OpenLoop BOOL & Error_ID ErrorID \\
\hline ResetPos BOOL & BOOL InPosition \\
\hline ExecutePos BOOL & BOOL InVelocity \\
\hline ExecuteVel BOOL & DWORD ActFrequency \\
\hline Stop BOOL & BOOL ActDirection \\
\hline RefPosition DINT & \\
\hline RefFrequency DWORD & \\
\hline RefDirection BOOL & \\
\hline RefAcceleration TIME & \\
\hline RefDeceleration TIME & \\
\hline MaxFrequency DWORD & \\
\hline MinFrequency DWORD & \\
\hline Cycle INT & \\
\hline ActPosition DIIVT & \\
\hline p2p OBIOPoint2Point & \\
\hline
\end{tabular}


Use the onboard I/Os as output pulse-width modulation (PWM)

\section*{Parameter configuration}

The user can configure the following output channels as pulse-width modulation (PWM).
- "PWM 0": Output channel 0
- "PWM 1": Output channel 1
- "PWM 2": Output channel 2
- "PWM 3": Output channel 3
- "PWM 4": Output channel 4
- "PWM 5": Output channel 5
- "PWM 6": Output channel 6
- "PWM 7": Output channel 7

More information about this topic can be found in the chapter 'Configuration the I/O channel'. \(\Leftrightarrow\) Chapter 6.3.2.5 "Configure the onboard I/O channel" on page 1483

\section*{Function block}


The complete cycle of the "PWM" is based on the "OnTime" and "OffTime".


The duty cycle ratio of the "PWM" is based on the formula \(=T_{\text {on }} /\left(T_{\text {on }}+T_{\text {off }}\right)\).

\section*{75\% Duty Cycle}


\section*{50\% Duty Cycle}


\section*{25\% Duty Cycle}


\subsection*{6.8.2.15.4 Function block description}

Reference for function blocks, functions, structures etc. are available in the reference documentation and in the Library Manager.
«y Chapter 6.5.14 "Reference, function blocks" on page 4086
y Chapter 6.5.3.3 "View embedded documentation of all libraries" on page 3775

\subsection*{6.8.2.15.5 AC500-eCo V3 option board slots for processor modules PM50xx}

Depending on the processor module type, up to three option board slots are available on the CPU for different purpose like digital or analog I/O extension, serial interface or special module for specific functionality.
The option board slots are not limited to one type of option board. The option boards can be plugged and used in any slot. The only limitation is the number of slots available on the processor module. All types of option boards can be mixed on all slots.
Chapter 5.2.1.1.3 "Option boards" on page 218

Table 846: Option board slots
\begin{tabular}{|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{} & Basic CPU & \multicolumn{2}{|c|}{Standard CPU} & \multicolumn{2}{|c|}{Pro CPU} \\
\hline & PM5012 & PM5032 & PM5052 & PM5072 & PM5082 \\
\hline Number of option board slots &  &  &  &  &  \\
\hline Usable option boards on AC500-eCo V3 & \multicolumn{5}{|l|}{} \\
\hline TA5130-KNXPB & - & - & - & \(\mathrm{X}^{1}\) ) & X \({ }^{1}\) ) \\
\hline TA5130KNXPBW & - & - & - & \(\left.\mathrm{X}^{1}\right)^{2}\) ) & - \\
\hline TA5131-RTC & X & - & - & - & - \\
\hline TA5101-4DI & X & X & X & X & X \\
\hline TA5101-4DIW & - & - & - & \(\mathrm{X}^{2}\) ) & - \\
\hline TA5105-4DOT & X & X & X & X & X \\
\hline TA5105-4DOTW & - & - & - & \(\mathrm{X}^{2}\) ) & - \\
\hline \begin{tabular}{l}
TA5110-2DI2DO \\
T
\end{tabular} & X & X & X & X & X \\
\hline TA5110-2DI2DO W & - & - & - & \(\mathrm{X}^{2}\) ) & - \\
\hline TA5120-2AI-UI & X & X & X & X & X \\
\hline TA5120-2AI-UIW & - & - & - & \(\mathrm{X}^{2}\) ) & - \\
\hline TA5123-2AI-RTD & X & X & X & X & X \\
\hline TA5123-2AI-RTW & - & - & - & \(\mathrm{X}^{2}\) ) & - \\
\hline TA5126-2AO-UI & X & X & X & X & X \\
\hline TA5126-2AOUIW & - & - & - & \(\mathrm{X}^{2}\) ) & - \\
\hline TA5141-RS2321 & X & X & X & X & X \\
\hline \[
\begin{aligned}
& \text { TA5141- } \\
& \text { RS232IW }
\end{aligned}
\] & - & - & - & \(\mathrm{X}^{2}\) ) & - \\
\hline TA5142-RS485I & X & X & X & X & X \\
\hline TA5142RS485IW & - & - & - & \(\mathrm{X}^{2}\) ) & - \\
\hline TA5142-RS485 & X & X & X & X & X \\
\hline TA5142-RS485W & - & - & - & \(\mathrm{X}^{2}\) ) & - \\
\hline \multicolumn{6}{|l|}{\({ }^{1}\) ) Can be used only once per CPU} \\
\hline
\end{tabular}

\section*{Option board for COMx serial communication}

One of these option boards is always needed for serial communication like Modbus RTU.
をy Information about selection and configuration
\& More information on technical data and ordering information of the option boards

\section*{Option board for digital I/O extension}
\({ }_{\xi}{ }^{\Downarrow}\) Information about selection and configuration
« More information on technical data and ordering information of the option boards

\section*{Option board for analog I/O extension}
\({ }^{\aleph}\) Information about selection and configuration
\& More information on technical data and ordering information of the option boards

\section*{Option board for specific function}

The TA5130-KNXPB can only be used on AC500-eCo V3 processor modules Pro PM5072-T-ETH(W) and PM5082-T-2ETH.

The TA5131-RTC can only be used on AC500-eCo V3 processor modules Basic PM5012-x-ETH.

These two option boards can only be used once on one slot at a time!
4) More information on technical data and ordering information of the option boards

\subsection*{6.8.2.16 Access to digital I/Os transferred as bit strings WORD/DWORD}

\subsection*{6.8.2.16.1 General}

The AC500 V3 PLCs runs in little-endian mode, means uses little-endian byte order.
\begin{tabular}{|l|l|l|l|l|}
\hline Address & Addr & Addr+1 & Addr+2 & Addr+3 \\
\hline & \(16 \# x x x x \times 000\) & \(16 \# x x x x \times 001\) & \(16 \# x x x x \times 002\) & \(16 \# x x x x \times 003\) \\
\hline BYTE & \(\mathbf{3}\) & \(\mathbf{2}\) & \(\mathbf{1}\) & \(\mathbf{0}\) \\
\hline BOOL & 7 & 7 & 7 & 7 \\
& 0 & 0 & 0 & 0 \\
\hline \multirow{2}{*}{ WORD } & \(\mathbf{1}\) & \(\mathbf{0}\) & \\
\cline { 2 - 5 } & 15 & 7 & 15 & 7 \\
& 8 & 0 & 8 & 0 \\
\hline
\end{tabular}
\begin{tabular}{|l|l|l|l|l|}
\hline Address & Addr & Addr+1 & Addr+2 & Addr+3 \\
\hline DWORD & \(\mathbf{0}\) & \multicolumn{4}{|l|}{} \\
\cline { 2 - 5 } & 7 & 15 & 23 & 31 \\
& & 0 & 8 & 16
\end{tabular}

The layout of the I/Os depends on the byte order of the I/O bus, fieldbus or protocol.

\subsection*{6.8.2.16.2 Bit strings transferred to PLC in little-endian byte order (Intel)}

\section*{Access to bit strings transferred to PLC in little-endian byte order}

If the fieldbus/protocol transfers the data in little-endian byte order, the access to the bits in a bit string will be as followed:
\begin{tabular}{|c|c|c|c|c|}
\hline Address & Addr & Addr+1 & Addr+2 & Addr+3 \\
\hline & 16\#xxxx x000 & 16\#xxxx x001 & 16\#xxxx x002 & 16\#xxxx x003 \\
\hline BYTE & \%IB3 & \%IB2 & \%IB1 & \%IB0 \\
\hline BOOL & \[
\begin{aligned}
& 7 \\
& 0
\end{aligned}
\] & \[
\begin{aligned}
& 7 \\
& 0
\end{aligned}
\] & \[
\begin{aligned}
& 7 \\
& 0
\end{aligned}
\] & \[
\begin{aligned}
& 7 \\
& 0
\end{aligned}
\] \\
\hline & \[
\begin{aligned}
& \hline \text { \%IX3.7 } \\
& \text { \%IX3.0 }
\end{aligned}
\] & \[
\begin{gathered}
\text { \%IX2.7 } \\
\text { \%IX2.0 }
\end{gathered}
\] & \[
\begin{aligned}
& \text { \%IX1.7 } \\
& \text { \%IX1.0 }
\end{aligned}
\] & \[
\begin{aligned}
& \text { \%IX0.7 } \\
& \text { \%IX0.0 }
\end{aligned}
\] \\
\hline WORD & \%IW1 & & \%IW0 & \\
\hline & \[
\begin{aligned}
& \hline 15 \\
& 8
\end{aligned}
\] & \[
\begin{aligned}
& 7 \\
& 0
\end{aligned}
\] & \[
\begin{aligned}
& 15 \\
& 8
\end{aligned}
\] & \[
\begin{aligned}
& 7 \\
& 0
\end{aligned}
\] \\
\hline DWORD & \%ID0 & & & \\
\hline & \({ }^{7} 0\) & \[
\begin{array}{r}
15 \\
8
\end{array}
\] & \[
\begin{aligned}
& 23 \\
& 16
\end{aligned}
\] & \[
\begin{aligned}
& 31 \\
& 24
\end{aligned}
\] \\
\hline
\end{tabular}

Example
\[
\begin{array}{lll}
\text { \%IX1.0 }:=\text { TRUE } & & \\
\text { \%IB0 }:=1 & :=16 \# 01 & \text { (Bit } 0=\text { TRUE) } \\
\text { \%IW1 }:=1 & :=16 \# 0001 & \text { (Bit } 0=\text { TRUE) } \\
\text { \%ID0 }:=1 & :=16 \# 00000001 & \text { (Bit } 0=\text { TRUE) } \\
\text { \%IX3.0 }:=\text { TRUE } & :=16 \# 01 & \\
\text { \%IB3 }:=1 & :=16 \# 0100 & \text { (Bit } 0=\text { TRUE) } \\
\text { \%IW1 }:=256 & :=16 \# 01000000 & \text { (Bit } 8=\text { TRUE) } \\
\text { \%ID0 }:=16777216 & \text { (Bit TRUE) }
\end{array}
\]

The AC500 V3 PLCs support the following busses/protocols with little-endian byte order:
- EtherCAT
- CANopen
- EtherNet/IP

Representation of bitfields in I/O configuration transferred in little-endian byte order
The values of I/Os can be monitored and controlled in the I/O configuration of a device. If the bit strings are transferred in little-endian byte order the bit channels will be displayed in the following way:


In general, it is not recommended to use the "Address", e.g. 으QX5. 2 directly in the IEC application. Use always symbolic variables to avoid access to wrong I/O channels.

\subsection*{6.8.2.16.3 Bit strings transferred to PLC in big-endian byte order (Motorola)}

Access to bit strings transferred to PLC in big-endian byte order
If the fieldbus/protocol transfers the data in big-endian byte order, the access to the bits in a bit string will be as followed:
\begin{tabular}{|c|c|c|c|c|}
\hline Address & Addr & Addr+1 & Addr+2 & Addr+3 \\
\hline & 16\#xxxx x000 & 16\#xxxx x001 & 16\#xxxx x002 & 16\#xxxx x003 \\
\hline BYTE & \%IB0 & \%IB1 & \%IB2 & \%IB3 \\
\hline \multirow[t]{2}{*}{BOOL} & \[
\begin{aligned}
& 7 \\
& 0
\end{aligned}
\] & \[
\begin{aligned}
& 7 \\
& 0
\end{aligned}
\] & \[
\begin{aligned}
& 7 \\
& 0
\end{aligned}
\] & \[
\begin{array}{|l}
7 \\
0
\end{array}
\] \\
\hline & \[
\begin{array}{|l|}
\hline \% I X 0.7 \\
\text { \%IX0.0 }
\end{array}
\] & \[
\begin{aligned}
& \hline \text { \%IX1.7 } \\
& \text { \%IX1.0 }
\end{aligned}
\] & \[
\begin{aligned}
& \text { \%IX2.7 } \\
& \text { \%IX2.0 }
\end{aligned}
\] & \[
\begin{aligned}
& \hline \text { \%IX3.7 } \\
& \text { \%IX3.0 }
\end{aligned}
\] \\
\hline \multirow[t]{2}{*}{WORD} & \multicolumn{2}{|l|}{\%IW0} & \multicolumn{2}{|l|}{\%IW1} \\
\hline & \[
\begin{aligned}
& \hline 15 \\
& 8
\end{aligned}
\] & \[
\begin{aligned}
& 7 \\
& 0
\end{aligned}
\] & \[
\begin{aligned}
& 15 \\
& \hline 8
\end{aligned}
\] & \[
\begin{aligned}
& 7 \\
& 0
\end{aligned}
\] \\
\hline \multirow[t]{2}{*}{DWORD} & \multicolumn{4}{|l|}{\%ID0} \\
\hline & \[
\begin{aligned}
& \hline 31 \\
& 24
\end{aligned}
\] & \(23 \quad 16\) & \[
\begin{aligned}
& 15 \\
& 8
\end{aligned}
\] & \[
\begin{aligned}
& 7 \\
& 0
\end{aligned}
\] \\
\hline
\end{tabular}

\section*{Example}
\%IX0.0 := TRUE
\%IB0 := 1
\[
:=16 \# 01
\]
(Bit \(0=\) TRUE)
\%IW1 := 256
\[
:=16 \# 0100
\]
\[
\text { (Bit } 8 \text { = TRUE) }
\]
\%ID0 := 16777216
:= 16\#01000000
(Bit \(24=\) TRUE)
\%IX3.0 := TRUE
\%IB3 := 1
:= 16\#01
(Bit \(0=\) TRUE)
\%IW1 := 1
:= 16\#0001
(Bit \(0=\) TRUE)
\%ID0 := 1
:= 16\#00000001
(Bit \(0=\) TRUE)

The AC500 V3 PLCs support the following busses/protocols with little-endian byte order:
- Local I/O bus
- Modbus TCP
- PROFINET
- PROFIBUS

\section*{Representation of bitfields in I/O configuration transferred in big-endian byte order}

The values of I/Os can be monitored and controlled in the I/O configuration of a device. If the bit strings are transferred in big-endian byte order the bytes inside the word will be swapped.

Up to Automation Builder version 2.2 the bit channels will be displayed in the following way:
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline Variable & Mapping & Channel & Address & Type & Default Value & Current Value \\
\hline - SafeDigI_W & * & Digital inputs DIO - DI15 & \%IW7 & UINT & 0 & 2048 \\
\hline * SafeDigI_0 & * & Digital input DIO & \%IX 14.0 & BOOL & & FALSE \\
\hline * SafeDigI_1 & * & Digital input DI1 & \%IX14.1 & BOOL & & FALSE \\
\hline * SafeDigi_2 & * & Digital input DI2 & \%IX14.2 & BOOL & & FALSE \\
\hline \({ }^{4}\) SafeDigI_3 & * & Digital input DI3 & \%IX14.3 & BOOL & & FALSE \\
\hline * SafeDigI_4 & * & Digital input DI4 & \%IX14.4 & BOOL & & FALSE \\
\hline * SafeDigI_5 & * & Digital input DI5 & \%IX14.5 & BOOL & & FALSE \\
\hline 4 SafeDigI_6 & * & Digital input DI6 & \%IX14.6 & BOOL & & FALSE \\
\hline * SafeDigI_7 & * & Digital input DI7 & \%IX 14.7 & BOOL & & FALSE \\
\hline 4 SafeDigI_8 & * & Digital input DI8 & \%IX 15.0 & BOOL & & FALSE \\
\hline * SafeDigI_9 & * & Digital input DI9 & \%IX15.1 & BOOL & & FALSE \\
\hline 4. SafeDigI_10 & * & Digital input DI10 & \%IX 15.2 & BOOL & & FALSE \\
\hline 4. SafeDigI_11 & * & Digital input DI11 & \%IX15.3 & BOOL & & TRUE \\
\hline * SafeDigI_12 & * & Digital input DI12 & \%IX 15.4 & BOOL & & FALSE \\
\hline 4 SafeDigI_13 & * & Digital input DI13 & \%IX15.5 & BOOL & & FALSE \\
\hline *) SafeDigI_14 & * & Digital input DI14 & \%IX15.6 & BOOL & & FALSE \\
\hline * SafeDigI_15 & * & Digital input DI15 & \%IX15.7 & BOOL & & FALSE \\
\hline
\end{tabular}

Input channel DI3 was set in bit string SafeDigi_W. Since the inputs are transferred in bigendian byte order, the input channel SafeDigi_11 or SafeDigI_W. 11 is set. User has to swap the bytes in the IEC application.

As of Automation Builder version 2.3.0 the bit channels will be displayed in the following way:
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline Variable & Mapping & Channel & Address & Type & Default Value & Current Value \\
\hline - SafeDigI_W & \({ }^{*}\) & Digital inputs DI0 - DI15 & \%IW7 & UINT & 0 & 2048 \\
\hline * SafeDigI_0 & \% & Digital input DIO & \%IX 15.0 & BOOL & & FALSE \\
\hline * SafeDigI_1 & \% & Digital input DI1 & \%IX15.1 & B0OL & & FALSE \\
\hline * SafeDigI_2 & \% & Digital input DI2 & \%IX 15.2 & BOOL & & FALSE \\
\hline * SafeDigi_3 & * & Digital input DI3 & \%IX15.3 & BOOL & & TRUE \\
\hline \({ }^{*}\) SafeDigI_4 & \% & Digital input DI4 & \%IX15.4 & BOOL & & FALSE \\
\hline 4 SafeDigI_5 & - & Digital input DI5 & \%IX 15.5 & BOOL & & FALSE \\
\hline * SafeDigI_6 & 4 & Digital input DI6 & \%IX 15.6 & BOOL & & FALSE \\
\hline 4 SafeDigl_7 & 4 & Digital input DI7 & \%IX 15.7 & BOOL & & FALSE \\
\hline 4 SafeDigI_8 & \% & Digital input DI8 & \%IX 14.0 & B00L & & FALSE \\
\hline * SafeDigI_9 & \% & Digital input DI9 & \%IX 14.1 & BOOL & & FALSE \\
\hline 4 SafeDigI_10 & - & Digital input DI10 & \%IX 14.2 & BOOL & & FALSE \\
\hline 4) SafeDigI_11 & \% & Digital input DI11 & \%DX14.3 & BOOL & & FALSE \\
\hline 4 SafeDigI_12 & 4 & Digital input DI12 & \%IX 14.4 & BOOL & & FALSE \\
\hline 4 SafeDigI_13 & \% & Digital input DI13 & \%DX14.5 & BOOL & & FALSE \\
\hline - SafeDigI_14 & 4 & Digital input DI14 & \%IX 14.6 & BOOL & & FALSE \\
\hline * SafeDigI_15 & * & Digital input DI15 & \%DX14.7 & B0OL & & FALSE \\
\hline
\end{tabular}

The representation of a bitfield is changed by swapping the bytes correspondingly to the type used in the IO configuration. Now the correct SafeDigI_3 or SafeDigI_W. 3 is set in case of DI3 is set.

In general, it is not recommended to use the "Address", e.g. 으X15. 3 directly in the IEC application. Use always symbolic variables to avoid access to wrong I/O channels.

\subsection*{6.8.3 System technology of the AC500 communication modules}

\subsection*{6.8.3.1 CANopen communication modules}

\subsection*{6.8.3.1.1 Triggering of event tasks with CAN-IDs}

For CM598-CAN module the execution of a PLC application task can be triggered automatically by a certain event, i.e. by incoming CAN 2.0 A or CAN 2.0 B frames. For this, the PLC application task is to be configured as external event task.

\section*{Prerequisites}
- PLC firmware version 3.2.5 and Automation Builder as of version 2.2.5.
- Only one PLC application task can be assigned to a communication module.
- Triggering of event tasks is only supported for the communication module CM598-CAN.


Every incoming CAN frame on a CM598-CAN module processes an event in the AC500 PLC. If the parameter "Trigger PLC Task" is set to TRUE, the CAN protocol task checks via the receive buffer configuration and the corresponding CAN-ID of the CAN frame whether a CAN frame is to be executed or not. Only those CAN-IDs that are configured in the protocol configuration will be processed. All other CAN frames will be rejected. If a CAN frame is to be processed, the CAN frame data is copied to the receive buffer and an event on the IEC event task is triggered.

The IEC event task will be executed for one cycle.
The IEC event task will be triggered continuously until all associated receive buffers have been emptied. Hence, ensure that the buffers are emptied by the task, otherwise the task will run into a loop.

Within the task the function block Cm598CanMsgRecEvt must be used to read the CAN frames from the receive buffers. The function block Cm598CanMsgRec is not suitable as it requires several task cycles for execution.

CAN frame pro- The following figure shows the sequence CAN frames processing when the triggering of event cessing task is used.

- Only one external event task can be assigned to a CM598-CAN.
- There is only one common event for an external event task and all selected CAN-IDs. It must be evaluated which CAN-IDs have been received.
- It is possible that CAN frames are lost when necessary system resources are in use or when the CAN frames could not be processed in time due to high system load. So, the PLC application must monitor the task which consumes the events of the CAN protocol with a watchdog mechanism or something similar.
- Received CAN frames of the same CAN-ID are internally stored in FIFO buffers. Reading and writing of the FIFO buffers is not possible at the same time.
- Within an external event task the function block Cm598CanMsgRecEvt must be used to read the received CAN frames. The function block Cm598CanMsgRec is not suitable since its execution needs more than one task cycle.
- The CAN-IDs that are enabled to trigger an external event task must be read by the associated task. Otherwise the task is triggered again and again, and the CPU load will be high.

Event task con- Add the external event task that should be executed to the task configuration of the PLC figuration application:
1. Right-click on "Task Configuration". Enter a name for the task and click "Add object".
2. Right-click on the new task and append a "Program Call". This contains the program code that is executed by the task.
3. Double-click on the task and setup the task parameters.


A parameter description is given in the chapter 'Tab 'configuration" \({ }^{〔}\) " Chapter 6.4.1.21.2.30.2 "Tab 'Configuration'" on page 2538. Deviations are described in the following:
\begin{tabular}{|l|l|l|l|}
\hline Parameter & Default & Value & Description \\
\hline Priority & 16 & \begin{tabular}{l}
\(0 . .16\) \\
Value '0' indicates the \\
highest priority
\end{tabular} & Priority of the task \\
\hline Type & n.a. & External & \begin{tabular}{l} 
Specifies the task \\
type.
\end{tabular} \\
\hline External Event & n.a. & \begin{tabular}{l} 
CouplerEvent<slot \\
index of the \\
CM>_CAN
\end{tabular} & \begin{tabular}{l} 
Specifies the event \\
that triggers execution \\
of the task.
\end{tabular} \\
\hline Interval & n.a. & Cycle time & Not used \\
\hline
\end{tabular}

Configuration of a CM598-CAN module is desribed in the configuration chapter \({ }^{4} \Rightarrow\) Chapter 6.3.2.11.1 "CANopen" on page 1521.

\subsection*{6.8.3.2 CM582-DP/CM592-DP PROFIBUS DP communication modules}
* \({ }^{4}\) Chapter 6.3.2.11.4.2 "Parameterization of the CM592-DP/CM582-DP communication modules" on page \(1553 \Leftrightarrow\) Chapter 6.3.2.11.4.3.1 "Configuration of a PROFIBUS DP master" on page \(1554 \Leftrightarrow\) Chapter 6.3.2.11.4.3.2 "Configuration of a PROFIBUS DP slave" on page 1557

\subsection*{6.8.3.3 CM5640-2ETH Ethernet communication module}

²) Chapter 6.3.2.11.5.1 "CM5640-2ETH - Ethernet communication modul" on page 1562

\subsection*{6.8.4 System technology of the communication interface modules}

\subsection*{6.8.4.1 Modbus communication interface module}

\subsection*{6.8.4.1.1 Overview}

The Modbus TCP communication interface module CI52x-MODTCP is used as decentralized I/O module in Modbus TCP networks. The network connection is performed via 2 RJ45 connectors which are integrated in the terminal unit.

\section*{Application examples}

In AC500 V3 projects in which several Modbus connections are needed at the same time, the number of Modbus sockets can be increased. How to increase the socket number with an additional run time license is described in the application note AC500 Line Mode with AC500 V3 CPU's.

A CI52x cluster can be used for the communication between the CPU and CI52x-MODTCP modules. An example on how to create a \(\mathrm{CI} 52 x\) cluster configuration using the bulk data manager tool, is described in the application example CI52x-MODTCP cluster.

A summary of the function codes that are supported for Modbus TCP communication with a PLC is given in the application example Modbus TCP and ModTcpMast2.

A description on how to establish a Modbus TCP communication with fixed and dynamic register mapping in an AC500 PLC is given in the application example Modbus TCP and register mapping.

To operate fast counter on Cl 52 x devices in combination with a V3 PLC correctly via Modbus TCP, a corresponding sample project is available. Setup and use are described in an application example.

CI521-MODTCP I/O channels properties:
- 4 analog inputs (1.0...1.3)
- 2 analog outputs (1.5...1.6)
- 8 digital inputs 24 V DC in 1 group (2.0...2.7)
- 8 digital outputs 24 V DC in 1 group (3.0...3.7)

\section*{Functionality}
\begin{tabular}{|l|l|}
\hline Parameter & Value \\
\hline Interface & Ethernet \\
\hline Protocol & Modbus TCP \\
\hline Power supply & from the process supply voltage UP \\
\hline \begin{tabular}{l} 
Supply of the electronic circuitry of the \\
I/O expansion modules attached
\end{tabular} & Through the I/O bus interface (I/O bus) \\
\hline Rotary switches & For setting the last BYTE of the IP (00h to FFh) \\
\hline Analog inputs & 4 (configurable via software) \\
\hline Analog outputs & 2 (configurable via software) \\
\hline Digital inputs & 8 (24 V DC; delay time configurable via software) \\
\hline Digital outputs & 8 (24 V DC, 0.5 A max.) \\
\hline LED displays & \begin{tabular}{l} 
For system displays, signal states, errors and \\
power supply
\end{tabular} \\
\hline External supply voltage & \begin{tabular}{l} 
Via terminals ZP, UP and UP3 (process supply \\
voltage 24 V DC)
\end{tabular} \\
\hline
\end{tabular}

\section*{CI522-MODTCP I/O channels properties:}
- 8 digital configurable inputs/outputs in 1 group (1.0...1.7)
- 8 digital inputs 24 V DC in 1 group (2.0...2.7)
- 8 digital outputs 24 V DC in 1 group (3.0...3.7)

\section*{Functionality}
\begin{tabular}{|l|l|}
\hline Parameter & Value \\
\hline Interface & Ethernet \\
\hline Protocol & Modbus TCP \\
\hline Power supply & from the process supply voltage UP \\
\hline \begin{tabular}{l} 
Supply of the electronic circuitry of the \\
I/O expansion modules attached
\end{tabular} & Through the I/O bus interface (I/O bus) \\
\hline Rotary switches & For setting the last BYTE of the IP (00h to FFh) \\
\hline Configurable digital inputs/outputs & 8 (configurable via software) \\
\hline Digital inputs & 8 (24 V DC; delay time configurable via software) \\
\hline Digital outputs & 8 (24 V DC, 0.5 A max.) \\
\hline LED displays & \begin{tabular}{l} 
For system displays, signal states, errors and \\
power supply
\end{tabular} \\
\hline External supply voltage & \begin{tabular}{l} 
Via terminals ZP, UP and UP3 (process supply \\
voltage 24 V DC)
\end{tabular} \\
\hline
\end{tabular}

The inputs/outputs are galvanically isolated from the Ethernet network. There is no potential separation between the channels.

The configuration of the inputs/outputs is performed by software.
For usage in enhanced ambient conditions (e.g. wider temperature and humidity range), a special XC version of the device is available.

\subsection*{6.8.4.1.2 Modbus TCP registers}

\section*{Register layout for CI52x-MODTCP}

The registers can be divided in 4 sections:
- Information data section 0x0000 to 0x0D50 (for acyclic use)
- I/O data and diagnosis section 0x0FFA to 0x2B00 (for cyclic use)
- Parameter data section \(0 \times 3000\) to 0x3B00 (for acyclic use)
- Special functionality section \(0 \times 5 \mathrm{~A} 00\) to \(0 \times 6 \mathrm{~A} 00\) (for acyclic use)

\section*{Information data section (acyclic data)}

\section*{General}

The information data section can be used to read out common and module specific information.
This section is read only.
\begin{tabular}{|r|l|l|l|}
\hline \begin{tabular}{r} 
Register \\
(hex)
\end{tabular} & Description & \begin{tabular}{l} 
Readable by \\
Modbus \\
function code
\end{tabular} & \begin{tabular}{l} 
Writeable by \\
Modbus \\
function code
\end{tabular} \\
\hline 0 & Device and FW information CI & 3 & x \\
\hline 50 & Production data CI & 3 & x \\
\hline 100 & Device and FW information 1. EXP & 3 & x \\
\hline 125 & \begin{tabular}{l} 
Device and FW information 1. Hot swap \\
terminal unit
\end{tabular} & \(3^{*}\) ) & x \\
\hline 150 & Production data 1. EXP & 3 & x \\
\hline 175 & Production data 1. Hot swap terminal unit & \(\left.3^{*}\right)\) & x \\
\hline\(\ldots\) & \(\ldots\) & x \\
\hline A00 & Device and FW information 10. EXP & 3 & x \\
\hline A25 & \begin{tabular}{l} 
Device and FW information 10. Hot swap \\
terminal unit
\end{tabular} & \(\left.3^{*}\right)\) & x \\
\hline A50 & Production data 10. EXP & 3 & x \\
\hline A75 & \begin{tabular}{l} 
Production data 10. Hot swap terminal \\
unit
\end{tabular} & 3 *) & x \\
\hline D00 & Common device information & 3 & \\
\hline
\end{tabular}
*) supported from CI52x firmware version V3.2.0 (device index F0)
This section can be divided again in two sections:
- The module specific section (containing information for each module CI52x-MODTCP and expansion modules and hot swap terminal units)
\({ }^{4}\) Chapter 6.8.4.1.2.2.2 "Module specific information registers" on page 4570
- The common device information block

② Chapter 6.8.4.1.2.2.3 "Common device information registers" on page 4572

\section*{Module specific information registers}

For each module (CI52x device, expansion modules and hot swap terminal units) the following data can be read out:
- Device and FW information

This section consists of 20 WORDs per module and contains information on each module using the following structure:
\begin{tabular}{|l|l|l|}
\hline Data & DATA TYPE & Description \\
\hline Module ID & WORD & The module ID of the requested module \\
\hline Module name & ARRAY [1..10] OF BYTE & The module name of the requested module \\
\hline Version \(1^{\text {st }}\) processor & ARRAY [1..4] OF BYTE & \begin{tabular}{l} 
The version of the \(1^{\text {st }}\) processor of the \\
requested module
\end{tabular} \\
\hline Version \(2^{\text {nd }}\) processor & ARRAY [1..4] OF BYTE & \begin{tabular}{l} 
The version of the \(2^{\text {nd }}\) processor of the \\
requested module
\end{tabular} \\
\hline Version \(3^{\text {rd }}\) processor & ARRAY [1..4] OF BYTE & \begin{tabular}{l} 
The version of the \(3^{\text {rd }}\) processor of the \\
requested module
\end{tabular} \\
\hline Version 4 \({ }^{\text {th }}\) processor & ARRAY [1..4] OF BYTE & \begin{tabular}{l} 
The version of the \(4^{\text {th }}\) processor of the \\
requested module
\end{tabular} \\
\hline Hardware version \(\left.{ }^{1}\right)\) & ARRAY [1..4] OF BYTE & The hardware version of the 4 processors \\
\hline Reserved & \begin{tabular}{l} 
ARRAY [1..8] OF BYTE \\
ARRAY [1..4] OF BYTE \\
2 Reserved
\end{tabular} & \begin{tabular}{l} 
Red
\end{tabular} \\
\hline Number input data & WORD & \begin{tabular}{l} 
Number of input data of the requested \\
module in BYTES
\end{tabular} \\
\hline Number output data & WORD & \begin{tabular}{l} 
Number of output data of the requested \\
module in BYTES
\end{tabular} \\
\hline
\end{tabular}
\({ }^{1}\) ) supported from CI52x firmware version V3.2.0 (device index F0)
\({ }^{2}\) ) from CI52x firmware version V3.2.0 (device index F0) "Reserved" is ARRAY [1..4] OF BYTE
- Production / Traceability data:

This section consists of 25 WORDs per module and contains the traceability data for each module using following structure:
- Article number: Byte \(01 . .15\)
- Index: Byte 16.. 17
- Name: Byte 18.. 29
- Production date: Byte 30.. 33
- Key number: Byte 34.. 38
- Site: Byte \(39 . .40\)
- Year: Byte 41.. 42
- Serial number: Byte \(41 . .50\) (The serial number implies the year)
- Production / Traceability data from \(\mathrm{CI} 5 \times 2\) firmware version V3.2.0 (device index F0): This section consists of 26 WORDs per module and contains the traceability data for each module using following structure:
- Article number: Byte 01.. 15
- Index: Byte 16.. 17
- Name: Byte 18.. 31
- Production date: Byte 32.. 35
- Key number: Byte \(36 . .40\)
- Site: Byte \(41 . .42\)
- Year: Byte 43.. 44
- Serial number: Byte \(42 . .52\) (The serial number implies the year)

\section*{Common device information registers}

Common device This section consists of 80 WORDs (90 WORDs from CI52x firmware version V3.2.0 (device
information block index F0)) and contains cluster wide information (CI52x device and connected expansion modules using the following structure:
\(\left.\begin{array}{|l|l|l|}\hline \text { Data } & \text { DATA TYPE } & \text { Description } \\ \hline \text { Device state } & \text { BYTE } & \begin{array}{l}\text { The actual state of the device: } \\ \text { 0: STATE_PREOP (device booting) } \\ \text { 1: STATE_OPERATION (device in operational, }\end{array} \\ \text { no bus supervision active) } \\ \text { 2: STATE_ERROR (device detected a bus } \\ \text { error, bus supervision active) } \\ \text { 3: STATE_IP_ERROR (the device has a IP } \\ \text { address error) } \\ \text { 4: STATE_CYCLIC_OPERATION (device in } \\ \text { operational, bus supervision active) }\end{array}\right\}\)
\begin{tabular}{|c|c|c|}
\hline Data & DATA TYPE & Description \\
\hline Input address onboard I/O & WORD & Modbus TCP register address for inputs of the onboard I/Os \\
\hline Input address \(1^{\text {st }}\) expansion & WORD & Modbus TCP register address for inputs of the \(1{ }^{\text {st }}\) expansion module \\
\hline Input address \(2^{\text {nd }}\) expansion & WORD & Modbus TCP register address for inputs of the \(2^{\text {nd }}\) expansion module \\
\hline \multicolumn{3}{|l|}{...} \\
\hline Input address \(10^{\text {th }}\) expansion & WORD & Modbus TCP register address for inputs of the \(10^{\text {th }}\) expansion module \\
\hline Output address onboard I/O & WORD & Modbus TCP register address for outputs of the onboard I/Os \\
\hline Output address \(1^{\text {st }}\) expansion & WORD & Modbus TCP register address for outputs of the \(1^{\text {st }}\) expansion module \\
\hline Output address \(2^{\text {nd }}\) expansion & WORD & Modbus TCP register address for outputs of the \(2^{\text {nd }}\) expansion module \\
\hline \multicolumn{3}{|l|}{...} \\
\hline Output address \(10^{\text {th }}\) expansion & WORD & Modbus TCP register address for outputs of the \(10^{\text {th }}\) expansion module \\
\hline Module ID \(1^{\text {st }}\) hot swap terminal unit *) & WORD & Module ID of the \(1^{\text {st }}\) connected hot swap terminal unit *) \\
\hline Module ID \(2^{\text {nd }}\) hot swap terminal unit *) & WORD & Module ID of the \(2^{\text {nd }}\) connected hot swap terminal unit *) \\
\hline \multicolumn{3}{|l|}{\(\ldots\)} \\
\hline Module ID \(10^{\text {th }}\) hot swap terminal unit *) & WORD & Module ID of the \(10^{\text {th }}\) connected hot swap terminal unit *) \\
\hline
\end{tabular}
*) supported from CI52x firmware version V3.2.0 (device index F0)

I/O / Process data and diagnosis section (cyclic data)

\section*{General}

Table 847: The cyclic data section for CI52x-MODTCP
\begin{tabular}{|r|l|l|l|}
\hline \begin{tabular}{r} 
Register \\
(hex)
\end{tabular} & Description & \begin{tabular}{l} 
Readable by \\
Modbus function \\
code
\end{tabular} & \begin{tabular}{l} 
Writeable by \\
Modbus function \\
code
\end{tabular} \\
\hline FCE \(^{*}\) ) & Module state & \(3,4,23\) & x \\
\hline FFA & Diagnosis & \(3,4,23\) & x \\
\hline 1000 & Inputs CI & \(3,4,23\) & x \\
\hline 1100 & Inputs 1.EXP & \(3,4,23\) & x \\
\hline\(\ldots\) & \(\ldots\) & & x \\
\hline 1 A00 & Inputs 10.EXP & \(3,4,23\) & x \\
\hline 2000 & Outputs CI & 3,23 & \(6,16,23\) \\
\hline 2100 & Outputs 1.EXP & 3,23 & \(6,16,23\) \\
\hline\(\ldots\) & \(\ldots\) & 3,23 & \(6,16,23\) \\
\hline 2 F 00 & Outputs 10.EXP & 3,23 & \(6,16,23\) \\
\hline \(2 \mathrm{B00}\) & Dummy output & & \\
\hline
\end{tabular}
*) supported from CI52x firmware version V3.2.0 (device index F0)
This section can be divided again in three sections:
- Module state (containing the state of connected expansion modules and hot swap terminal units)
(4) Chapter 6.8.4.1.2.3.2 "Module state" on page 4574
- Diagnosis data (containing diagnosis data in AC500 specific format)

² Chapter 6.8.4.1.2.3.3 "Diagnosis data" on page 4576
- Process data (containing I/O data)
\& Chapter 6.8.4.1.2.3.4 "I/O data" on page 4576

\section*{Module state}

The module state section consists of 44 WORDs and contains the module state of connected expansion modules and hot swap terminal units using the following structure:
\begin{tabular}{|l|l|l|}
\hline Data & DATA TYPE & Description \\
\hline Module ID & WORD & Module ID of the CI52x \\
\hline Expected module ID & WORD & Expected (configured) module ID of the CI52x \\
\hline Module state & BYTE & \begin{tabular}{l} 
The current module state of the CI52x: \\
0: NO_MOD (no module detected) \\
1: MOD_INIT (module detected, module is in \\
initialization phase) \\
2: MOD_RUN (module detected and running or in \\
failsafe state, input data are valid) \\
3: WRONG_MOD (wrong module detected, module ID \\
doesn't match expected module ID) \\
4: MOD_REMOVED (module removed or defective on \\
hot swap terminal unit, no communication to module \\
possible) \\
5: MOD_ERROR (module defective on hot swap \\
terminal unit, no communication to module possible) \\
6: MOD_LOST (lost communication to module on not \\
hot swap capable terminal unit) \\
7: UNKNOWN (module detected but not configured)
\end{tabular} \\
& & \\
& & \\
& & \begin{tabular}{l} 
Diagnosis flag for the CI52x: \\
0: NO_DIAG (no diagnosis evailable from CI52x I/O \\
cards) \\
1: DIAG_AVAILABLE (diagnosis available for CI52x I/O \\
cards)
\end{tabular} \\
\hline Diagnosis flag & BYTE & \\
& & \begin{tabular}{l} 
Terminal unit state for the CI52x: \\
0: NO_HOTSWAP_TU (not hot swap terminal unit \\
detected) \\
1: HOTSWAP_TU_RUNNING (hot swap terminal unit \\
detected and working) \\
2: HOTSWAP_TU_ERROR (hot swap terminal unit \\
detected, but communication errors for hot swap \\
terminal unit detected)
\end{tabular} \\
\hline Terminal unit state & BYTE & \\
& & \\
& &
\end{tabular}
\begin{tabular}{|c|c|c|}
\hline Data & DATA TYPE & Description \\
\hline Parameter state & BYTE & \begin{tabular}{l}
Parameter state of the CI52x: \\
0 : NO_PARA (module is in initialization phase and not ready for parameterization) \\
1: WAIT_PARA (module awaits parameterization) \\
2: PARA_RUN (parameterization running) \\
3: LEN_ERR (length of parameters not correct) \\
4: ID_ERR (module ID inside parameters not correct) \\
5: PARA_DONE (parameterization finished without errors)
\end{tabular} \\
\hline Module ID & WORD & Module ID of the \(1^{\text {st }}\) connected expansion module \\
\hline Expected module ID & WORD & Expected (configured) module ID of the \(1^{\text {st }}\) connected expansion module \\
\hline Module state & BYTE & The current module state of the \(1^{\text {st }}\) connected expansion module \\
\hline Diagnosis flag & BYTE & \begin{tabular}{l}
Diagnosis flag for the \(1^{\text {st }}\) connected expansion module 0 : NO_DIAG (no diagnosis evailable for expansion module) \\
1: DIAG_AVAILABLE (diagnosis available for expansion module)
\end{tabular} \\
\hline Terminal unit state & BYTE & Terminal unit state for the \(1^{\text {st }}\) connected expansion module \\
\hline Parameter state & BYTE & Parameter state of the \(1^{\text {st }}\) connected expansion module \\
\hline \(\ldots\) & & \\
\hline Module ID & WORD & Module ID of the \(10^{\text {th }}\) connected expansion module \\
\hline Expected module ID & WORD & Expected (configured) module ID of the \(10^{\text {th }}\) connected expansion module \\
\hline Module state & BYTE & The current module state of the \(10^{\text {th }}\) connected expansion module \\
\hline Diagnosis flag & BYTE & Diagnosis flag for the \(10^{\text {th }}\) connected expansion module \\
\hline Terminal unit state & BYTE & Terminal unit state for the \(10^{\text {th }}\) connected expansion module \\
\hline Parameter state & BYTE & Parameter state of the \(10^{\text {th }}\) connected expansion module \\
\hline
\end{tabular}

\section*{Diagnosis data}

The diagnosis data section contains one diagnosis message with the following structure (according to AC500 diagnosis):
\begin{tabular}{|c|c|c|}
\hline Byte Number & Description & Possible Values \\
\hline \multirow[t]{4}{*}{1} & \multirow[t]{4}{*}{Diagnosis Byte, slot number} & 31 = CI52x-MODTCP (e. g. error at integrated 8 DI / 8 DO) \\
\hline & & 1 = \(1^{\text {st }}\) connected S500 I/O Module \\
\hline & & \(\ldots\) \\
\hline & & \(10=10^{\text {th }}\) connected S500 I/O Module \\
\hline 2 & Diagnosis Byte, module number & According to the I/O bus specification passed on by modules to the fieldbus master \\
\hline 3 & Diagnosis Byte, channel & According to the I/O bus specification passed on by modules to the fieldbus master \\
\hline 4 & Diagnosis Byte, error code & \begin{tabular}{l}
According to the I/O bus specification Bit 7 and Bit 6, coded error class
\[
\begin{aligned}
& 0=E 1 \\
& 1=E 2 \\
& 2=E 3 \\
& 3=E 4
\end{aligned}
\] \\
Bit 0 to Bit 5, coded error description
\end{tabular} \\
\hline 5 & Diagnosis Byte, flags & \begin{tabular}{l}
According to the I/O bus specification \\
Bit 7: 1 = coming error \\
Bit 6: 1 = leaving error
\end{tabular} \\
\hline 6 & Reserved & 0 \\
\hline
\end{tabular}

If a diagnosis message is read out, the next one will be automatically filled in.
If no more diagnosis messages are available the buffer will be reset to zero.
This ensures that each diagnosis message can be delivered to the Modbus TCP client/slave and no diagnosis will be lost.

\section*{I/O data}

The I/O data section can use two different formats according to the module parameter "I/O Mapping Structure".
- Fixed I/O mapping

In case of fixed I/O mapping each module has a predefined register range for each Inputs and Outputs.
- Dynamic I/O mapping

In case of dynamic I/O mapping the mapping is build according to the actual configuration.
The dummy output at the end of the I/O data section can be used to retrigger the bus supervision and has no effect on the HW outputs.

Fixed I/O mapping

In case of fixed I/O mapping the following predefined register table is used:
\begin{tabular}{|r|l|l|l|}
\hline \begin{tabular}{r} 
Register \\
(hex)
\end{tabular} & Description & \begin{tabular}{l} 
Readable by \\
Modbus function \\
code
\end{tabular} & \begin{tabular}{l} 
Writeable by \\
Modbus function \\
code
\end{tabular} \\
\hline 1000 & Inputs CI & \(3,4,23\) & \(x\) \\
\hline 1100 & Inputs 1.EXP & \(3,4,23\) & x \\
\hline\(\ldots\) & \(\ldots\) & & x \\
\hline \(1 \mathrm{A00}\) & Inputs 10.EXP & \(3,4,23\) & x \\
\hline 2000 & Outputs CI & 3,23 & \(6,16,23\) \\
\hline 2100 & Outputs 1.EXP & 3,23 & \(6,16,23\) \\
\hline\(\ldots\) & \(\ldots\) & & \\
\hline 2 A 00 & Outputs 10.EXP & 3,23 & \(6,16,23\) \\
\hline \(2 \mathrm{B00}\) & Dummy output & 3,23 & \(6,16,23\) \\
\hline
\end{tabular}

If a certain expansion module has no inputs or outputs the corresponding registers remain empty.

Dynamic I/O mapping

In case of dynamic mapping only the start addresses of inputs and outputs are predefined:
\begin{tabular}{|r|l|l|l|}
\hline \begin{tabular}{r} 
Register \\
(hex)
\end{tabular} & Description & \begin{tabular}{l} 
Readable by \\
Modbus function \\
code
\end{tabular} & \begin{tabular}{l} 
Writeable by \\
Modbus function \\
code
\end{tabular} \\
\hline 1000 & Inputs CI & \(3,4,23\) & x \\
\hline\(\ldots\) & \(\ldots\) & & x \\
\hline 2000 & Outputs Cl & 3,23 & \(6,16,23\) \\
\hline\(\ldots\) & \(\ldots\) & & \\
\hline \(2 \mathrm{B00}\) & Dummy output & 3,23 & \(6,16,23\) \\
\hline
\end{tabular}

The register addresses of the connected expansion modules are calculated dynamically based on the number of inputs and outputs of the previous modules (each module starts directly on the next register after the previous module).

The register addresses of each module can be read out via the common device register \(\Leftrightarrow\) Chapter 6.8.4.1.2.2.3 "Common device information registers" on page 4572.

Comparative The difference between fixed I/O mapping and dynamic I/O mapping is shown in the following example table.

For this comparison a cluster with CI522, AX522, DC532, AX521, DC523, DC532, AO523, AI523, DI524, AX522 and DC523 is used.
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline \multicolumn{4}{|c|}{Fixed Mapping} & \multicolumn{4}{|c|}{Dynamic Mapping} \\
\hline Register (hex) & Description & Type & Data & Register (hex) & Description & Type & Data \\
\hline 1000 & Inputs Cl & \[
\begin{aligned}
& 8 \mathrm{DC}, 8 \mathrm{DI}, \\
& \text { FC }
\end{aligned}
\] & \[
\begin{aligned}
& 4 \text { BYTE + } 4 \\
& \text { WORD }
\end{aligned}
\] & 1000 & Inputs Cl & \[
\begin{aligned}
& 8 \mathrm{DC}, 8 \mathrm{DI}, \\
& \text { FC }
\end{aligned}
\] & \[
\begin{aligned}
& 4 \text { BYTE + } 4 \\
& \text { WORD }
\end{aligned}
\] \\
\hline 1100 & Inputs AX522 & 8 Al & 8 WORD & 1006 & Inputs AX522 & 8 Al & 8 WORD \\
\hline 1200 & Inputs DC532 & \[
\begin{aligned}
& 16 \mathrm{DI}, 16 \\
& \mathrm{DC}
\end{aligned}
\] & 4 BYTE & 100E & Inputs DC532 & \[
\begin{aligned}
& 16 \mathrm{DI}, 16 \\
& \mathrm{DC}
\end{aligned}
\] & 4 BYTE \\
\hline 1300 & Inputs AX521 & 4 Al & 4 WORD & 1010 & Inputs AX521 & 4 Al & 4 WORD \\
\hline 1400 & Inputs DC523 & 24 DC & 3 BYTE & 1014 & Inputs DC523 & 24 DC & 3 BYTE \\
\hline 1500 & Inputs DC532 & \[
\begin{aligned}
& 16 \mathrm{DI}, 16 \\
& \mathrm{DC}
\end{aligned}
\] & 4 BYTE & 1016 & Inputs DC532 & \[
\begin{aligned}
& 16 \mathrm{DI}, 16 \\
& \mathrm{DC}
\end{aligned}
\] & 4 BYTE \\
\hline 1600 & Inputs AO523 & --- & --- & --- & Inputs AO523 & --- & --- \\
\hline 1700 & Inputs AI523 & 16AI & 16 WORD & 1018 & Inputs AI523 & 16AI & 16 WORD \\
\hline 1800 & Inputs DI524 & 32 DI & 4 BYTE & 1028 & Inputs DI524 & 32 DI & 4 BYTE \\
\hline 1900 & Inputs AX522 & 8 Al & 8 WORD & 102A & Inputs AX522 & 8 Al & 8 WORD \\
\hline 1A00 & Inputs DC523 & 24 DC & 3 BYTE & 1032 & Inputs DC523 & 24 DC & 3 BYTE \\
\hline 2000 & Outputs Cl & \[
\begin{aligned}
& 8 \mathrm{DC}, 8 \mathrm{DO}, \\
& \text { FC }
\end{aligned}
\] & \[
\begin{aligned}
& 4 \text { BYTE + } 8 \\
& \text { WORD }
\end{aligned}
\] & 2000 & Outputs Cl & \[
\begin{aligned}
& 8 \mathrm{DC}, 8 \mathrm{DO}, \\
& \text { FC }
\end{aligned}
\] & \[
\begin{aligned}
& 4 \text { BYTE + } 8 \\
& \text { WORD }
\end{aligned}
\] \\
\hline 2100 & Outputs
AX522 & 8 AO & 8 WORD & 200A & Outputs AX522 & 8 AO & 8 WORD \\
\hline 2200 & Outputs DC532 & 16 DC & 2 BYTE & 2012 & Outputs DC532 & 16 DC & 2 BYTE \\
\hline 2300 & Outputs AX521 & 4 AO & 4 WORD & 2013 & Outputs AX521 & 4 AO & 4 WORD \\
\hline 2400 & Outputs DC523 & 24 DC & 3 BYTE & 2017 & Outputs DC523 & 24 DC & 3 BYTE \\
\hline 2500 & Outputs DC532 & 16 DC & 2 BYTE & 2019 & Outputs DC532 & 16 DC & 2 BYTE \\
\hline 2600 & Outputs AO523 & 16 AO & 16 WORD & 201A & Outputs AO523 & 16 AO & 16 WORD \\
\hline 2700 & Outputs AI523 & --- & --- & --- & Outputs AI523 & --- & --- \\
\hline 2800 & Outputs DI524 & --- & --- & --- & Outputs DI524 & --- & --- \\
\hline 2900 & Outputs AX522 & 8 AO & 8 WORD & 202A & Outputs AX522 & 8 AO & 8 WORD \\
\hline 2A00 & Outputs DC523 & 24 DC & 3 BYTE & 2032 & Outputs DC523 & 24 DC & 3 BYTE \\
\hline
\end{tabular}

When commissioning a CI521 module with byte order "big endian" in combination with a V3 PLC. BYTE and DWORD data of inputs and outputs has to be swapped by user, V3 PLC processor is using little endian as standard, high- and low bytes are interchanged! Analog values do not have to be swapped.

Table 848: I/O data (Inputs 20 BYTEs)
\begin{tabular}{|c|c|c|}
\hline Signal & DATA TYPE & Description \\
\hline AIO & WORD & Input value of the \(1^{\text {st }}\) analog input \\
\hline Al1 & WORD & Input value of the \(2^{\text {nd }}\) analog input \\
\hline Al2 & WORD & Input value of the \(3^{\text {rd }}\) analog input \\
\hline Al3 & WORD & Input value of the \(4^{\text {th }}\) analog input \\
\hline DI & BYTE & Input value of the DI channels \\
\hline Fast counter actual value counter 1 & DWORD & \\
\hline Fast counter actual value counter 2 & DWORD & \\
\hline Fast counter state counter 1 & BYTE & \\
\hline Fast counter state counter 2 & BYTE & \\
\hline Additional reserve byte & BYTE & reserved, not used \\
\hline
\end{tabular}

Table 849: I/O data (Outputs 24 BYTEs)
\begin{tabular}{|c|c|l|}
\hline Signal & DATA TYPE & \multicolumn{1}{|c|}{ Description } \\
\hline AO0 & WORD & Output value of the \(1^{\text {st }}\) analog output \\
\hline AO1 & WORD & \begin{tabular}{l} 
Output value of the \(2^{\text {nd }}\) analog \\
output
\end{tabular} \\
\hline DO & BYTE & Output value of the DO channels \\
\hline \begin{tabular}{c} 
Fast counter start value \\
counter 1
\end{tabular} & DWORD & \\
\hline \begin{tabular}{c} 
Fast counter end value \\
counter 1
\end{tabular} & DWORD & \\
\hline \begin{tabular}{c} 
Fast counter start value \\
counter 2
\end{tabular} & DWORD & \\
\hline \begin{tabular}{c} 
Fast counter end value \\
counter 2
\end{tabular} & DWORD & \\
\hline \begin{tabular}{c|c|}
\hline Fast counter control counter 1 & BYTE \\
\hline Fast counter control counter 2 & BYTE \\
\hline Additional reserve byte & BYTE
\end{tabular} \\
\hline
\end{tabular}

\section*{Process Data} Structure CI522MODTCP

When commissioning a CI522 module with byte order "big endian" in combination with a V3 PLC. BYTE and DWORD data of inputs and outputs has to be swapped by user, V3 PLC processor is using little endian as standard, high- and low bytes are interchanged! Analog values do not have to be swapped.

Table 850: I/O data (Inputs 12 BYTEs)
\begin{tabular}{|c|c|c|}
\hline Signal & DATA TYPE & \multicolumn{1}{c|}{ Description } \\
\hline DC & BYTE & Input value of the DC channels \\
\hline DI & BYTE & Input value of the DI channels \\
\hline \begin{tabular}{c} 
Fast counter actual value \\
counter 1
\end{tabular} & DWORD & \\
\cline { 1 - 2 } \begin{tabular}{c} 
Fast counter actual value \\
counter 2
\end{tabular} & DWORD & \\
\hline Fast counter state counter 1 & BYTE & \\
\hline Fast counter state counter 2 & BYTE & \\
\hline
\end{tabular}

To operate fast counter on CI52x devices in combination with a V3 PLC correctly via Modbus TCP, a corresponding sample project is available. Setup and use are described in an application example.

Table 851: I/O data (Outputs 20 BYTEs)
\begin{tabular}{|c|c|c|}
\hline Signal & DATA TYPE & \multicolumn{1}{|c|}{ Description } \\
\hline DC & BYTE & Output value of the DC channels \\
\hline DO & BYTE & Output value of the DO channels \\
\hline \begin{tabular}{c} 
Fast counter start value \\
counter 1
\end{tabular} & DWORD & \\
\hline \begin{tabular}{c} 
Fast counter end value \\
counter 1
\end{tabular} & DWORD & \\
\cline { 1 - 2 } \begin{tabular}{c} 
Fast counter start value \\
counter 2
\end{tabular} & DWORD & \\
\hline \begin{tabular}{c} 
Fast counter end value \\
counter 2
\end{tabular} & DWORD & \\
\hline Fast counter control counter 1 & BYTE & \\
\hline Fast counter control counter 2 & BYTE & \\
\hline
\end{tabular}

\section*{Parameter data (acyclic data)}

\section*{General}
\begin{tabular}{|r|l|l|l|}
\hline \begin{tabular}{r} 
Register \\
(hex)
\end{tabular} & Description & \begin{tabular}{l} 
Readable by \\
Modbus function \\
code
\end{tabular} & \begin{tabular}{l} 
Writeable by \\
Modbus function \\
code
\end{tabular} \\
\hline 3000 & Parameters CI & 3 & 6,16 \\
\hline 3080 & Stored parameters CI & 3 & \(x\) \\
\hline 3100 & Parameters 1. EXP & 3 & 6,16 \\
\hline 3180 & Stored parameters 1. EXP & 3 & \(x\) \\
\hline\(\ldots\) & & & \\
\hline 3A00 & Parameters 10. EXP & 3 & 6,16 \\
\hline 3A80 & Stored parameters 10. EXP & 3 & 6,16 \\
\hline 3B00 & controlword/statusword & 3 & \\
\hline
\end{tabular}

For each connected module the following parameter data are defined (the parameters are represented as ARRAY OF BYTE):
- Actual used parameter for each module

In these sections the actual parameters are stored. This section is also used to write parameters to the module.
« Chapter 6.8.4.1.3.2 "Parameterization" on page 4587
- Stored parameters for each module

If the module has stored nonvolatile parameters these can be read out using the corresponding registers.

The controlword/statusword is used to trigger a parameterization process.
\begin{tabular}{|c|l|}
\hline Bit & \multicolumn{1}{c|}{ Meaning } \\
\hline 0 & End of parameterization use parameters \\
1 & store parameters temporarily, use stored parameters after bus reconnect \\
2 & store parameters in flash, use stored parameters after power cycle \\
3 & reserved \\
4 & delete stored parameters in flash \\
5 & ignore parameter errors for saving \\
6 & reserved \\
7 & reserved \\
8 & new diagnosis available \\
9 & new parameters are available \\
10 & reserved \\
11 & reserved \\
12 & reserved \\
13 & reserved \\
14 & reserved \\
15 & reserved
\end{tabular}

Fig. 363: Meaning of the single bits
The direction of the first 8 bits is client to server (master to slave).
The direction of the second 8 bits is server to client (slave to master). * Description of the bits.
The parameter register sections (actual and stored parameters) have the structure as explained in the documentation of the corresponding module.

\section*{Short description of the CI521-MODTCP parameters}
\begin{tabular}{|c|c|c|c|}
\hline Parameter & Single parameter index & Description & Additional Info \\
\hline 0 & & Module ID (high Byte) & Fixed, must be 16\#1C \\
\hline 1 & & Module ID (low Byte) & Fixed, must be 16\#E8 \\
\hline 2 & & Ignore Module & Reserved, must be 0 \\
\hline 3 & & Length of following parameter block & Fixed, must be 16\#3F \\
\hline 4 & 0 & Error LED / Failsafe & \\
\hline 5 & 1 & Master IP Byte 0 & \multirow[t]{4}{*}{IP Address for write restrictions " "Configurable write restriction" on page 4589} \\
\hline 6 & & Master IP Byte 1 & \\
\hline 7 & & Master IP Byte 2 & \\
\hline 8 & & Master IP Byte 3 & \\
\hline 9 & 2 & Master IP 1 Byte 0 & \multirow[t]{4}{*}{IP Address for write restric tions (3) "Configurable write restriction" on page 4589} \\
\hline 10 & & Master IP 1 Byte 1 & \\
\hline 11 & & Master IP 1 Byte 2 & \\
\hline 12 & & Master IP 1 Byte 3 & \\
\hline 13 & 3 & Master IP 2 Byte 0 & \multirow[t]{4}{*}{IP Address for write restrictions 権" "Configurable write restriction" on page 4589} \\
\hline 14 & & Master IP 2 Byte 1 & \\
\hline 15 & & Master IP 2 Byte 2 & \\
\hline 16 & & Master IP 2 Byte 3 & \\
\hline 17 & 4 & Master IP 3 Byte 0 & \multirow[t]{4}{*}{IP Address for write restric tions 点"Configurable write restriction" on page 4589} \\
\hline 18 & & Master IP 3 Byte 1 & \\
\hline 19 & & Master IP 3 Byte 2 & \\
\hline 20 & & Master IP 3 Byte 3 & \\
\hline 21 & 5 & Master IP 4 Byte 0 & \multirow[t]{4}{*}{IP Address for write restrictions (3) "Configurable write restriction" on page 4589} \\
\hline 22 & & Master IP 4 Byte 1 & \\
\hline 23 & & Master IP 4 Byte 2 & \\
\hline 24 & & Master IP 4 Byte 3 & \\
\hline 25 & 6 & Master IP 5 Byte 0 & \multirow[t]{4}{*}{IP Address for write restrictions (3) "Configurable write restriction" on page 4589} \\
\hline 26 & & Master IP 5 Byte 1 & \\
\hline 27 & & Master IP 5 Byte 2 & \\
\hline 28 & & Master IP 5 Byte 3 & \\
\hline 29 & 7 & Master IP 6 Byte 0 & \multirow[t]{4}{*}{IP Address for write restrictions \({ }^{4}\), "Configurable write restriction" on page 4589} \\
\hline 30 & & Master IP 6 Byte 1 & \\
\hline 31 & & Master IP 6 Byte 2 & \\
\hline 32 & & Master IP 6 Byte 3 & \\
\hline 33 & 8 & Master IP 7 Byte 0 & \multirow[t]{2}{*}{IP Address for write restrictions \({ }^{\circledR}\) "Configurable write restriction" on page 4589} \\
\hline 34 & & Master IP 7 Byte 1 & \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|}
\hline Parameter & Single parameter index & Description & Additional Info \\
\hline 36 & & Master IP 7 Byte 2 & \\
\hline 36 & & Master IP 7 Byte 3 & \\
\hline 37 & 9 & Timeout & \begin{tabular}{l}
Timeout for bus supervision in 10 ms steps \\
if set to 0 no bus supervision is active
\end{tabular} \\
\hline 38 & 10 (read only) & I/O Mapping Structure & \\
\hline 39 & 11 & Reserved & Reserved, must be 0 \\
\hline 40 & 12 & Reserved & Reserved, must be 0 \\
\hline 41 & 13 & Reserved & Reserved, must be 0 \\
\hline 42 & 14 & Check supply & \\
\hline 43 & 15 & Analog data format & Reserved, must be 0 \\
\hline 44 & 16 & Input delay & \\
\hline 46 & 17 & Fast counter & \\
\hline 46 & 18 & Short circuit detection & \\
\hline 47 & 19 & Behavior binary outputs at com. fault & \\
\hline 48 & 20 & Substitute value binary outputs & \\
\hline 49 & 21 & Overvoltage monitoring & \\
\hline 50 & 22 & Behavior analog outputs & \\
\hline 51 & 23 & Channel Config AIO & \\
\hline 52 & 24 & Check Channel AIO & \\
\hline 53 & 25 & Channel Config Al1 & \\
\hline 54 & 26 & Check Channel Al1 & \\
\hline 55 & 27 & Channel Config Al2 & \\
\hline 56 & 28 & Check Channel Al2 & \\
\hline 57 & 29 & Channel Config Al3 & \\
\hline 58 & 30 & Check Channel Al3 & \\
\hline 59 & 31 & Channel Config AOO & \\
\hline 60 & 32 & Check Channel AOO & \\
\hline 61 & 33 & Substitute value AOO (high Byte) & \\
\hline 62 & & Substitute value AOO (low Byte) & \\
\hline 63 & 34 & Channel Config AO1 & \\
\hline 64 & 35 & Check Channel AO1 & \\
\hline 65 & 36 & Substitute value AO1 (high Byte) & \\
\hline 66 & & Substitute value AO1 (low Byte) & \\
\hline
\end{tabular}

Short description of the CI522-MODTCP parameters
\begin{tabular}{|c|c|c|c|}
\hline Parameter & Single parameter index & Description & Additional Info \\
\hline 0 & & Module ID (high Byte) & Fixed, must be 16\#1C \\
\hline 1 & & Module ID (low Byte) & Fixed, must be 16\#ED \\
\hline 2 & & Ignore Module & Reserved, must be 0 \\
\hline 3 & & Length of following parameter block & Fixed, must be 16\#2F \\
\hline 4 & 0 & Error LED / Failsafe & \\
\hline 5 & 1 & Master IP Byte 0 & \multirow[t]{4}{*}{IP Address for write restrictions \(\Leftrightarrow\) "Configurable write restriction" on page 4589} \\
\hline 6 & & Master IP Byte 1 & \\
\hline 7 & & Master IP Byte 2 & \\
\hline 8 & & Master IP Byte 3 & \\
\hline 9 & 2 & Master IP 1 Byte 0 & \multirow[t]{4}{*}{IP Address for write restrictions \({ }^{4}\) "Configurable write restriction" on page 4589} \\
\hline 10 & & Master IP 1 Byte 1 & \\
\hline 11 & & Master IP 1 Byte 2 & \\
\hline 12 & & Master IP 1 Byte 3 & \\
\hline 13 & 3 & Master IP 2 Byte 0 & \multirow[t]{4}{*}{IP Address for write restrictions "y "Configurable write restriction" on page 4589} \\
\hline 14 & & Master IP 2 Byte 1 & \\
\hline 15 & & Master IP 2 Byte 2 & \\
\hline 16 & & Master IP 2 Byte 3 & \\
\hline 17 & 4 & Master IP 3 Byte 0 & \multirow[t]{4}{*}{IP Address for write restrictions \({ }^{4}\) " "Configurable write restriction" on page 4589} \\
\hline 18 & & Master IP 3 Byte 1 & \\
\hline 19 & & Master IP 3 Byte 2 & \\
\hline 20 & & Master IP 3 Byte 3 & \\
\hline 21 & 5 & Master IP 4 Byte 0 & \multirow[t]{4}{*}{IP Address for write restrictions \({ }^{4}\) " "Configurable write restriction" on page 4589} \\
\hline 22 & & Master IP 4 Byte 1 & \\
\hline 23 & & Master IP 4 Byte 2 & \\
\hline 24 & & Master IP 4 Byte 3 & \\
\hline 25 & 6 & Master IP 5 Byte 0 & \multirow[t]{4}{*}{IP Address for write restrictions 缶"Configurable write restriction" on page 4589} \\
\hline 26 & & Master IP 5 Byte 1 & \\
\hline 27 & & Master IP 5 Byte 2 & \\
\hline 28 & & Master IP 5 Byte 3 & \\
\hline 29 & 7 & Master IP 6 Byte 0 & \multirow[t]{4}{*}{IP Address for write restrictions \({ }^{4}\) "Configurable write restriction" on page 4589} \\
\hline 30 & & Master IP 6 Byte 1 & \\
\hline 31 & & Master IP 6 Byte 2 & \\
\hline 32 & & Master IP 6 Byte 3 & \\
\hline 33 & 8 & Master IP 7 Byte 0 & \multirow[t]{4}{*}{IP Address for write restrictions «y "Configurable write restriction" on page 4589} \\
\hline 34 & & Master IP 7 Byte 1 & \\
\hline 36 & & Master IP 7 Byte 2 & \\
\hline 36 & & Master IP 7 Byte 3 & \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|}
\hline Parameter & Single parameter index & Description & Additional Info \\
\hline 37 & 2 & Timeout & \begin{tabular}{l}
Timeout for bus supervision in 10 ms steps \\
if set to 0 no bus supervision is active
\end{tabular} \\
\hline 38 & 3 (read only) & I/O Mapping Structure & \\
\hline 39 & 4 & Reserved & \multirow[t]{3}{*}{Reserved, must be 0} \\
\hline 40 & 5 & Reserved & \\
\hline 41 & 6 & Reserved & \\
\hline 42 & 7 & Check supply & \\
\hline 43 & 8 & Input delay & \\
\hline 44 & 9 & Fast counter & \\
\hline 46 & 10 & Short circuit detection & \\
\hline 46 & 11 & Behavior binary outputs at com. fault & \\
\hline 47 & 12 & Substitute value binary outputs (high byte) & \\
\hline 48 & & Substitute value binary outputs (low byte) & \\
\hline 49 & 13 & Voltage feedback monitoring & \\
\hline 50 & 14 & Overvoltage monitoring & \\
\hline
\end{tabular}

\section*{Parameters of connected expansion modules}

The parameters of the connected expansion modules are represented as byte array (the parameters valid for "CPU" in the corresponding module are used):
\begin{tabular}{|c|l|l|}
\hline Parameter & \multicolumn{1}{c|}{ Description } & \multicolumn{1}{c|}{ Additional Info } \\
\hline 0 & Module ID (high byte) & \begin{tabular}{l} 
Fixed, see corresponding module (the \\
module ID of FBP is used)
\end{tabular} \\
\hline 1 & Module ID (low byte) & \begin{tabular}{l} 
Fixed, see corresponding module (the \\
module ID of FBP is used)
\end{tabular} \\
\hline 2 & Ignore module & Reserved must be 0 \\
\hline 3 & Length of following parameter block & Fixed, see corresponding module \\
\hline \(4 \ldots\) & \begin{tabular}{l} 
The rest of the parameter are \\
described in the corresponding \\
module
\end{tabular} & \\
\hline
\end{tabular}

\section*{Special functionality}

This section contains special services like firmware update or single parameterization.
\begin{tabular}{|c|c|c|c|}
\hline Register (hex) & Description & Readable by Modbus function code & Writeable by Modbus function code \\
\hline 4000 & Firmware download & 3 & 16 \\
\hline 4100 & Firmware download state & 3 & X \\
\hline 5000 & Write single parameterization of Cl & X & 16 \\
\hline 5100 & Write single parameterization of 1 . EXP & X & 16 \\
\hline \(\ldots\) & & & \\
\hline 5A00 & Write single parameterization of 10 . EXP & X & 16 \\
\hline 6000 & Read single parameterization of Cl & 3 & 16 \\
\hline 6100 & Read single parameterization of 1. EXP & 3 & 16 \\
\hline \(\ldots\) & & & \\
\hline 6A00 & Read single parameterization of 10. EXP & 3 & 16 \\
\hline
\end{tabular}

\subsection*{6.8.4.1.3 Behavior}

\section*{IP address assignment}

\section*{General}

The delivery IP address of the CI52x-MODTCP is 192.168.0.xx. xx is the hardware address switch position of the device.
The devices support BOOTP, DHCP and fixed IP address setting. These can be set individual or together.

If BOOTP and DHCP are enabled the following priority takes place:
- If DHCP configuration fails, the device will fall back to BOOTP.
- In case of a BOOTP failure, the fixed IP address will be used.

A new IP address (or changing of BOOTP and DHCP) can be set in two different ways:
- With the address switches of the corresponding module
- With the IP configuration tool
y Chapter 6.3.2.9.2 "Configuration of the IP settings with the IP configuration tool" on page 1506

\section*{Using the address switches}

With the address switches only the last byte of the IP address can be changed.
The IP address can only be set via the address switches in case of factory default or in case of the last byte of the IP address is set to zero.
„ Chapter 6.3.2.9.2 "Configuration of the IP settings with the IP configuration tool" on page 1506

The not allowed IP addresses are mapped as followed:
- Address switch position 255 is mapped to fixed IP 192.168.0.254 independent of other stored settings (by IP Configuration Tool).
This is a backup so the module can always get a valid IP address and can be configured by the IP Configuration Tool.
- Address switch position 0 is mapped to last byte equal 1 and DHCP enabled.

\section*{Using the IP configuration tool}

With the 'Configuration of the IP settings with the IP configuration tool' a network scan can be executed, and the found devices can be assigned with new settings, e.g. enable BOOTP or DHCP and set a new fixed IP.
\(\left.{ }^{*}\right\rangle\) Chapter 6.3.2.9.2 "Configuration of the IP settings with the IP configuration tool" on page 1506

If the last byte of the IP address of the CI52x-MODTCP devices is set to 0 with the IP Configuration Tool the address switch position is used instead.
\& Chapter 6.8.4.1.3.1.2 "Using the address switches" on page 4587

\section*{Parameterization}

The parameterization is done via the corresponding registers explained in the Modbus TCP registers \({ }^{4}\), Chapter 6.8.4.1.2.4 "Parameter data (acyclic data)" on page 4581.

In addition to that the parameters can be directly transferred via Automation Builder.
ⓨ Chapter 6.3.2.12.2 "Unbundled CI52x-MODTCP configuration" on page 1565
There are two different parameter sections with different behavior.

\section*{Actual used parameters}

After startup this section contains the following data:
- Default parameters (only module id and parameter length set all others zero) if no valid stored parameters are available (no or invalid parameters stored).
- Actual used / stored parameters if valid parameters are stored nonvolatile.

These parameters can be read out and changed by reading or writing of the corresponding registers, but will not be used automatically after writing them, the use of new written parameters has to be triggered by writing the parameter control word with the corresponding bits set (see below).

\section*{Stored parameters}

This section always contains a copy of the nonvolatile stored parameters, if no parameters are stored nonvolatile this sections will be 0 .

\section*{Controlword/statusword parameter}

This parameter can be used to trigger and save new parameters.
The direction of the first 8 bit is client to server (master to slave). The direction of the second 8 bits is server to client (slave to master).
\begin{tabular}{|c|l|l|}
\hline Bit & \multicolumn{2}{c|}{ Description } \\
\hline 0 & \begin{tabular}{l} 
Use parameters / start \\
parameterization
\end{tabular} & \begin{tabular}{l} 
lf this bit is set the CI Device starts the parameterization \\
with the parameters in the actual parameters registers.
\end{tabular} \\
\hline 1 & Store parameters volatile & \begin{tabular}{l} 
lf this bit is set the CI device will use the parameters \\
temporarily, which means after a bus error detection and \\
reconnection the parameters will be used again. \\
This bit should always be set. \\
This bit is only evaluated when bit \(\mathbf{0}\) is set.
\end{tabular} \\
\hline 2 & \begin{tabular}{l} 
Store parameters \\
nonvolatile
\end{tabular} & \begin{tabular}{l} 
If this bit is set the CI device will store the parameters \\
nonvolatile, which means after a power cycle the stored \\
parameter data will be used again. \\
This bit is only evaluated when bit \(\mathbf{0}\) is set.
\end{tabular} \\
\hline 3 & Reserved & - \\
\hline 4 & \begin{tabular}{l} 
Delete nonvolatile stored \\
parameters
\end{tabular} & \begin{tabular}{l} 
If this bit is set the CI device will delete its nonvolatile \\
stored parameters. \\
This bit is only evaluated when bit \(\mathbf{0}\) is set.
\end{tabular} \\
\hline 5 & \begin{tabular}{l} 
lgnore parameter error \\
for nonvolatile parameter \\
storage
\end{tabular} & \begin{tabular}{l} 
lf this bit is set a parameter error during nonvola- \\
tile storage of parameters will be ignored, and the \\
parameters will be stored. \\
This bit can only be set in combination with bit \(\mathbf{0}\) and \\
bit 2.
\end{tabular} \\
\hline 6 & Reserved & - \\
\hline 7 & Reserved & - \\
\hline 8 & New diagnosis available & \begin{tabular}{l} 
The device will set this bit if new diagnosis data are \\
available in the diagnosis data section.
\end{tabular} \\
\hline 9 & New parameters available & \begin{tabular}{l} 
The device will set this bit if new parameters are \\
available in the actual parameter data section and these \\
were not activated by setting bit 0 in the control word.
\end{tabular} \\
\hline 10 N..15 & Reserved & - \\
\hline
\end{tabular}

\section*{Cyclic I/O data exchange}

The I/O data can be exchanged cyclic by the master by reading, writing the corresponding registers.

I/O data exchange is only possible after successful parameterization of the device.
For writing of outputs bus failure detection can be activated by setting the corresponding parameter. This bus failure detection is described in the following chapter.

Bus failure If the parameter ""timeout"" in the module parameters of the CI52x-MODTCP is set, the module detection will supervise the Modbus TCP "write telegrams".

After the first "write telegram" the bus will be supervised. If no new "write telegram" arrives at the CI52x-MODTCP within the configured time, the module will detect a bus failure and switch off its outputs or switch them to the configured failsafe state \({ }^{*} \Rightarrow\) Chapter 5.2.6.5.1.8 "Parameterization" on page 1090 \# Chapter 5.2.6.5.2.8 "Parameterization" on page 1120.

\section*{Configurable With the module parameters "Master IP"- "Master IP 7" it is possible to set write restrictions on write restriction the \(\mathrm{CI} 52 \mathrm{x}-\mathrm{MODTCP}\) device.}

If none of the parameters is set, all masters / clients in the network have read and write rights on the CI52x-MODTCP device and its connected expansion modules.

If at least one parameter is set only the configured masters / clients have write rights on the CI52x-MODTCP device.

All other masters / clients still have read access to the CI52x-MODTCP device.

\section*{Diagnosis behavior}

Each diagnosis message signals if this error is coming or going, so it is possible to create a list in the master of actual pending diagnosis.

Diagnosis messages will be transferred again after a bus failure detection and reconnection.
Diagnosis messages can be read out with function code \(3,4,23\). Function codes 3 and 4 can always read out diagnosis messages, function code 23 can only read out after successful parameterization of the device. See also table 'Diagnosis data' \(\Leftrightarrow\) Chapter 6.8.4.1.2.3.3 "Diagnosis data" on page 4576.

\section*{Single parameterization}

The single parameterization services can be used to read or write parameters during run time of device without the need of triggering a new parameterization process.

For indexes used for single parameterization services see parameter lists in section Modbus TCP registers of this document.

The read and write parameterization services are explained below, for each module (CI52x-MODTCP and connected expansion modules) a different section for read and write is defined see chapter Modbus TCP registers in this document). Both services are using the following data structure:


The length of the read / write service depends on the count of parameters that should be transferred (length \(=4+\) count* \(^{*} 8\) ).

\section*{Reading of single parameters}

The read single parameterization works in two steps:
- Writing of a request list containing the indexes that should be read using the structure explained above.
Only CNT and PARA_IDX has to be set.
Up to 5 parameters can be requested with one telegram.
The length of the write service depends on the count of parameters that should be transferred (length \(=4+\) count \(^{*} 8\) ).
- Reading of the parameters list with the same length then the previous write request. If the internal reading process inside the CI 52 x -MODTCP device is done the data will be read out.
If the internal reading process inside the \(\mathrm{CI} 52 \mathrm{x}-\mathrm{MODTCP}\) device is not yet finished the read service will be rejected with Modbus TCP exception code 6 (device busy).

Writing of single For writing of single parameters only one step is necessary, the parameters are transferred with parameters one write request using the structure described above.

The length of the write service depends on the count of parameters that should be transferred (length \(=4+\) count* 8 ).

In case of write of single parameters the following values have to be set:
- CNT: number of parameters to be set
- And for each parameter:

Parameter index
Parameter length
New parameter value
Written single parameters are not stored volatile and not stored nonvolatile. That means after a bus reconnection or power cycle the written parameters will be discarded.

\subsection*{6.8.4.1.4 Commissioning example}

Set IP Address:
- The setting of the IP address is the first step to integrate the CI52x-MODTCP devices into a running system.
- The setting of the IP address of the CI52x-MODTCP devices is described in the chapter 'IP address assignment' \(\Leftrightarrow\) Chapter 6.8.4.1.3.1 "IP address assignment" on page 4586 in this document

Set Parameters (optional read parameters):
- The second step in configuring the CI52x-MODTCP devices is to set the module and channel parameters.
- A read of parameters is optional but can be used the get the module IDs and the parameter length.
- The reading and or writing of parameters is described in chapter 'Parameterization' \& Chapter 6.8.4.1.3.2 "Parameterization" on page 4587.
Set Control Word:
- After setting the parameter data these have to be activated by writing the control word.
- The meaning and usage of the control word is described in chapter 'Parameterization' Chapter 6.8.4.1.3.2 "Parameterization" on page 4587.

Exchange data:
- After setting and activating the parameters the CI52x-MODTCP device is ready for data exchange.
- The registers for data exchange are described in chapter 'I/O / Process data and diagnosis section (Cyclic data)' \(\Longleftrightarrow\) Chapter 6.8.4.1.2.3 "I/O / Process data and diagnosis section (cyclic data)" on page 4573.

\subsection*{6.8.4.1.5 Hot swap}

\section*{Introduction}

With hot swap for AC500 and S500 it is possible to exchange I/O modules (with same type) during runtime.

\section*{Preconditions for using hot swap}

\section*{Hot swap}

\section*{WARNING!}

Risk of explosion or fire in hazardous environments during hot swapping!
Hot swap must not be performed in flammable environments to avoid life-threatening injury and property damage resulting from fire or explosion.

\section*{WARNING!}

\section*{Electric shock due to negligent behavior during hot swapping!}

To avoid electric shock
- make sure the following conditions apply:
- Digital outputs are not under load.
- Input/output voltages above safety extra low voltage/ protective extra low voltage (SELV/PELV) are switched off.
- Modules are fully interlocked with the terminal unit with both snap-fits engaged before switching on loads or input/output voltage.
- Never touch exposed contacts (dangerous voltages).
- Stay away from electrical contacts to avoid arc discharge.
- Do not operate a mechanical installation improperly.

\section*{NOTICE!}

Risk of damage to I/O modules!
Hot swapping is only allowed for I/O modules.
Processor modules and communication interface modules must not be removed or inserted during operation.

H = Hot swap

\section*{Hot swap}

System requirements for hot swapping of I/O modules:
- Types of terminal units that support hot swapping of I/O modules have the appendix TU5xx-H.
- I/O modules as of index FO.

The following I/O bus masters support hot swapping of attached I/O modules:
- Communication interface modules CI5xx as of index FO.
- Processor modules PM56xx-2ETH with firmware version as of V3.2.0.

Hot swap is not supported by AC500-eCo V3 CPU!


The index of the module is in the right corner of the label.

\section*{NOTICE!}

Risk of damage to I/O modules!
Modules with index below F0 can be damaged when inserted or removed from the terminal unit in a powered system.

\section*{NOTICE!}

Risk of damage to I/O modules!
Do not perform hot swapping if any I/O module with firmware version lower than 3.0.14 is part of the I/O configuration.

For min. required device index see table below.
\begin{tabular}{|l|l|}
\hline Device & \begin{tabular}{l} 
Min. required device index for I/O module as of \\
FW Version 3.0.14
\end{tabular} \\
\hline AC522(-XC) & F0 \\
\hline AI523 (-XC) & D2 \\
\hline AI531 & D4 \\
\hline AI531-XC & D2 \\
\hline AI561 & B2 \\
\hline AI562 & B2 \\
\hline AI563 & B3 \\
\hline AO523 (-XC) & D2 \\
\hline AO561 & B2 \\
\hline AX521 (-XC) & D2 \\
\hline AX522 (-XC) & D2 \\
\hline AX561 & B2 \\
\hline CD522 (-XC) & D1 \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline Device & \begin{tabular}{l} 
Min. required device index for I/O module as of \\
FW Version 3.0.14
\end{tabular} \\
\hline DA501 (-XC) & D2 \\
\hline DA502 (-XC) & F0 \\
\hline DC522 (-XC) & D2 \\
\hline DC523 (-XC) & D2 \\
\hline DC532 (-XC) & D2 \\
\hline DC562 & A2 \\
\hline D1524 (-XC) & D2 \\
\hline D1561 & B2 \\
\hline D1562 & B2 \\
\hline D1571 & B2 \\
\hline D1572 & A1 \\
\hline DO524 (-XC) & A3 \\
\hline DO526 & A2 \\
\hline DO526-XC & A0 \\
\hline DO561 & B2 \\
\hline DO562 & A2 \\
\hline DO571 & B3 \\
\hline DO572 & B2 \\
\hline DO573 & A1 \\
\hline DX522 (-XC) & D2 \\
\hline DX531 & D2 \\
\hline DX561 & B2 \\
\hline DX571 & B3 \\
\hline FM562 & A1 \\
\hline & \\
\hline
\end{tabular}

\section*{Compatibility of hot swap}
\begin{tabular}{|l|l|}
\hline & Modbus remote I/O \\
\hline \begin{tabular}{l} 
I/O module on TU5xx-H connected to I/O bus \\
master
\end{tabular} & CI521-MODTCP or CI522-MODTCP \\
\hline Required version of I/O bus master & \begin{tabular}{l} 
Module index as of F0 \\
Firmware as of V3.2.3
\end{tabular} \\
\hline \begin{tabular}{l} 
Fieldbus master when used as remote I/O with \\
AC500 V3
\end{tabular} & Any AC500 V3 CPU with on-board Ethernet \\
\hline \begin{tabular}{l} 
When used as remote I/O on third party con- \\
troller (PLC or DCS)
\end{tabular} & No limitation known \\
\hline
\end{tabular}

\section*{Hot swap behavior}

Table 852: Behavior in case of I/O attached to communication interface module for Modbus
TCP, CI521-MODTCP or CI522-MODTCP
\begin{tabular}{|c|c|}
\hline Hot swap behavior & Modbus TCP remote I/O \\
\hline Start-up behavior with missing or damaged I/O module on hot swap terminal unit TU5xx-H & \begin{tabular}{l}
Remote I/O station is not starting \\
As of device index F4 and Automation Builder Version 2.4.1 it is possible to configure the startup in case of missing modules on hot swap terminal units. If configured, the remote I/O station is starting up with missing or damaged \(I / O\) module, if the module is plugged later or replaced it will be automatically parameterized and I/O data will be exchanged. As the Automation Builder checks that all modules are available during configuration process, it is necessary that all I/O modules are available and in working order during configuration via Automation Builder. As the parameters are stored nonvolatile inside the Cl 52 x devices later one the parameters have effect for power cycle or reconnection operations.
\end{tabular} \\
\hline Start-up behavior with wrong I/O module type on any terminal unit & Remote I/O station is not starting \\
\hline Diagnosis of presence of hot swap terminal unit & \begin{tabular}{l}
Information is available in Modbus registers of the communication interface module which can be accessed by the application program \\
As of device index F4 and Automation Builder Version 2.4.1 it is possible to configure a list of required hot swap terminal units. If a required hot swap terminal unit is missing (normal one plugged) this will not prevent a normal operation but a diagnosis message will be generated for the corresponding slot.
\end{tabular} \\
\hline Diagnosis of hot swap capability of I/O module mounted on hot swap terminal unit & \begin{tabular}{l}
Information can be obtained by reading Modbus registers in the communication interface module. Those Modbus registers contain: \\
- Diagnosis in case that a not hot-swappable I/O module is plugged on a hot swap terminal unit \\
- Diagnosis In case that in a mixed configuration with at least one hot swap terminal unit an I/O module, that must not be used in a hot swap configuration, is mounted on any terminal unit of the configuration \\
- Production data and version index of the modules
\end{tabular} \\
\hline Diagnosis while hot swap module is pulled or module (mounted on hot swap terminal unit) has stopped working & Diagnosis is available in Modbus registers in the communication interface module \\
\hline Input state in process image of controller while module is pulled or module is not operational & Input = ZERO \\
\hline Diagnosis after plugging the I/O module on the hot swap terminal unit & Diagnose "diagnosis gone" is available in Modbus registers in the communication interface module \\
\hline
\end{tabular}

\section*{System behavior}

If an expansion module is removed or defective during run time, the input data of this module will be set to " 0 " and the module state will be set to the corresponding value (see \({ }^{4}\) ) Chapter 6.8.4.1.2.3 "I/O / Process data and diagnosis section (cyclic data)" on page 4573). A diagnosis message will be created in that case (see hardware description of CI521 \({ }^{\mu}\) Chapter 5.2.6.5.1 "Cl521-MODTCP" on page 1070 / CI522 \(\Leftrightarrow\) Chapter 5.2.6.5.2 "CI522-MODTCP" on page 1111 for diagnosis messages).

In case a module is replaced, the new module will automatically be parameterized with the last parameters of the removed module (if single parameters were written to the previously removed module, this parameters will be ignored).

During pulling or plugging of a certain module, all other module will continue to operate with one limitation: The reaction time of modules connected to the right of the affected module will be bigger in that case (up to 50 ms ).

If the bus failure detection is active for Cl 52 x and failsafe is configured \(\Leftrightarrow\) Chapter 6.8.4.1.3.3 "Cyclic I/O data exchange" on page 4588 the following behavior applies if a module is removed and replugged during failsafe condition:
- Last value configured for output:
- After a bus failure is detected, failsafe will be activated and the output will remain at its last value.
- If the module is removed and plugged again, the output will remain off, and not be kept its last value, as the last value of the new module is " 0 " in that case.
- Substitute value configured for output:
- After a bus failure is detected, failsafe will be activated and the output will be according to the configured substitute value.
- If the module is removed and plugged again now, the output will be set according to the configured substitute value again.
- Substitute value for \(x\) seconds configured for output:
- After a bus failure is detected, failsafe will be activated and the output will be according to the configured substitute value for the configured time.
- If the module is removed and plugged again now, the output will be set according to the configured substitute value again, and the configured time starts again.

\section*{Mandatory rules for hot swapping}

Mandatory rules for hot swapping:
- Between two pull and / or plug operations of I/O modules a pause of at least 1 second must be observed.
- That means if a module is pulled or plugged there has to be at least a break of 1 second before the next module is pulled or plugged.
- At boot up of CI52x all configured expansion modules have to be physically available.
- Start up with missing modules is not supported.
- In the application program it is possible to detect if a hot swap terminal unit is mounted in a specific position on the I/O bus. The information is available in the common device information registers. These can be accessed when the version of the communication interface module supports hot swap.
- This has to be checked by application:

Best way for checking if a hot swap terminal unit is available or not, is reading out the common device information registers \(\stackrel{y}{ } \boldsymbol{y}\) Chapter 6.8.4.1.2.2 "Information data section (acyclic data)" on page 4570 . If the CI 52 x rejects this read out the CI 52 x doesn't support hot swap at all.

\subsection*{6.8.4.2 PROFINET communication interface module}

\subsection*{6.8.4.2.1 Hot swap}

\section*{Introduction}

With hot swap for AC500 and S500 it is possible to exchange I/O modules (with same type) during runtime.

\section*{Preconditions for Using Hot Swap}

\section*{Hot swap}

\section*{WARNING!}

Risk of explosion or fire in hazardous environments during hot swapping!
Hot swap must not be performed in flammable environments to avoid life-threatening injury and property damage resulting from fire or explosion.

\section*{WARNING!}

Electric shock due to negligent behavior during hot swapping!
To avoid electric shock
- make sure the following conditions apply:
- Digital outputs are not under load.
- Input/output voltages above safety extra low voltage/ protective extra low voltage (SELV/PELV) are switched off.
- Modules are fully interlocked with the terminal unit with both snap-fits engaged before switching on loads or input/output voltage.
- Never touch exposed contacts (dangerous voltages).
- Stay away from electrical contacts to avoid arc discharge.
- Do not operate a mechanical installation improperly.

\section*{NOTICE!}

\section*{Risk of damage to I/O modules!}

Hot swapping is only allowed for I/O modules.
Processor modules and communication interface modules must not be removed or inserted during operation.

H = Hot swap

\section*{Hot swap}

System requirements for hot swapping of I/O modules:
- Types of terminal units that support hot swapping of I/O modules have the appendix TU5xx-H.
- I/O modules as of index FO.

The following I/O bus masters support hot swapping of attached I/O modules:
- Communication interface modules CI5xx as of index F0.
- Processor modules PM56xx-2ETH with firmware version as of V3.2.0.

Hot swap is not supported by AC500-eCo V3 CPU!


The index of the module is in the right corner of the label.

\section*{NOTICE!}

Risk of damage to I/O modules!
Modules with index below F0 can be damaged when inserted or removed from the terminal unit in a powered system.

\section*{NOTICE!}

Risk of damage to I/O modules!
Do not perform hot swapping if any I/O module with firmware version lower than 3.0.14 is part of the I/O configuration.

For min. required device index see table below.
\begin{tabular}{|l|l|}
\hline Device & \begin{tabular}{l} 
Min. required device index for I/O module as of \\
FW Version 3.0.14
\end{tabular} \\
\hline AC522(-XC) & F0 \\
\hline AI523 (-XC) & D2 \\
\hline AI531 & D4 \\
\hline AI531-XC & D2 \\
\hline AI561 & B2 \\
\hline AI562 & B2 \\
\hline AI563 & B3 \\
\hline AO523 (-XC) & D2 \\
\hline AO561 & B2 \\
\hline AX521 (-XC) & D2 \\
\hline AX522 (-XC) & D2 \\
\hline AX561 & B2 \\
\hline CD522 (-XC) & D1 \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline Device & \begin{tabular}{l} 
Min. required device index for I/O module as of \\
FW Version 3.0.14
\end{tabular} \\
\hline DA501 (-XC) & D2 \\
\hline DA502 (-XC) & F0 \\
\hline DC522 (-XC) & D2 \\
\hline DC523 (-XC) & D2 \\
\hline DC532 (-XC) & D2 \\
\hline DC562 & A2 \\
\hline D1524 (-XC) & D2 \\
\hline D1561 & B2 \\
\hline D1562 & B2 \\
\hline D1571 & B2 \\
\hline D1572 & A1 \\
\hline DO524 (-XC) & A3 \\
\hline DO526 & A2 \\
\hline DO526-XC & A0 \\
\hline DO561 & B2 \\
\hline DO562 & A2 \\
\hline DO571 & B3 \\
\hline DO572 & B2 \\
\hline DO573 & A1 \\
\hline DX522 (-XC) & D2 \\
\hline DX531 & D2 \\
\hline DX561 & B2 \\
\hline DX571 & B3 \\
\hline FM562 & A1 \\
\hline & \\
\hline
\end{tabular}

\section*{Compatibility of hot swap}
\begin{tabular}{|l|l|}
\hline & PROFINET remote I/O \\
\hline \begin{tabular}{l} 
I/O module on TU5xx-H connected to I/O bus \\
master
\end{tabular} & CI501-PNIO or CI502-PNIO \\
\hline Required version of I/O bus master & \begin{tabular}{l} 
Module index as of F0 \\
Firmware as of V3.2.10
\end{tabular} \\
\hline \begin{tabular}{l} 
Fieldbus master when used as remote I/O with \\
AC500 V3
\end{tabular} & Not supported \\
\hline \begin{tabular}{l} 
When used as remote I/O on third party con- \\
troller (PLC or DCS)
\end{tabular} & \begin{tabular}{l} 
Note: alarms must be acknowledged by \\
fieldbus master. \\
GSDML as of version
\end{tabular} \\
& \begin{tabular}{l} 
GSDML-V2.3-ABB-S500-CI501- \\
PNIO-20180822.xml or \\
GSDML-V2.3-ABB-S500-CI502-- \\
PNIO-20180822.xml \\
needed for full scope of vendor specific \\
diagnosis.
\end{tabular} \\
\hline
\end{tabular}

\section*{Hot swap behavior}

The following table describes the behavior in case of I/O attached to communication interface module for PROFINET, CI501-PNIO or CI502-PNIO.
\begin{tabular}{|c|c|}
\hline Hot swap behavior & PROFINET remote I/O with AC500 V3 CPU and CM579-PNIO controller \\
\hline & PROFINET remote I/O with third party controller \\
\hline Start-up behavior with missing or damaged I/O module on hot swap terminal unit TU5xx-H & \begin{tabular}{l}
Remote I/O station is not starting \\
As of device index F1 and Automation Builder Version 2.4.1 it is possible to configure the startup in case of missing modules on hot swap terminal units. If configured, the remote I/O station is starting up with missing or damaged I/O module, if the module is plugged later or replaced it will be automatically parameterized and I/O data will be exchanged.
\end{tabular} \\
\hline Start-up behavior with wrong I/O module type on any terminal unit & Remote I/O station is not starting \\
\hline Diagnosis of presence of hot swap terminal unit & \begin{tabular}{l}
Information is available either: \\
- via acyclic services or \\
- as cyclic state information in the process image \\
As of device index F1 and Automation Builder Version 2.4.1 it is possible to configure a list of required hot swap terminal units. If a required hot swap terminal unit is missing (normal one plugged) this will not prevent a normal operation but a diagnosis message will be generated for the corresponding slot.
\end{tabular} \\
\hline Diagnosis of hot swap capability of I/O module mounted on hot swap terminal unit & \begin{tabular}{l}
Diagnosis is transmitted as vendor specific PROFINET channel diagnosis: \\
- Diagnosis in case that a not hot-swappable I/O module is plugged on a hot swap terminal unit \\
- Diagnosis in case that in a mixed configuration with at least one hot swap terminal unit an I/O module, that must not be used in a hot swap configuration, is mounted on any terminal unit of the configuration \\
Production data and version index of the modules is accessible via acyclic services
\end{tabular} \\
\hline Diagnosis while hot swap module is pulled or module (mounted on hot swap terminal unit) has stopped working & PROFINET channel diagnosis is generated together with standard "pull alarm" which must be acknowledged \\
\hline Input state in process image of controller while module is pulled or module is not operational & \begin{tabular}{l}
Input = ZERO \\
In addition a standard PROFINET state information is transmitted saying "Inputs not valid"
\end{tabular} \\
\hline Diagnosis after plugging of the I/O module on the hot swap terminal unit & PROFINET channel diagnosis is generated together with standard "plug alarm" which must be acknowledged \\
\hline
\end{tabular}

\section*{System behavior}

If an expansion module is removed or defective during run time, the input data of this module will be set to " 0 " and the module state will be set to the corresponding value. A diagnosis message will be created in that case (see hardware description of CI501 \(\Leftrightarrow\) Chapter 5.2.6.7.2 "Cl501-PNIO" on page 1203 / CI502 \(\Rightarrow\) Chapter 5.2.6.7.3 "CI502-PNIO" on page 1244 for diagnosis messages).

In case a module is replaced, the new module will automatically be parameterized with the last parameters of the removed module (if single parameters were written to the previously removed module, this parameters will be ignored).

During pulling or plugging of a certain module, all other module will continue to operate with one limitation: The reaction time of modules connected to the right of the affected module will be bigger in that case (up to 50 ms ).
If the bus failure detection is active for CI 50 x and failsafe is configured the following behavior applies if a module is removed and replugged during failsafe condition:
- Last value configured for output:
- After a bus failure is detected, failsafe will be activated and the output will remain at its last value.
- If the module is removed and plugged again, the output will remain off, and not be kept its last value, as the last value of the new module is " 0 " in that case.
- Substitute value configured for output:
- After a bus failure is detected, failsafe will be activated and the output will be according to the configured substitute value.
- If the module is removed and plugged again now, the output will be set according to the configured substitute value again.
- Substitute value for \(x\) seconds configured for output:
- After a bus failure is detected, failsafe will be activated and the output will be according to the configured substitute value for the configured time.
- If the module is removed and plugged again now, the output will be set according to the configured substitute value again, and the configured time starts again.

\section*{Mandatory rules for hot swapping}

Mandatory rules for hot swapping:
- Between two pull and / or plug operations of I/O modules a pause of at least 1 second must be observed.
- That means if a module is pulled or plugged there has to be at least a break of 1 second before the next module is pulled or plugged.
- At boot up of CI50x all configured expansion modules have to be physically available.
- Start up with missing modules is not supported.
- In the application program it is possible to detect if a hot swap terminal unit is mounted in a specific position on the I/O bus. The information is available in the process data area or can be read out via acyclic read. These can be accessed when the version of the communication interface module supports hot swap.
- This has to be checked by application: Best way for checking if a hot swap terminal unit is available or not, is checking the corresponding information inside the process image.

\subsection*{6.8.4.3 PROFIBUS communication interface module}

\subsection*{6.8.4.3.1 Hot swap}

\section*{Introduction}

With hot swap for AC500 and S500 it is possible to exchange I/O modules (with same type) during runtime.

\section*{Preconditions for using hot swap}

\section*{Hot swap}

\section*{WARNING!}

Risk of explosion or fire in hazardous environments during hot swapping!
Hot swap must not be performed in flammable environments to avoid life-threatening injury and property damage resulting from fire or explosion.

\section*{WARNING!}

Electric shock due to negligent behavior during hot swapping!
To avoid electric shock
- make sure the following conditions apply:
- Digital outputs are not under load.
- Input/output voltages above safety extra low voltage/ protective extra low voltage (SELV/PELV) are switched off.
- Modules are fully interlocked with the terminal unit with both snap-fits engaged before switching on loads or input/output voltage.
- Never touch exposed contacts (dangerous voltages).
- Stay away from electrical contacts to avoid arc discharge.
- Do not operate a mechanical installation improperly.

\section*{NOTICE!}

\section*{Risk of damage to I/O modules!}

Hot swapping is only allowed for I/O modules.
Processor modules and communication interface modules must not be removed or inserted during operation.

H = Hot swap

\section*{Hot swap}

System requirements for hot swapping of I/O modules:
- Types of terminal units that support hot swapping of I/O modules have the appendix TU5xx-H.
- I/O modules as of index FO.

The following I/O bus masters support hot swapping of attached I/O modules:
- Communication interface modules CI5xx as of index F0.
- Processor modules PM56xx-2ETH with firmware version as of V3.2.0.

Hot swap is not supported by AC500-eCo V3 CPU!


The index of the module is in the right corner of the label.

\section*{NOTICE!}

Risk of damage to I/O modules!
Modules with index below F0 can be damaged when inserted or removed from the terminal unit in a powered system.

\section*{NOTICE!}

Risk of damage to I/O modules!
Do not perform hot swapping if any I/O module with firmware version lower than 3.0.14 is part of the I/O configuration.

For min. required device index see table below.
\begin{tabular}{|l|l|}
\hline Device & \begin{tabular}{l} 
Min. required device index for I/O module as of \\
FW Version 3.0.14
\end{tabular} \\
\hline AC522(-XC) & F0 \\
\hline AI523 (-XC) & D2 \\
\hline AI531 & D4 \\
\hline AI531-XC & D2 \\
\hline AI561 & B2 \\
\hline AI562 & B2 \\
\hline AI563 & B3 \\
\hline AO523 (-XC) & D2 \\
\hline AO561 & B2 \\
\hline AX521 (-XC) & D2 \\
\hline AX522 (-XC) & D2 \\
\hline AX561 & B2 \\
\hline CD522 (-XC) & D1 \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline Device & \begin{tabular}{l} 
Min. required device index for I/O module as of \\
FW Version 3.0.14
\end{tabular} \\
\hline DA501 (-XC) & D2 \\
\hline DA502 (-XC) & F0 \\
\hline DC522 (-XC) & D2 \\
\hline DC523 (-XC) & D2 \\
\hline DC532 (-XC) & D2 \\
\hline DC562 & A2 \\
\hline DI524 (-XC) & D2 \\
\hline DI561 & B2 \\
\hline DI562 & B2 \\
\hline DI571 & B2 \\
\hline DI572 & A1 \\
\hline DO524 (-XC) & A3 \\
\hline DO526 & A2 \\
\hline DO526-XC & A0 \\
\hline DO561 & B2 \\
\hline DO562 & A2 \\
\hline DO571 & B3 \\
\hline DO572 & B2 \\
\hline DO573 & A1 \\
\hline DX522 (-XC) & D2 \\
\hline DX531 & D2 \\
\hline DX561 & B2 \\
\hline DX571 & A1 \\
\hline FM562 & \\
\hline
\end{tabular}

\section*{Cl54x-DP, index >= F1}

To support the hot swap feature for PROFIBUS below V3 PLCs, new slaves are available for the CI54x-DP - Cl541-DP, index >= F1 and the CI542-DP, index >=F1.
To be able to configure and to use the hot swap functionalities for the S 500 modules below PROFIBUS of V3 PLCs, these slaves have to be configured. A double-click on a device object in the device tree opens the associated device editor. The editor provides generic and devicespecific tabs for the device configuration.
Copy and paste of I/O modules between the CI54x-DP and the CI54x-DP, index \(>=F 1\) is possible.
Drag and drop between the \(\mathrm{Cl} 54 \mathrm{x}-\mathrm{DP}\) and the \(\mathrm{CI} 54 \mathrm{x}-\mathrm{DP}\), index \(>=\mathrm{F} 1\) is not possible .

If the I/O modules must be moved from one CI54x-DP to the other one, cut and past must be executed. Drag and drop is not possible.

\subsection*{6.9 Diagnosis and debugging}

\subsection*{6.9.1 The diagnosis system}

\subsection*{6.9.1.1 General}

The diagnosis system enables uniform diagnosis of the CPU and its local interfaces, of the I/O bus with the connected S500 I/O devices and of the fieldbuses connected via communication modules, considering the special features of the various fieldbuses. The safety CPU is also integrated into the diagnosis system.

Diagnosis data of the devices can be accessed by
- CPU display
- Automation Builder
- IEC application

To forward the information to notify them by, e.g., webserver or OPC UA server, the data retrieved in IEC application can be stored in variables.


Fig. 364: Overview of the diagnosis system
All diagnosis data is assigned to a device. System diagnosis (e.g., battery low) is coming from the CPU device or one of the child objects (e.g., watchdog diagnosis of a task object).
Diagnosis is available for devices with representation in the Automation Builder device tree.
Diagnosis messages include the severity of an error. Error severity can be used for defining system behavior, e.g., activating the error LED or stop the PLC \(\Leftrightarrow\) Chapter 6.9.1.6.1 "Error severity" on page 4638.

Details on how to integrate and use function blocks to receive diagnosis messages from the CPU and fieldbus devices are given in the application examples:
- AC500 V3 diagnosis in IEC application
- AC500 V3 diagnosis

Types of diagnosis messages
- An event describes the current state of the device. It does not have to be acknowledged.
- An alarm describes that at a certain point of time, there was a diagnosis message. It does not say anything about the current state of the device.
Alarms must be acknowledged by the user. After acknowledging, the alarm disappears for all consumers.

Every diagnosis message has a come time.

\section*{Device state With reference to diagnosis, there are different device states:}
- Device without events and without unacknowledged alarms.
- Device with events or unacknowledged alarms.
- Device does not respond and is not available for online connection.

Diagnosis Diagnosis messages are always available for all consumers.
descriptions
- ABB AC500 V3 devices:
- Events and unacknowledged alarms.
- Every diagnosis message with come time, location, error number and text.
- \(3^{\text {rd }}\) party devices:
- Events and unacknowledged alarms.
- Every diagnosis message with come time, location and error number.
- Clear text information if available either from a standard or from the device description.
- If available: Extended diagnosis: Additional data coming from the device for manual analysis.

Extended diag- Some devices are able to provide extended diagnosis. This additional device-dependant diagnosis nosis will only be collected on request and will be device type specific (e.g. bus scan request on PROFINET I/O controller). Main intention is to cover commissioning use cases, when very specific information is required that typically cannot be stored in error numbers in a reasonable way.

\subsection*{6.9.1.2 Access to diagnosis data}

Access to device state
- Error LED on CPU \({ }^{\star}\) Chapter 6.9.1.3 "Diagnosis in CPU display" on page 4607
- Automation Builder device tree \({ }^{\wedge}>\) Chapter 6.9.1.4 "Diagnosis in Automation Builder" on page 4611
- IEC application via device name \(\Leftrightarrow>\) Chapter 6.9.1.5.4 "Device diagnosis" on page 4628
- IEC application via list of all available diagnosis \({ }^{*} \Rightarrow\) Chapter 6.9.1.5.3 "System diagnosis" on page 4619
- External access via global IEC variables Fig. 364

Access to diagnosis descriptions
- CPU display \({ }^{\circledR}\) Chapter 6.9.1.3.2 "Diagnosis descriptions" on page 4607 (for CPU, I/O bus and connected S500 I/O modules, not for communication modules and field buses)
- Automation Builder via "All messages" window \(\stackrel{y}{ }\) Chapter 6.9.1.4.2 "Diagnosis descriptions" on page 4611:
- Support for (bulk) acknowledgement of alarm
- Access to extended diagnosis data of \(3^{\text {rd }}\) party devices devices (if available), without any interpretation
- IEC application via a list of all current diagnosis either of a device (device object from the Automation Builder tree) \(\stackrel{y}{ }{ }^{\circ}\) Chapter 6.9.1.5.4 "Device diagnosis" on page 4628 or of the complete PLC \(\Longleftrightarrow\) Chapter 6.9.1.5.3 "System diagnosis" on page 4619:
- Navigation chronologically in both directions (starting either from the oldest or newest diagnosis)
- Access to the diagnosis numerically (evaluation by IEC application), textual (use on HMI ) or to extended diagnosis data of \(3^{\text {rd }}\) party devices (if available)
- Acknowledgement of alarms
- External access via global IEC variables by using the IEC application features for getting all relevant information

Access to Extended diagnosis data will be displayed in Automation Builder via "All messages" window.
extended diag-
nosis

The data is displayed as it is provided from the device without any interpretation. Refer to e.g., the manual of the device to get information about the extended diagnosis data.

\subsection*{6.9.1.3 Diagnosis in CPU display}

\subsection*{6.9.1.3.1 Device state}

If there is at least one active diagnosis message, the error LED ERR is on.
The behavior of the error LED depends on the setting of CPU parameter "Error LED" « Table on page 1474.


General information on the LEDs, the display and the function keys can be found in chapter 'Displays, LEDs and function keys on the front panel' \(\Longleftrightarrow\) Chapter 6.8.2.5 "Display, LEDs and function keys on the front panel" on page 4452.

\subsection*{6.9.1.3.2 Diagnosis descriptions}

Chapter 6.9.1.3.3 "DIAG - reading out diagnosis messages" on page 4609.
\begin{tabular}{|l|c|c|l|l|}
\hline Entry & \begin{tabular}{c} 
Length \\
[byte]
\end{tabular} & Values & Description & Display \\
\hline Error severity & 1 & \(0 . .255\) & \begin{tabular}{l} 
Used values: 1, 2, 3, 4, 11 « Chapter \\
6.9 .1 .6 .1 "Error severity" on page 4638
\end{tabular} & Ex abc \\
\hline Hardware ID (HwId) & 1 & \(0 . .255\) & \begin{tabular}{l} 
Location of diagnosis, e.g., subdevice, as \\
three-letter word \\
on page 4608 Further information
\end{tabular} & Ex abc \\
\hline Error code & 2 & \(1 . .65535\) & Error number (low word) & 12345 \\
\hline SubSysteminfo byte 1 & 1 & \(0 . .255\) & Depends on hardware ID & d1 123 \\
\hline SubSysteminfo byte 2 & 1 & \(0 . .255\) & Depends on hardware ID & d2 123 \\
\hline SubSysteminfo byte 3 & 1 & \(0 . .255\) & Depends on hardware ID & d3 123 \\
\hline SubSysteminfo byte 4 & 1 & \(0 . .255\) & Depends on hardware ID & d4 123 \\
\hline
\end{tabular}

The display of the processor module does not show any communication modules or fieldbus diagnosis. To view these diagnosis messages use Automation Builder or IEC application.

Chapter 6.9.1.4 "Diagnosis in Automation Builder" on page 4611
« Chapter 6.9.1.5 "Diagnosis in IEC application" on page 4615
This is valid for:
- all external communication modules including safety processor module
- CANopen on onboard CAN interface
- fieldbuses on Ethernet interfaces ETH1/ETH2 like PROFINET IO controller, EtherCAT master, etc.

The location of a diagnosis is shown in the CPU display with 3 characters.
\begin{tabular}{|c|c|}
\hline Display & Location of diagnosis \\
\hline (\%) [PU & CPU \\
\hline  & CPU RAM \\
\hline  & CPU flash \\
\hline  & CPU flashdisk \\
\hline  & CPU memory card \\
\hline  & CPU display \\
\hline  & CPU battery \\
\hline  & Real-time clock (RTC) \\
\hline  & Floating point unit (FPU) \\
\hline  & Power supply \\
\hline  & Communication module 1 \\
\hline  & Communication module 2 \\
\hline  & Communication module 3 \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline Display & Location of diagnosis \\
\hline  & Communication module 4 \\
\hline  & Communication module 5 \\
\hline  & Communication module 6 \\
\hline  & COM1 serial interface 1 \\
\hline  & COM2 reserved for serial interface 2 \\
\hline  & CAN interface \\
\hline  & I/O bus \\
\hline  & Ethernet ETH1 \\
\hline  & Ethernet ETH2 \\
\hline
\end{tabular}

\subsection*{6.9.1.3.3 DIAG - reading out diagnosis messages}

Table 853: Example - no diagnosis message in state list
\begin{tabular}{|l|l|l|l|l|l|l|l|}
\hline State & Display & \multicolumn{4}{|l|}{ Result on pressing one of the function keys } \\
\cline { 2 - 6 } & & DIAG & & & & \\
\hline 0 & \begin{tabular}{l} 
The processor \\
module is in \\
RUN/STOP \\
mode.
\end{tabular} & \begin{tabular}{l} 
State 1 is \\
displayed.
\end{tabular} & - & - & - & \\
\hline 1 & ESC & & & \\
\hline
\end{tabular}

Table 854: Example - diagnosis messages in state list
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{State} & \multirow[t]{2}{*}{Display} & \multicolumn{5}{|l|}{Result on pressing one of the function keys} \\
\hline & & DIAG &  &  & ESC & OK \\
\hline 0 & The processor module is in RUN/STOP mode. & State 1 is displayed. & & - & & - \\
\hline 1 & \begin{tabular}{l}
…Err 4 \\
Number of diagnosis messages; here 4
\end{tabular} & & Go to first/ next diagnosis message in state list (e.g., state 2). & Go to last/ previous diagnosis message in state list. & Return into RUN/STOP mode. & Return into RUN/STOP mode. \\
\hline 2 & \begin{tabular}{l}
- . E4 bRt \(=\) \\
Diagnosis message example: Error battery empty or missing. \\
Toggling between state 2 and 3.
\end{tabular} & \multirow[t]{2}{*}{Selects displayed diagnosis message and shows details. * Table 8 55 "Exam ple - error battery empty or missing" on page 461} & \multirow[t]{2}{*}{Go to first/ next diagnosis message in state list.} & \multirow[t]{2}{*}{Go to last/ previous diagnosis message in state list.} & \multirow[t]{2}{*}{Return into RUN/STOP mode.} & \multirow[t]{2}{*}{Acknowledge and return into RUN/STOP mode.} \\
\hline 3 & \begin{tabular}{l}
ㄷ. id 1= \\
Error ID example \\
Toggling between state 2 and 3
\end{tabular} & & & & & \\
\hline
\end{tabular}

Table 855: Example - error battery empty or missing
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{State} & \multirow[t]{2}{*}{Display} & \multicolumn{5}{|l|}{Result on pressing one of the function keys} \\
\hline & & DIAG & \(\square\) & \[
4
\] & ESC & OK \\
\hline 0 & E4 = error
E4 bRt
severity 4
bAt = subdevice
battery
Toggling
between state 0
and 1 & \multirow[t]{2}{*}{State 2 is displayed.} & \multirow[t]{2}{*}{State 2 is displayed.} & \multirow[t]{2}{*}{State 6 is displayed.} & \multirow[t]{2}{*}{State 0 is displayed. Return to diagnosis state list.} & \multirow[t]{2}{*}{State 0 is displayed. Return to diagnosis state list.} \\
\hline 1 & \begin{tabular}{l}
Error ID example \\
Toggling between state 0 and 1.
\end{tabular} & & & & & \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{State} & \multirow[t]{2}{*}{Display} & \multicolumn{5}{|l|}{Result on pressing one of the function keys} \\
\hline & & DIAG &  &  & ESC & \[
\mathrm{OK}
\] \\
\hline 2 & \begin{tabular}{l}
 \\
Error number 8 \\
Battery is missing or empty.
\end{tabular} & & State 3 is displayed. & State 0 is displayed. & \begin{tabular}{l}
State 0 is displayed. \\
Return to diagnosis state list.
\end{tabular} & \begin{tabular}{l}
State 0 is displayed. \\
Displays state 0 . \\
Return to diagnosis state list.
\end{tabular} \\
\hline 3 & \begin{tabular}{l} 
事: dl 22 \\
Detail 1 \\
\begin{tabular}{l} 
Subdevice 22: \\
battery
\end{tabular} \\
\hline
\end{tabular} & & State 4 is displayed. & State 2 is displayed. & \begin{tabular}{l}
State 0 is displayed. \\
Return to diagnosis state list.
\end{tabular} & \begin{tabular}{l}
State 0 is displayed. \\
Return to diagnosis state list.
\end{tabular} \\
\hline 4 & \begin{tabular}{l}
는:d己 0 를 \\
Detail 2 \\
Error type 0: device
\end{tabular} & & State 5 is displayed. & State 3 is displayed. & \begin{tabular}{l}
State 0 is displayed. \\
Return to diagnosis state list.
\end{tabular} & \begin{tabular}{l}
State 0 is displayed. \\
Return to diagnosis state list.
\end{tabular} \\
\hline 5 &  & & State 6 is displayed. & State 4 is displayed. & \begin{tabular}{l}
State 0 is displayed. \\
Return to diagnosis state list.
\end{tabular} & State 0 is displayed Return to diagnosis state list. \\
\hline 6 & \begin{tabular}{l}
문․ 14 플 \\
Detail 4 \\
Additional information 0: none
\end{tabular} & & State 1 is displayed. & State 5 is displayed. & \begin{tabular}{l}
State 0 is displayed. \\
Return to diagnosis state list.
\end{tabular} & \begin{tabular}{l}
State 0 is displayed. \\
Return to diagnosis state list.
\end{tabular} \\
\hline
\end{tabular}

\subsection*{6.9.1.4 Diagnosis in Automation Builder}

\subsection*{6.9.1.4.1 Device state}

In Automation Builder, colored icons next to the devices' nodes in the device tree indicate the device state of each single device \(\gg\) Chapter 6.9.2.3 "Project tree in online mode" on page 4641.

\subsection*{6.9.1.4.2 Diagnosis descriptions}

Displayed text For output of diagnosis messages in textual format Automation Builder and IEC application use text lists. Both application use the same text lists. The text lists are part of the device description. When inserting a new device in device tree of project, the corresponding text list is loaded. This text lists are part of PLC program and will be downloaded into the PLC.
\begin{tabular}{|c|c|c|c|}
\hline Devices \(\quad 7 \times\) & 目 PLC＿PRG & nager Diag＿V3＿PLC \(\times\) & \\
\hline \multirow[t]{24}{*}{\begin{tabular}{l}
fill Lbrary Manager \\
目 PLC＿PRG（PRG） \\
－Task Configuration \\
＊ 21 Visualization Manager Visualization \\
＝IO＿Bus
DC532（DC532） \\
DI524（DI524）
\(\square\) DO526（DO526） \\
\(+\boldsymbol{H}_{\square}\) Interfaces \\
－明：Extension＿Bus
CM579＿PNIO（CM579．PNIO） \\
－ 4 ．f PNIO＿Controller（PROFINET－IO－Controler） \\
\(=15\) CI501＿PNIO＿Device（CI501－PNIO－Device） \\
（3）CI501＿IO（CI501－IO） \\
Slot＿2（＜Empty＞）
\end{tabular}} & ID & Default & en \\
\hline & Err＿10 & Medium has almost used its ．．． & Medium has almost used its ．．． \\
\hline & Err＿17 & Error setting parameter & Error setting parameter \\
\hline & Err＿18 & At least one not found & At least one not found \\
\hline & Err＿19 & Unable to read at least one ．．． & Unable to read at least one ．．． \\
\hline & Err＿2 & Resource failure & Resource failure \\
\hline & Err＿20 & Invalid value of＂error LED＂ & Invalid value of＂error LED＂ \\
\hline & Err＿21 & Invalid value of＂check batte．．． & Invalid value of＂check batte．．． \\
\hline & Err＿22 & Error at initialization & Error at initialization \\
\hline & Err＿23 & Wrong version of firmware & Wrong version of firmware \\
\hline & Err＿3 & Timeout & Timeout \\
\hline & Err＿8 & Empty or missing & Empty or missing \\
\hline & Err＿9 & Medium has used \(80 \%\) of its．．． & Medium has used \(80 \%\) of its．．． \\
\hline & Hwd＿0 & CPU & CPU \\
\hline & Sub1＿0 & CPU & CPU \\
\hline & Sub1＿14 & I／O－Bus & I／O－Bus \\
\hline & Sub1＿19 & Flashdisk & Flashdisk \\
\hline & Sub1＿21 & Display & Display \\
\hline & Sub1＿22 & Battery & Battery \\
\hline & Sub2＿0 & Device & Device \\
\hline & Sub2＿2 & Parameter & Parameter \\
\hline & Sub2＿5 & Index & Index \\
\hline & Sub4＿1 & too big & too big \\
\hline & Sub4＿2 & too small & too small \\
\hline
\end{tabular}

It is necessary to include a visualization，even if visualization will not be used． \({ }^{〔}\) Chapter 6．4．5＂CODESYS Visualization＂on page 2852

Without visualization the text lists will not be included．

Displayed text The text lists for \(3^{\text {rd }}\) party devices are created during reading of the device description sheets， for \(3^{\text {rd }}\) party devices e．g．，GSDML files for PROFINET I／O devices．

The text lists are generated automatically．We recommend that you do not change them manually because the changes can be overwritten automatically and without prompting．
Example \(\quad\)\begin{tabular}{l} 
CI501－PNIO：Diag＿PNIO＿26＿22 \\
\(26=\) vendor ID ABB， \(22=\) device ID CI501－PNIO
\end{tabular}

The text list for the AC500 PROFINET I／O modules contains all text needed for PROFINET standard diagnosis and AC500 process alarm handling．
Which texts are used，depends on parameter＂Selection of diagnosis method＂：Double－click on a PROFINET I／O module and open tab＂General＂．

\section*{6．9．1．4．3 System diagnosis}

In Automation Builder the system diagnosis is activated by default and can be deactivated in：
＂Tools \(\rightarrow\) Options \(\rightarrow\) Diagnosis \(\rightarrow\) Enable subtree diagnosis＂


\section*{6．9．1．4．4 Device diagnosis}

Each node in the device tree has a diagnosis view，which displays the diagnosis messages for this device only．
The message consists of：
－Type \({ }^{\circ}\)－Chapter 6．9．1．4．5＂Diagnosis history＂on page 4614
－Timestamp in date and time YYYY－MM－DD hh：mm：ss．ms
－Error severity
－Error code
－Diagnosis description
－Additional data
To view the diagnosis message：
1．Double－click on a device．
2．Select the tab＂Diagnosis＂．
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline \multirow[t]{2}{*}{Example} & \multicolumn{6}{|l|}{Battery empty or missing．} \\
\hline & \multicolumn{6}{|l|}{ius 戒 PLC＿AC500＿V3 x} \\
\hline & \multicolumn{6}{|l|}{Start refresh Stop refresh Acknowledge Selected Alarms} \\
\hline & Type & Device & Timestamp & Severity & Error Code & Description \\
\hline & ＋ & PLC＿AC500＿V3 & 1970－01－01；12：00：13．916 & 4 & 8 & Battery，Device，Empty or missing \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline \multirow[t]{5}{*}{Example} & \multicolumn{7}{|l|}{Wrong module configured on I／O bus．} \\
\hline & \multicolumn{7}{|l|}{；Y㧫 PLC＿AC500＿V3 比 DI524 x} \\
\hline & \multicolumn{7}{|l|}{Start refresh Stop refresh Acknowledge Selected Alarms} \\
\hline & Type & Device & Timestamp & Severity & Error Code & Description & Additi \\
\hline & ＋ & DI524 & 1970－01－01；12：16：30．633 & 11 & 16158 & Module 1，Ty & \\
\hline
\end{tabular}

Device diagnosis is disabled by default．
To enable／disable device diagnosis：
1．Double－click on the PLC．
2．Select the tab＂PLC Settings＂．

3．Under＂Additional Settings＂enable／disable＂Diagnosis for devices＂．
\(\Rightarrow\) When the device diganosis is disabled，this symbol will be displayed in the device tree and no diagnosis messages will be shown．

\section*{6．9．1．4．5 Diagnosis history}

The＇Diagnosis History＇view provides an overview of the current and past system events that resulted in a diagnosis event．
－Incoming diagnosis events are indicated with＋．
After the problem that causes a diagnosis event has been resolved，this diagnosis event is indicated automatically with－．
－Alarm events，e．g．PROFINET alarm are indicated with 7.
In the＇Diagnosis＇view the user can acknowledge an alarm．Note that an alarm event can be acknowledged though the problem that causes the alarm still persists．
The acknowledge action is indicated with \(\checkmark\) on the concerning event entry．If the icon changes to \(\checkmark\) ，the acknowledge action has been completed by the PLC．

The following buttons are available in the＇Diagnosis History＇view：
－Start／Stop refresh：
Enables or disables the automatic refresh mode．In refresh mode new diagnosis events will be displayed automatically．Only the last 100 entries are shown in this view，the latest events on top of the list．
－Get next entries：
Adds the previous（older） 100 diagnosis events at the bottom of the list．
－Export complete history：
Creates a csv file with all events from the diagnosis history（not only the visible ones）．
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline File Edit View Project Build Online & Debug Iools Window & & & & & & \\
\hline \multicolumn{8}{|l|}{} \\
\hline Devices－\({ }^{4}\) & \multicolumn{7}{|l|}{} \\
\hline \multirow[t]{39}{*}{} & \multirow[t]{2}{*}{\begin{tabular}{l}
Communication Settings \\
PLC Settings
\end{tabular}} & \multicolumn{6}{|l|}{Stop refresh Get next entries Export complete history} \\
\hline & & Type & Device \(\square\) & Timestamp & Severity & Error Code & Description \\
\hline & \multirow[t]{2}{*}{Version information} & － & PLC＿AC500＿V3 & 01／01／1970 12：06：11，364 4 & 4 & 9 & Battery，Device，Medium has used 80\％of its spare capacity \\
\hline & & ＋ & PLC＿AC500＿V3 & 01／01／1970 12：06：07，164 4 & 4 & 9 & Battery，Device，Medium has used \(80 \%\) of its spare capacity \\
\hline & \multirow[t]{2}{*}{Statistics} & ＋ & IO＿Bus & 01／01／1970 12：06：05，564 4 & 4 & 16159 & Module 22，Configured number of modules differs from found ones \\
\hline & & － & PLC＿AC500＿V3 & 01／01／1970 12：06：01，164 4 & 4 & 9 & Battery，Device，Medium has used \(80 \%\) of its spare capacity \\
\hline & & － & IO＿Bus & 01／01／1970 12：05：59，764 4 & 4 & 16159 & Module 22，Configured number of modules differs from found ones \\
\hline & \multirow[t]{2}{*}{Log} & \(\checkmark\) & IO＿Bus & 01／01／1970 12：05：59，164 4 & 4 & 123 & Module 22，Diagnosis text not available \\
\hline & & ＋ & PLC＿AC500＿V3 & 01／01／1970 12：05：57，165 4 & 4 & 9 & Battery，Device，Medium has used \(80 \%\) of its spare capacity \\
\hline & \multirow[t]{2}{*}{PLC Shell} & ＋ & IO＿Bus & 01／01／1970 12：05：55，164 4 & 4 & 16159 & Module 22，Configured number of modules differs from found ones \\
\hline & & 4 & IO＿Bus & 01／01／1970 12：05：53，964 4 & 4 & 123 & Module 22，Diagnosis text not available \\
\hline & Users and Groups & \(\checkmark\) & IO＿Bus & 01／01／1970 12：05：53，565 4 & 4 & 123 & Module 22，Diagnosis text not available \\
\hline & \multirow[t]{2}{*}{Access Rights} & － & PLC＿AC500＿V3 & 01／01／1970 12：05：50，964 4 & 4 & 9 & Battery，Device，Medium has used \(80 \%\) of its spare capacity \\
\hline & & － & IO＿Bus & 01／01／1970 12：05：49，165 4 & 4 & 16159 & Module 22，Configured number of modules differs from found ones \\
\hline & \multirow[t]{2}{*}{Symbol Rights} & ＋ & PLC＿AC500＿V3 & 01／01／1970 12：05：47，164 4 & 4 & 9 & Battery，Device，Medium has used \(80 \%\) of its spare capacity \\
\hline & & ＋ & IO＿Bus & 01／01／1970 12：05：44，764 4 & 4 & 16159 & Module 22，Configured number of modules differs from found ones \\
\hline & PM5670－2ETH Hardware & 4 & IO＿Bus & 01／01／1970 12：05：43，164 4 & 4 & 123 & Module 22，Diagnosis text not available \\
\hline & \multirow[t]{2}{*}{CPU－Parameters Parameters} & － & PLC＿AC500＿V3 & 01／01／1970 12：05：40，764 4 & 4 & 9 & Battery，Device，Medium has used \(80 \%\) of its spare capacity \\
\hline & & ＋ & IO＿Bus & 01／01／1970 12：05：39，404 4 & 4 & 16159 & Module 22，Configured number of modules differs from found ones \\
\hline & \multirow[t]{2}{*}{IEC objects} & ＋ & PLC＿AC500＿V3 & 01／01／1970 12：05：36，804 4 & 4 & 9 & Battery，Device，Medium has used \(80 \%\) of its spare capacity \\
\hline & & － & IO＿Bus & 01／01／1970 12：05：35，104 4 & 4 & 16159 & Module 22，Configured number of modules differs from found ones \\
\hline & 10 Mapping List & － & PLC＿AC500＿V3 & 01／01／1970 12：05：33，104 4 & 4 & 9 & Battery，Device，Medium has used \(80 \%\) of its spare capacity \\
\hline & \multirow[t]{2}{*}{I／O－Bus I／C Mapping} & \(\checkmark\) & IO＿Bus & 01／01／1970 12：05：31，004 4 & 4 & 123 & Module 22，Diagnosis text not available \\
\hline & & 7 & IO＿Bus & 01／01／1970 12：05：30，404 4 & 4 & 123 & Module 22，Diagnosis text not available \\
\hline & Task Deployment & ＋ & IO＿Bus & 01／01／1970 12：05：29，004 4 & 4 & 16159 & Module 22，Configured number of modules differs from found ones \\
\hline & \multirow[t]{2}{*}{Applications} & ＋ & PLC＿AC500＿V3 & 01／01／1970 12：05：26，804 4 & 4 & 9 & Battery，Device，Medium has used \(80 \%\) of its spare capacity \\
\hline & & － & IO＿Bus & 01／01／1970 12：05：24，505 4 & 4 & 16159 & Module 22，Configured number of modules differs from found ones \\
\hline & Backup and Restore & － & PLC＿AC500＿V3 & 01／01／1970 12：05：22，904 4 & 4 & 9 & Battery，Device，Medium has used \(80 \%\) of its spare capacity \\
\hline & \multirow[b]{2}{*}{Status} & \(\checkmark\) & IO＿Bus & 01／01／1970 12：05：19，804 4 & 4 & 123 & Module 22，Diagnosis text not available \\
\hline & & 4 & IO＿Bus & 01／01／1970 12：05：19，604 4 & 4 & 123 & Module 22，Diagnosis text not available \\
\hline & \multirow[t]{2}{*}{Diagnosis} & ＋ & IO＿Bus & 01／01／1970 12：05：18，604 4 & 4 & 16159 & Module 22，Configured number of modules differs from found ones \\
\hline & & ＋ & PLC＿AC500＿V3 & 01／01／1970 12：05：16，804 4 & 4 & 9 & Battery，Device，Medium has used \(80 \%\) of its spare capacity \\
\hline & Diagnosis History & － & IO＿Bus & 01／01／1970 12：05：13，904 4 & 4 & 16159 & Module 22，Configured number of modules differs from found ones \\
\hline & \multirow[t]{3}{*}{Information} & － & PLC＿AC500＿V3 & 01／01／1970 12：05：12，704 4 & 4 & 9 & Battery，Device，Medium has used \(80 \%\) of its spare capacity \\
\hline & & \(\checkmark\) & IO＿Bus & 01／01／1970 12：05：08，605 4 & 4 & 123 & Module 22，Diagnosis text not available \\
\hline & & ＋ & IO＿Bus & 01／01／1970 12：05：08，204 4 & 4 & 16159 & Module 22，Configured number of modules differs from found ones \\
\hline & & ＋ & PLC＿AC500＿V3 & 01／01／1970 12：05：06，804 4 & 4 & 9 & Battery，Device，Medium has used \(80 \%\) of its spare capacity \\
\hline & & \(\checkmark\) & IO＿Bus & 01／01／1970 12：05：03，004 4 & 4 & 123 & Module 22，Diagnosis text not available \\
\hline & & － & PLC＿AC500＿V3 & 01／01／1970 12：05：02，504 4 & 4 & 9 & Battery，Device，Medium has used \(80 \%\) of its spare capacity \\
\hline \multirow[t]{2}{*}{\＄Devices \(\square^{\text {Pous }}\)} & & 4 & IO＿Bus & 01／01／1970 12：04：58，004 4 & 4 & 123 & Module 22，Diagnosis text not available \\
\hline & & & & vice user：Anonymous & Last b & uld： 00 © & 0 Precomple \(\checkmark\) R RUN Progr \\
\hline
\end{tabular}

\section*{6．9．1．5 Diagnosis in IEC application}

\section*{6．9．1．5．1 General}

There are two possibilities for accessing the diagnosis messages in the IEC application：
－System diagnosis：Access to diagnosis messages of the whole PLC
－Device diagnosis：Access to the diagnosis messages of a device
For both possibilities common data types（structures and enumerations）are defined in the library AC500＿DiagTypes \(\stackrel{\text { \＆Chapter 6．9．1．5．2＂Data types in library AC500＿DiagTypes＂}}{ }\) on page 4616．The library is automatically included in PLC project．

Details on how to integrate and use function blocks to receive diagnosis messages from the CPU and fieldbus devices are given in the application examples：
－AC500 V3 diagnosis in IEC application
－AC500 V3 diagnosis

Displayed text For output of diagnosis messages in textual format Automation Builder and IEC application use text lists．Both application use the same text lists．The text lists are part of the device description．When inserting a new device in device tree of project，the corresponding text list is loaded．This text lists are part of PLC program and will be downloaded into the PLC．
\begin{tabular}{|c|c|c|c|}
\hline Devices \(\quad 7 \times\) & ［國 PLC＿PRG & ［1i］Lbrary Manager Diag＿V3＿PLC \(\times\) & \\
\hline ```
- #1 PLC_AC500_V3 (PM5670-2ETH - TB5620-2ETH)
``` & ID & Default & en \\
\hline  & Err＿10 & Medium has almost used its ．．． & Medium has almost used its ．．． \\
\hline \(\square\) Diagnosis & Err＿17 & Error setting parameter & Error setting parameter \\
\hline Til］Dia & Err＿18 & At least one not found & At least one not found \\
\hline 7 Diag pNiO 26.22 & Err＿19 & Unable to read at least one ．．． & Unable to read at least one ．．． \\
\hline  & Err＿2 & Resource failure & Resource failure \\
\hline  & Err＿20 & Invalid value of＂error LED＂ & Invalid value of＂error LED＂ \\
\hline Diag＿V3 PLC & Err＿21 & Invalid value of＂check batte．．． & Invalid value of＂check batte．．． \\
\hline  & Err＿22 & Error at initialization & Error at initialization \\
\hline PIC PRG（PRG）E & Err＿23 & Wrong version of firmware & Wrong version of firmware \\
\hline & Err＿3 & Timeout & Timeout \\
\hline  & Err＿8 & Empty or missing & Empty or missing \\
\hline Visualization & Err＿9 & Medium has used \(80 \%\) of its．．． & Medium has used \(80 \%\) of its．．． \\
\hline －\(T\) to Bus & Hwd＿0 & CPU & CPU \\
\hline & Sub1＿0 & CPU & CPU \\
\hline DC532（DC532） & Sub1＿14 & I／O－Bus & 1／O－Bus \\
\hline D0526（D0526） & Sub1＿19 & Flashdisk & Flashdisk \\
\hline  & Sub1＿21 & Display & Display \\
\hline －Extension Bus & Sub1＿22 & Battery & Battery \\
\hline －日里 CM579 PNTO（CM579．PNIO） & Sub2＿0 & Device & Device \\
\hline －An PNTO Controller（PROFINET－IO－Controller） & Sub2＿2 & Parameter & Parameter \\
\hline 1 ：CI501 PNTO Device（CI501－PNTO－Device） & Sub2＿5 & Index & Index \\
\hline \[
\text { L } 31 \text { CI501_IO (CI501-IO) }
\] & Sub4＿1 & too big & too big \\
\hline ¢ Slot＿2（＜Empty＞） & Sub4＿2 & too small & too small \\
\hline
\end{tabular}

It is necessary to include a visualization，even if visualization will not be used． ② Chapter 6．4．5＂CODESYS Visualization＂on page 2852

Without visualization the text lists will not be included．

The text lists are generated automatically．We recommend that you do not change them manually because the changes can be overwritten automatically and without prompting．

Displayed text The text lists for \(3^{\text {rd }}\) party devices are created during reading of the device description sheets, for \(3^{\text {rd }}\) party devices e.g., GSDML files for PROFINET I/O devices.

The name of a text list for a PROFINET I/O device is: Diag_PNIO_Vendor ID_Device ID
```

Example CI501-PNIO: Diag_PNIO_26_22
26 = vendor ID ABB, 22 = device ID CI501-PNIO

```

The text list for the AC500 PROFINET I/O modules contains all text needed for PROFINET standard diagnosis and AC500 process alarm handling.
Which texts are used, depends on parameter "Selection of diagnosis method": Double-click on a PROFINET I/O module and open tab "General".

\subsection*{6.9.1.5.2 Data types in library AC500_DiagTypes}

\section*{General}

All data types regarding diagnosis are defined in the library AC500_DiagTypes.

\section*{Structure DIAG_VAL_TYPE}

This data type specifies the format of all kinds of diagnosis messages in numeric format. It consists of one element for each detail of a diagnosis message.
\begin{tabular}{|l|l|l|l|}
\hline Name & Type & Initial & Comment \\
\hline diTimestamp & DT & DATE_AND_TIME\#1970-1-1-0:0 & RTC time of eDiagEvent_Occured \\
\hline uiMs & UINT & 0 & Milliseconds of event \\
\hline eClass & teClass & teClass.eDiagClass_4_Warning & Error severity of diagnosis message \\
\hline szDevice & STRING(80) & "" & Name of device, max. 80 characters \\
\hline eHwInterfaceld & teHwId & teHwId_CPU & Identifier of hardware interface \\
\hline dwSubSysteminfo & DWORD & 0 & \begin{tabular}{l} 
Any number describing details or loca- \\
tion within device, device-specific
\end{tabular} \\
\hline dwAdditional & DWORD & 0 & \begin{tabular}{l} 
Additional number describing details or \\
location within device, optional, device- \\
specific
\end{tabular} \\
\hline dwErrorCode & DWORD & 0 & Error code \\
\hline wSizeExtDiag & DWORD & 0 & \begin{tabular}{l} 
Number of bytes of extended diagnosis \\
data
\end{tabular} \\
\hline
\end{tabular}

\section*{Structure DIAG_TXT_TYPE}

This data type specifies the format of all kinds of diagnosis messages in textual format. It consists of a single string containing all details of a diagnosis message.
\begin{tabular}{|l|l|l|l|}
\hline Name & Type & Initial & Comment \\
\hline szDiag & STRING(512) & "" & \begin{tabular}{l} 
Diagnosis message as text, max. 512 \\
characters
\end{tabular} \\
\hline
\end{tabular}

The text consists of the following data, separated by semicolon:
- Timestamp in Date_And_Time (DT) format of eDiagEvent_Occured and added milliseconds
- Error severity
- Device name (max. 80 characters) as defined in Automation Builder device tree
- The error text itself, composed of the interpretation of dwSubSysteminfo and dwAdditional and the error text plus remedy (if available) from Automation Builder text list according dwErrorCode. Displayed as: error text -> remedy.

\section*{Example \(\quad\) Battery empty or missing.}
```

= strData0utTxt

## Enumeration ERROR_ID

Type of the return values of all methods and functions to request information on diagnosis.

| Name | Type | Comment | Remedy |
| :--- | :--- | :--- | :--- |
| NO_ERROR | $16 \# 0$ | Execution successfully completed |  |
| ERR_PARAMETER | $16 \# 1$ | Invalid parameter value in function call | Correct parameter |
| ERR_NO_SINK | $16 \# 2$ | Failed to register as sink |  |
| ERR_NO_TEXT_LIST | $16 \# 3$ | Failed to get a device text list | Check text lists |
| ERR_NO_TEXT_ <br> CONTENT | $16 \# 4$ | Failed to get at least one content from text <br> list |  |
| ERR_COMPETING | $16 \# 5$ | Failed due to competing access of other <br> method | Try again |
| ERR_ASYNC | $16 \# 6$ | Failed to create async process |  |
| ERR_INTERNAL | $16 \# 7$ | Any internal error during execution |  |
| BUSY | $16 \# F F F$ | Busy | Call again to get final result |
| NO_ERROR_NO_DATA | $16 \# F F F F$ | Execution successfully completed, no more <br> diagnosis messages |  |

All values except "BUSY" are final results. In case "NO_ERROR" is returned, the requested action has been successfully performed. "NO_ERROR_NO_DATA" also indicates a successful completion. The only difference to "NO_ERROR" is the fact, that there is one (more) data to be provided. All other return values (except "BUSY") are final error states. In case a method or function returns "BUSY", it has to be called again in the following cycles until it returns a final result.

## Enumeration teClass

Specifies the error severity of diagnosis messages ${ }^{\wedge} \Rightarrow$ Chapter 6.9.1.6.1 "Error severity" on page 4638.

| Name | Error severity |
| :--- | :---: |
| eDiagClass_2_SeriousError | 2 |
| eDiagClass_3_Error | 3 |
| eDiagClass_4_Warning | 4 |
| eDiagClass_Parameter | 11 |

## Enumeration teEvent

The enumeration teEvent specifies the severity of diagnosis messages.

| Name | Type | Initial | Comment |
| :--- | :---: | :---: | :--- |
| eDiagEvent_Occured | DINT | 1 | Error occurred, remains "active" until <br> eDiagEvent_Disappeared |
| eDiagEvent_Disappeared | DINT | 2 | Error disappeared, became "active" due to <br> eDiagEvent_Occured earlier on |
| eDiagEvent_Received | DINT | 4 | Received a diagnosis message which cannot be analyzed in <br> detail, cannot disappear, needs to be acknowledged |
| eDiagEvent_Acknowledged | DINT | 8 | Acknowledge a diagnosis message which has been received <br> by eDiagEvent_Received, removes diagnosis message from <br> diagnosis system although error may still be present |

## Enumeration teHwid

The enumeration teHwld specifies the hardware component as location of a diagnosis message.

| Name | Initial | Comment |
| :--- | :---: | :--- |
| eDiagHwld_CPU | 0 | Any diagnosis message regarding CPU itself, like "battery", "sd card", <br> etc. |
| eDiagHwld_Coupler1 | 1 | Any diagnosis message regarding communication module at slot 1. <br> May be indicated by any corresponding communication module driver <br> (instance) or a protocol driver (instance) |
| eDiagHwld_Coupler2 | 2 | Any diagnosis message regarding communication module at slot 2. <br> May be indicated by any corresponding communication module driver <br> (instance) or a protocol driver (instance) |
| eDiagHwld_Coupler3 | 3 | Any diagnosis message regarding communication module at slot 3. <br> May be indicated by any corresponding communication module driver <br> (instance) or a protocol driver (instance) |
| eDiagHwld_Coupler4 | 4 | Any diagnosis message regarding communication module at slot 4. <br> May be indicated by any corresponding communication module driver <br> (instance) or a protocol driver (instance) |


| Name | Initial | Comment |
| :--- | :---: | :--- |
| eDiagHwld_Coupler5 | 5 | Any diagnosis message regarding communication module at slot 5. <br> May be indicated by any corresponding communication module driver <br> (instance) or a protocol driver (instance) |
| eDiagHwld_Coupler6 | 6 | Any diagnosis message regarding communication module at slot 6. <br> May be indicated by any corresponding communication module driver <br> (instance) or a protocol driver (instance) |
| eDiagHwld_COM1 | 7 | Any diagnosis message regarding COM1. May be indicated by any corre- <br> sponding interface driver (instance) or a protocol driver (instance) |
| eDiagHwld_COM2 | 8 | Any diagnosis message regarding COM2. May be indicated by any corre- <br> sponding interface driver (instance) or a protocol driver (instance) |
| eDiagHwld_CAN | 9 | Any diagnosis message regarding CAN interface. May be indicated <br> by any corresponding interface driver (instance) or a protocol driver <br> (instance) |
| eDiagHwld_OnboardIO | 10 | Onboard I/O, AC500-eCo only |
| eDiagHwld_OptionBoard1 | 11 | Option board 1, AC500-eCo only |
| eDiagHwld_OptionBoard2 | 12 | Option board 2, AC500-eCo only |
| eDiagHwld_OptionBoard3 | 13 | Option board 3, AC500-eCo only |
| eDiagHwld_IOBus | 14 | l/O bus |
| eDiagHwld_ETH1 | 15 | Any diagnosis message regarding ETH1. May be indicated by any corre- <br> sponding interface driver (instance) or a protocol driver (instance) |
| eDiagHwld_ETH2 | 16 | Any diagnosis message regarding ETH2. May be indicated by any corre- <br> sponding interface driver (instance) or a protocol driver (instance) |

The hardware ID Hwld is only used for diagnosis output in CPU display to identify the location of a diagnosis message ${ }^{\mu} \Rightarrow$ Chapter 6.9.1.3 "Diagnosis in CPU display" on page 4607.

### 6.9.1.5.3 System diagnosis

## Introduction

Library AC500_Diag provides several methods and functions to access the diagnosis messages on all devices in the PLC application. It contains also a function to convert numeric diagnosis into a textual format.

The variables and their assigned values can be referred to within the IEC application as well as they can be used to transfer diagnosis messages to any visualization client.

## Device state

## General

The library contains a single function block named "Diag", providing several methods to process the device state.

| Method | Description |
| :--- | :--- |
| NumTotal | Provides the total number of currently active diagnosis messages |
| NumClass | Provides the number of currently active diagnosis messages related to the <br> error severity |

## Method NumTotal

This method provides the total number of currently active diagnosis messages (including parameter errors, etc.).

| Scope | Name | Type | Comment |
| :--- | :--- | :--- | :--- |
| Return | NumTotal | DWORD | Number of diagnosis messages |

```
Example VAR
    fbDiag: AC500_Diag.Diag; (* instance of FB Diag *)
    dwNumTotal : DWORD;
END_VAR
```

dwNumTotal := fbDiag.NumTotal();

## Method NumClass

This method provides the number of currently active diagnosis messages related to the error severity.

| Scope | Name | Type | Initial | Comment |
| :--- | :--- | :--- | :--- | :--- |
| Return | NumClass | DWORD |  |  |
| Input | DatalnVal | AC500_DiagTypes.teClass | eClass.eDiagClass_2 <br> _SeriousError | Error severity <br> of diagnosis <br> message |

```
Example
VAR
    fbDiag:AC500_Diag.Diag;
    eDiagClass_4: AC500_DiagTypes.teClass := AC500_DiagTypes.teClass.eDiagClass_4_Warning; (* 4 *)
    eDiagClass_3: AC500_DiagTypes.teClass := AC500_DiagTypes.teClass.eDiagClass_3_Error; (* 3 *)
    eDiagClass_11: AC500_DiagTypes.teClass := AC500_DiagTypes.teClass.eDiagClass_Parameter; (* 11 *)
    dwNumClass_4 : DWORD;
    dwNumClass_3 : DWORD;
    dwNumClass_11: DWORD;
END_VAR
dwNumClass_4 := fbDiag.NumClass(eDiagClass := eDiagClass_4);
dwNumClass_3 := fbDiag.NumClass(eDiagClass := eDiagClass_3);
dwNumClass_11 := fbDiag.NumClass(eDiagClass := eDiagClass_11);
```


## Diagnosis descriptions

## General

The library contains a single function block "Diag", providing several methods to process diagnosis descriptions.

All methods for system diagnosis start with "Get...". For device diagnosis the prefix "Diag" is added: "DiagGet...". For better readability, only the method names for system diagnosis is used in the descriptions of the methods.

Table 856: Methods for handling diagnosis entries

| Method for system diagnosis | Method for device diagnosis | Description |
| :---: | :---: | :---: |
| Ack | DiagAck | Acknowledge a diagnosis alarm previously requested by using any Get. . . / DiagGet. . . method ${ }^{4}$ Chapter 6.9.1.5.3.3.2 "Method Ack / DiagAck: acknowledgement" on page 4623 |
| GetFirstVal | DiagGetFirstVal | Get the first (oldest) diagnosis message, numeric values ${ }^{4}$ Chapter 6.9.1.5.3.3.3 "Methods Get... / DiagGet...: get and sort diagnosis messages" on page 4623 |
| GetNextVal | DiagGetNextVal | Get the next diagnosis message, numeric values ${ }^{*}$ " Chapter 6.9.1.5.3.3.4 "Method Get-xxx-Val / DiagGet-xxx-Val: numeric values" on page 4624 |
| GetLastVal | DiagGetLastVal | Get the last (newest) diagnosis message, numeric values Chapter 6.9.1.5.3.3.4 "Method Get-xxx-Val / DiagGet-xxx-Val: numeric values" on page 4624 |
| GetPrevVal | DiagGetPrevVal | Get the previous diagnosis message, numeric values ${ }^{\circ}$ " Chapter 6.9.1.5.3.3.4 "Method Get-xxx-Val / DiagGet-xxx-Val: numeric values" on page 4624 |
| GetFirstValExt | DiagGetFirstValExt | Get the first (oldest) diagnosis message, numeric and extended numeric values (2) Chapter 6.9.1.5.3.3.5 "Method Get-xxxValExt / DiagGet-xxx-ValExt: numeric values and extended numeric values" on page 4624 |
| GetNextValExt | DiagGetNextValExt | Get the next diagnosis message, numeric and extended numeric values ${ }^{4}$ Chapter 6.9.1.5.3.3.5 "Method Get-xxx-ValExt / Dia-gGet-xxx-ValExt: numeric values and extended numeric values" on page 4624 |
| GetLastValExt | Diag GetLastValExt | Get the last (newest) diagnosis message, numeric and extended numeric values \# Chapter 6.9.1.5.3.3.5 "Method Get-xxxValExt / DiagGet-xxx-ValExt: numeric values and extended numeric values" on page 4624 |
| GetPrevValExt | DiagGetPrevValExt | Get the previous diagnosis message, numeric and extended numeric values \# Chapter 6.9.1.5.3.3.5 "Method Get-xxxValExt / DiagGet-xxx-ValExt: numeric values and extended numeric values" on page 4624 |


| Method for system diagnosis | Method for device diagnosis | Description |
| :---: | :---: | :---: |
| GetFirstValAndTxt | DiagGetFirstVaIAndTxt | Get the first (oldest) diagnosis message, numeric values and text $\Leftrightarrow$ Chapter 6.9.1.5.3.3.6 "Method Get-xxx-ValAndTxt / DiagGet-xxx-ValAndTxt: numeric values and text" on page 4625 |
| GetNextValAndTxt | DiagGetNextVaIAndTxt | Get the next diagnosis message, numeric values and text ${ }^{\circ} \boldsymbol{y}$ Chapter 6.9.1.5.3.3.6 "Method Get-xxx-ValAndTxt / DiagGet-xxxValAndTxt: numeric values and text" on page 4625 |
| GetLastValAndTxt | DiagGetLastVaIAndTxt | Get the last (newest) diagnosis message, numeric values and text $\Leftrightarrow$ Chapter 6.9.1.5.3.3.6 "Method Get-xxx-ValAndTxt / DiagGet-xxx-ValAndTxt: numeric values and text" on page 4625 |
| GetPrevValAndTxt | DiagGetPrevVaIAndTxt | Get the previous diagnosis message, numeric values and text ${ }^{\circ} \Rightarrow$ Chapter 6.9.1.5.3.3.6 "Method Get-xxx-ValAndTxt / DiagGet-xxx-ValAndTxt: numeric values and text" on page 4625 |
| GetFirstValAndTxtExt | DiagGetFirstVaIAndTxtExt | Get the first (oldest) diagnosis message, numeric, extended numeric values and text ${ }^{4} \boldsymbol{z}$ Chapter 6.9.1.5.3.3.7 "Method Get-xxxValAndTxtExt / DiagGet-xxx-ValAndTxtExt: numeric values, extended numeric values and text" on page 4626 |
| GetNextValAndTxtExt | DiagGetNextVaIAndTxtExt | Get the next diagnosis message, numeric, extended numeric values and text $\stackrel{y}{ } \Rightarrow$ Chapter 6.9.1.5.3.3.7 "Method Get-xxx-ValAndTxtExt / DiagGet-xxx-ValAndTxtExt: numeric values, extended numeric values and text" on page 4626 |
| GetLastValAndTxtExt | DiagGetLastVaIAndTxtExt | Get the last (newest) diagnosis message, extended numeric values and text ${ }^{3} \boldsymbol{z}$ Chapter 6.9.1.5.3.3.7 "Method Get-xxxValAndTxtExt / DiagGet-xxx-ValAndTxtExt: numeric values, extended numeric values and text" on page 4626 |
| GetPrevValAndTxtExt | DiagGetPrevVaIAndTxtExt | Get the previous diagnosis message, extended numeric values and text $\stackrel{\leftrightarrow}{ }$ Chapter 6.9.1.5.3.3.7 "Method Get-xxx-ValAndTxtExt / DiagGet-xxx-ValAndTxtExt: numeric values, extended numeric values and text" on page 4626 |

## Method Ack / DiagAck: acknowledgement

This method can be used to acknowledge a diagnosis alarm previously requested by using any Get. . . / DiagGet . . . method. Alternatively, you can acknowledge an alarm in Automation Builder.

After acknowledgement, the alarm is deleted from the diagnosis system.

| Scope | Name for <br> device <br> diagnosis | Name for <br> device <br> diagnosis | Type | Comment |
| :--- | :--- | :--- | :--- | :--- |
| Return | Ack | DiagAck | AC500_DiagTypes.ERROR_ID |  |
| Input | Data | Data | AC500_Dia- <br> gTypes.DIAG_VAL_TYPE | Variable containing <br> details of diagnosis <br> alarm to be acknowl- <br> edged |


| Example | System diagnosis: acknowledge first diagnosis message |
| :--- | :--- | :--- |

eErrorID_First := fbDiag.Ack(Data := sFirstVal);

## Methods Get... / DiagGet...: get and sort diagnosis messages

All these methods can be used to get the first (oldest), next, last (newest) or previous diagnosis message stored in diagnosis system. The only difference are the details the methods provide. While, e.g., Get-xxx-Val just provides the basic information in numeric format, Get-xxxValExt additionally provides this information by the extended diagnosis data of the entry.

The numeric format provided by these methods can be converted into textual format later on if required ${ }^{\&}$ \& Chapter 6.9.1.5.3.3.8 "Function DiagValToTxt" on page 4627. Alternatively, the methods Get-xxx-ValAndTxt and Get-xxx-ValAndTxtExt can be used for numeric and textual format in parallel ${ }^{\mu}>$ Chapter 6.9.1.5.3.3.6 "Method Get-xxx-ValAndTxt / DiagGet-xxx-ValAndTxt: numeric values and text" on page 4625 " ${ }^{\text {s }}$ Chapter 6.9.1.5.3.3.7 "Method Get-xxx-ValAndTxtExt / DiagGet-xxx-ValAndTxtExt: numeric values, extended numeric values and text" on page 4626.

All methods may need multiple cycles to process the request. Therefore, they must be called in successive cycles until they return a final result $\stackrel{\text { ® }}{ }$ Chapter 6.9.1.5.2.4 "Enumeration $E R R O R$ _ID" on page 4617.

All diagnosis messages sorted by time, ascending

1. Call any GetFirst. . . method until it indicates a final result.
2. If the result is not "NO_ERROR_NO_DATA": Call any GetNext . . . method as long as its final result is "NO_ERROR".

All diagnosis messages sorted by time, descending

1. Call any GetLast . . . method until it indicates a final result.
2. If the result is not "NO_ERROR_NO_DATA": Call any GetPrev . . . method as long as its final result is "NO_ERROR".

Method Get-xxx-Val / DiagGet-xxx-Val: numeric values
-xxx- = First, Next, Last, Prev. Example: GetFirstVal, DiagGetLastVal.

| Scope | Name for <br> device diag- <br> nosis | Name for <br> device diag- <br> nosis | Type | Comment |
| :--- | :--- | :--- | :--- | :--- |
| Return | Get-xxx-Val | DiagGet-xxx- <br> Val | AC500_DiagTypes.ERROR_ID |  |
| Inout | Data | Data | AC500_Dia- <br> gTypes.DIAG_VAL_TYPE | Variable to write data <br> to |


| Example | System diagnosis: get values for first diagnosis message |  |
| :---: | :---: | :---: |
|  | VAR <br> fbDiag: AC500_Diag.Diag; <br> eErrorID_First: AC500_DiagTypes.ERROR_ID; <br> sFirstVal : AC500_DiagTypes.DIAG_VAL_TYPE; <br> END VAR | (* instance of FB Diag *) (* return value *) (* structure of returned values *) |
|  | eErrorID_First := fbDiag.GetFirstVal(Data $:=$ sFirstVal); |  |

Example Online mode: battery empty or missing

| 盏 PLC_PRG $\times$ |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| PLC_AC500_V3.Application.PLC_PRG |  |  |  |  |  |
| Expression $\pm \text { fbDiag }$ | Type <br> AC500_Diag.Diag | Value | Prepared value | Address | Comment instance of FB Diag |
| - eErrorid_First | ERROR_ID | NO_ERROR |  |  | return value |
| - sFirstVal | AC500_Diag.AC500... |  |  |  | structure of returend values |
| - dtTimestamp | DATE_AND_TIME | DT\#1970-1-1-0:0:15 |  |  | RTC time of event |
| - uiMs | UINT | 356 |  |  | Milliseconds of event |
| - eClass | TECLASS | eDiagClass_4_Warning |  |  | Severity of error event |
| - szDevice | STRING(80) | 'PLC_AC500_V3' |  |  | Name of device |
| - eEvent | TEEVENT | eDiagEvent_Occurred |  |  | Type of event |
| - eHwinterfac... | TEHWID | eDiagHwId_CPU |  |  | Identifier of hardware interface |
| - dwSubSyste... | DWORD | 369098752 |  |  | Any number describin...ail/location within d... |
| - dwadditional | DWORD | 0 |  |  | Additional number des ...ng detail/location wi... |
| - udErrorCode | UDINT | 8 |  |  | Error code |
| - uiSizeExtDiag | UINT | 0 |  |  | Number of bytes of extended diagnosis data |
| $\pm$ hSource | POINTER TO BYTE | 16\#008703D0 |  |  | internal reference needed for text convertion |
| $\pm$ pConn | POINTER TO IoStan... | 16\#B134A1D4 |  |  | internal reference needed for text convertion |

1) eErrorID_First NO_ERROR := fbDiag.GetFirstVal(Data := sFirstVal);

Method Get-xxx-ValExt / DiagGet-xxx-ValExt: numeric values and extended numeric values
-xxx- = First, Next, Last, Prev. Example: GetNextValExt, DiagGetPrevValExt.
\(\left.$$
\begin{array}{|l|l|l|l|l|}\hline \begin{array}{l}\text { Scop } \\
\text { e }\end{array} & \begin{array}{l}\text { Name for } \\
\text { device diag- } \\
\text { nosis }\end{array} & \begin{array}{l}\text { Name for } \\
\text { device diag- } \\
\text { nosis }\end{array} & \text { Type } & \text { Comment } \\
\hline \begin{array}{l}\text { Retur } \\
n\end{array} & \text { Get-xxx-ValExt } & \begin{array}{l}\text { DiagGet-xxx- } \\
\text { ValExt }\end{array} & \begin{array}{l}\text { AC500_Dia- } \\
\text { gTypes.ERROR_ID }\end{array} & \begin{array}{l}\text { AC500_Dia- } \\
\text { gTypes.DIAG_VAL_TYP } \\
\text { E }\end{array} \\
\hline \text { Inout } & \text { Data } & \text { Data } & \begin{array}{l}\text { Variable to write data } \\
\text { to }\end{array} \\
\hline \text { Input } & \text { pExt } & \text { pExt } & \text { Size } & \text { WORD }\end{array}
$$ \begin{array}{l}Address of buffer to <br>

copy extended data to\end{array}\right]\)| Size of buffer to copy |
| :--- |
| extended data to |$|$| Size of extended data |
| :--- |
| copied to buffer |, | Input |
| :--- |
| Size |


| Example | System diagnosis: get numeric values and ex sage | numeric values for first diagnosis mes- |
| :---: | :---: | :---: |
|  | VAR |  |
|  | fbDiag: AC500_Diag.Diag; | (* instance of FB Diag *) |
|  | eErrorID_First : AC500_DiagTypes.ERROR_ID; | (* return value *) |
|  | sFirstVal : AC500_DiagTypes.DIAG_VAL_TYPE; | (* structure of returned values *) |
|  | abyExtData : ARRAY[0..1023] OF BYTE; | (* byte array for extended diagnosis data *) |
|  | wSize : WORD := 1024; | (*Size of buffer to copy extended data to *) |
|  | wLength : WORD; | (*Size of extended data copied to buffer *) |
|  | END_VAR |  |

Method Get-xxx-ValAndTxt / DiagGet-xxx-ValAndTxt: numeric values and text
-xxx- = First, Next, Last, Prev. Example: GetFirstValAndTxt, DiagGetPrevValAndTxt

| Scop <br> e | Name for <br> device diag- <br> nosis | Name for <br> device diag- <br> nosis | Type | Comment |
| :--- | :--- | :--- | :--- | :--- |
| Retur <br> n | Get-xxx-Val- <br> AndTxt | DiagGet-xxx-Val- <br> AndTxt | AC500_Dia- <br> gTypes.ERROR_ID |  |
| Inout | DataVal | DataVal | AC500_Dia- <br> gTypes.DIAG_VAL_TYPE | Variable to write <br> data to |
| Inout | DataTxt | DataTxt | AC500_Dia- <br> gTypes.DIAG_TXT_TYPE | Variable to write <br> text to |


| Example | System diagnosis: get numeric values and text for first diagnosis message |  |
| :---: | :---: | :---: |
|  | VAR <br> fbDiag: AC500_Diag.Diag; eErrorID_First: AC500_DiagTypes.ERROR_ID; sFirstVal : AC500_DiagTypes.DIAG_VAL_TYPE; sFirstTxt:AC500_DiagTypes.DIAG_TXT_TYPE; <br> END VAR | ```(* instance of FB Diag *) (* return value *) (* structure of returned values *) (* structure of returned text *)``` |
|  | eErrorID_First := fbDiag.GetFirstValAndTxt(DataVal := sFirstVal, DataTxt := sFirstTxt); |  |

Example Online mode: battery empty or missing

| 㝣 PLC_PRG_1 $\times$ |  |  |
| :---: | :---: | :---: |
| PLC_AC500_V3.Application.PLC_PRG_1 |  |  |
| Expression $\pm \text { fbDiag }$ | Type AC500_Diag.Diag | Value |
| ( eErrorID_First | ERROR_ID | NO_ERROR |
| - sFirstVal | AC500_Diag.AC500... |  |
| - dtTimestamp | DATE_AND_TIME | DT\#1970-1-1-0:0:15 |
| - uiMs | UINT | 356 |
| - eClass | TECLASS | eDiagClass_4_Warning |
| - szDevice | STRING(80) | 'PLC_AC500_V3' |
| - EEvent | TEEVENT | eDiagEvent_Occurred |
| - eHwinterfac... | TEHWID | eDiagHwid_CPU |
| - dwSubSyste... | DWORD | 369098752 |
| - dwadditional | DWORD | 0 |
| - udErrorCode | UDINT | 8 |
| - uiSizeExtDiag | UINT | 0 |
| $\pm$ hSource | POINTER TO BYTE | 16\#008703D0 |
| $\pm$ pConn | POINTER TO IoStan... | 16\#B134A1D4 |
| - sFirstTxt | AC500_Diag.AC500... |  |
| - szDiag | STRING(512) | '1970-01-01;00:00:15,356;E4;PLC_AC500_V3;8;Battery Device Empty or missing' |
| - szColumn | STRING(1) | ';' |

I| eErrorID_First NO_ERROR := fbDiag.GetFirstValAndTxt(DataVal := sFirstVal, DataTxt := sFirstTxt);

Method Get-xxx-ValAndTxtExt / DiagGet-xxx-ValAndTxtExt: numeric values, extended numeric values and text
-xxx- = First, Next, Last, Prev. Example: GetLastValAndTxtExt, DiagGetFirstValAndTxtExt

| Scop <br> e | Name for device <br> diagnosis | Name for device <br> diagnosis | Type | Initial | Comment |
| :--- | :--- | :--- | :--- | :--- | :--- |
| Retur <br> $n$ | Get-xxx-Val- <br> AndTxtExt | DiagGet-xxx-Val- <br> AndTxtExt | AC500_Dia- <br> gTypes.ERROR_ID |  |  |
| Inout | DataVal | DataVal | AC500_Dia- <br> gTypes.DIAG_VAL_ <br> TYPE | Variable to write <br> data to |  |


| Scop <br> e | Name for device <br> diagnosis | Name for device <br> diagnosis | Type | Initial | Comment |
| :--- | :--- | :--- | :--- | :--- | :--- |
| Inout | DataTxt | DataTxt | AC500_Dia- <br> gTypes_DIAG_TXT_ <br> TYPE |  | Variable to write <br> text to |
| Input | pExt | pExt | POINTER TO BYTE | 0 | Address of buffer <br> to copy extended <br> data to |
| Input | Size | Size | WORD | Size of buffer to <br> copy extended <br> data to |  |
| Inout | Length | Length | WORD | Size of extended <br> data copied to <br> buffer |  |


| Example | System diagnosis: get numeric values, extended numeric values and text of first diagnosis message |  |
| :---: | :---: | :---: |
|  | VAR |  |
|  | fbDiag: AC500_Diag.Diag; | (* instance of FB Diag *) |
|  | eErrorID_First : AC500_DiagTypes.ERROR_ID; | (* return value *) |
|  | sFirstVal: AC500_DiagTypes.DIAG_VAL_TYPE; | (* structure of returned values *) |
|  | sFirstTxt : AC500_DiagTypes.DIAG_TXT_TYPE; | (* structure of returned text *) |
|  | abyExtData : ARRAY[0...1023] OF BYTE; | (* byte array for extended diagnosis data *) |
|  | wSize : WORD := 1024; | (* Size of buffer to copy extended data to *) |
|  | wLength : WORD; | (* Size of extended data copied to buffer *) |
|  | END_VAR |  |

eErrorID_First := fbDiag.GetFirstValExt(DataVal := sFirstVal, DataTxt := sFirstTxt,
pExt := ADR(abyExtData), Size := wSize, Length := wLength);

## Function DiagValToTxt

Call this function to convert a numeric diagnosis message into a textual one at any time, in case this has not yet been done using a method providing both types when requesting this information.

| Scope | Name | Type | Comment |
| :--- | :--- | :--- | :--- |
| Return | DiagValToTxt | AC500_DiagTypes.ERROR_ID |  |
| Inout | DataInVal | AC500_DiagTypes.DIAG_VAL_TYPE | Variable to convert |
| Inout | DataOutTxt | AC500_DiagTypes.DIAG_TXT_TYPE | Variable to write text to |


| Example | System diagnosis: convert first diagnosis message from numeric value to text |  |
| :---: | :---: | :---: |
|  | VAR <br> eErrorID_ValToTxt : AC500_DiagTypes.ERROR_ID; <br> sFirstVal:AC500_DiagTypes.DIAG_VAL_TYPE; szDataOutTxt:AC500_DiagTypes.DIAG_TXT_TYPE; | (* return value *) <br> (* structure of returned values *) <br> (* string of converted data *) |

eErrorID_ValToTx := DiagValToTxt(DataInVal := sFirstVal, DataOutTxt := szDataOutTxt);


### 6.9.1.5.4 Device diagnosis

## Activate device diagnosis

While the notification of diagnosis messages at the display and the Automation Builder is enabled by default, the functionality to access diagnosis messages from within the IEC application needs to be explicitly enabled.

1. Double-click on the CPU in the device tree.
2. Select tab "PLC Settings".
3. Under "Additional Settings" select "Enable Diagnosis for devices".
$\Rightarrow$ Library CAA Device Diagnosis (namespace DED) is automatically included in the project. This library is needed for displaying and processing the device state.

In case the functionality of diagnosis is no longer needed in IEC application, we recommend to disable this setting.

## Device state

1．Open one of the IEC application code editors．
2．Type the device＇s name as it is written in the device tree，followed by a dot（＂．＂）．
3．Select the method GetDeviceState from the context menu or type the name of the method on yourself．

```
State_DC532 := DC532.
```

| －${ }_{\text {M }}$ DiagGetNextValAndTxt | ＊ |
| :---: | :---: |
| 产DiagGetNextValAndTxtExt |  |
| 二M DiagGetNextValExt |  |
| $\cdots{ }^{-1}$ DiagGetPrevVal |  |
| 产 DiagGetPrevValAndTxt |  |
| 誊DiagGetPrevValAndTxtExt |  |
| 产DiagGetPrevValExt | 三 |
| $\bar{Z}^{-1}$ GetDeviceInfo |  |
| － 7 GetDeviceState |  |
| －${ }_{\text {M }}$ IoDrvGetInfo | $\checkmark$ |

4．Assign the function＇s parameters．

```
eError : DED.ERROR
bDiagAvailable : BOOL; (* at least one diagnosis avilable for this device *)
DeviceState_DC532 : DED.DEVICE_STATE; (* general device state *)
DeviceState_DC532 := DC532.GetDeviceState(xDiagnosisInfoAvailable => bDiagAvailable, eError => eError);
```

| Scope | Name | Type | Comment |
| :--- | :--- | :--- | :--- |
| Return | GetDeviceState | DEVICE＿STATE <br> ¿ Further information <br> on page 4629 | Current device state |
| Output | xDiagnosisInfoAvailable | BOOL | If TRUE，diagnosis messages <br> are available regarding the <br> concerning device（＝node）． |
| Output | eError | ERROR \＆Fur－ <br> ther information <br> on page 4630 | Type of the return values of <br> all methods and functions of <br> library CAA Device Diagnosis |

Table 857：Enumeration DEVICE＿STATE（part of the library CAA Device Diagnosis（DED））

| Name | Type | Initial | Icon in AB | Comment |
| :--- | :---: | :---: | :---: | :--- |
| UNKNOWN | INT | 0 |  | The device is in state unknown． <br> Example：No supervision mechanism <br> active |
| STOPPED | INT | 1 |  | The device is stopped． |
| RUNNING | INT | 2 |  | The device is running． |
|  |  |  |  |  |
| ERROR | INT | 3 |  | The device is in error state． |
| DISABLED | INT | 4 |  | The device is disabled in device tree． |


| Name | Type | Initial | Icon in AB | Comment |
| :--- | :---: | :---: | :---: | :--- |
| NOT_CONFIG- <br> URED | INT | 5 |  | The device has not been yet configured by <br> the stack. <br> Example: Configuration phase not yet <br> started |
| CONFIGURED | INT | 6 |  | The device has been configured by the <br> stack. <br> Example: Configuration phase finished but <br> the device is not in running state |
| NOT_FOUND | INT | 7 | $\mathbf{A r}$ | The device was not found on bus. |

Table 858: Enumeration ERROR (part of the library CAA Device Diagnosis (DED))

| Name | Type | Initial | Comment |
| :--- | :---: | :--- | :--- |
| NO_ERROR | INT | 0 | No error |
| FIRST_ERROR | INT | 1300 | First library-specific error |
| TIME_OUT | INT | 1301 | Timeout occured. |
| ABORT | INT | 1302 | Operation was aborted. |
| REF_INVALID | INT | 1303 | The interface reference was invalid. |
| NOT_SUPPORTED | INT | 1304 | The function is not supported. |
| ERROR_IO | INT | 1305 | A general I/O configuration error occured. |
| PARAM_INVALID | INT | 1306 | Invalid parameter |
| NODE_NOT_EXISTING | INT | 1307 | The specified node does not exist. |
| NO_MEMORY | INT | 1308 | Dynamic memory allocation is disabled, or <br> system is out of memory. |
| ADR_NOT_FOUND | INT | 1309 | The specified I/O address is not valid. |
| INST_NOT_FOUND | INT | 1310 | There is no associated [Device] instance <br> for the specific I/O address. |
| NO_DATA | INT | 1311 | There is no data available. |
| OPERATION_INVALID | INT | 1312 | Operation not possible due to the current <br> state |
| FIRST_MF | INT | 1350 | First manufacturer-specific error |
| LAST_ERROR | INT | 1399 | Last error |

## Diagnosis descriptions

## General

The library contains a single function block "Diag", providing several methods to process diagnosis descriptions.

All methods for system diagnosis start with "Get...". For device diagnosis the prefix "Diag" is added: "DiagGet...". For better readability, only the method names for system diagnosis is used in the descriptions of the methods.

Table 859: Methods for handling diagnosis entries

| Method for system diagnosis | Method for device diagnosis | Description |
| :---: | :---: | :---: |
| Ack | DiagAck | Acknowledge a diagnosis alarm previously requested by using any Get. . . / DiagGet. . . method ${ }^{4}$ Chapter 6.9.1.5.4.3.2 "Method Ack / DiagAck: acknowledgement" on page 4633 |
| GetFirstVal | DiagGetFirstVal | Get the first (oldest) diagnosis message, numeric values ${ }^{4}$ Chapter 6.9.1.5.4.3.3 "Methods Get... / DiagGet...: get and sort diagnosis messages" on page 4633 |
| GetNextVal | DiagGetNextVal | Get the next diagnosis message, numeric values ${ }^{*}$ " Chapter 6.9.1.5.4.3.4 "Method Get-xxx-Val / DiagGet-xxx-Val: numeric values" on page 4634 |
| GetLastVal | DiagGetLastVal | Get the last (newest) diagnosis message, numeric values ${ }^{*}$, Chapter 6.9.1.5.4.3.4 "Method Get-xxx-Val / DiagGet-xxx-Val: numeric values" on page 4634 |
| GetPrevVal | DiagGetPrevVal | Get the previous diagnosis message, numeric values ${ }^{\circ}$ " Chapter 6.9.1.5.4.3.4 "Method Get-xxx-Val / DiagGet-xxx-Val: numeric values" on page 4634 |
| GetFirstValExt | DiagGetFirstValExt | Get the first (oldest) diagnosis message, numeric and extended numeric values (2) Chapter 6.9.1.5.4.3.5 "Method Get-xxxValExt / DiagGet-xxx-ValExt: numeric values and extended numeric values" on page 4634 |
| GetNextValExt | DiagGetNextValExt | Get the next diagnosis message, numeric and extended numeric values ${ }^{4}$ Chapter 6.9.1.5.4.3.5 "Method Get-xxx-ValExt / Dia-gGet-xxx-ValExt: numeric values and extended numeric values" on page 4634 |
| GetLastValExt | Diag GetLastValExt | Get the last (newest) diagnosis message, numeric and extended numeric values \# Chapter 6.9.1.5.4.3.5 "Method Get-xxxValExt / DiagGet-xxx-ValExt: numeric values and extended numeric values" on page 4634 |
| GetPrevValExt | DiagGetPrevValExt | Get the previous diagnosis message, numeric and extended numeric values \# Chapter 6.9.1.5.4.3.5 "Method Get-xxxValExt / DiagGet-xxx-ValExt: numeric values and extended numeric values" on page 4634 |


| Method for system diagnosis | Method for device diagnosis | Description |
| :---: | :---: | :---: |
| GetFirstValAndTxt | DiagGetFirstVaIAndTxt | Get the first (oldest) diagnosis message, numeric values and text $\Leftrightarrow$ Chapter 6.9.1.5.4.3.6 "Method Get-xxx-ValAndTxt / DiagGet-xxx-ValAndTxt: numeric values and text" on page 4635 |
| GetNextValAndTxt | DiagGetNextVaIAndTxt | Get the next diagnosis message, numeric values and text ${ }^{\circ} \boldsymbol{y}$ Chapter 6.9.1.5.4.3.6 "Method Get-xxx-ValAndTxt / DiagGet-xxxValAndTxt: numeric values and text" on page 4635 |
| GetLastValAndTxt | DiagGetLastVaIAndTxt | Get the last (newest) diagnosis message, numeric values and text $\stackrel{\leftrightarrow}{\leftrightarrows}$ Chapter 6.9.1.5.4.3.6 "Method Get-xxx-ValAndTxt / DiagGet-xxx-ValAndTxt: numeric values and text" on page 4635 |
| GetPrevValAndTxt | DiagGetPrevVaIAndTxt | Get the previous diagnosis message, numeric values and text $\Leftrightarrow$ Chapter 6.9.1.5.4.3.6 "Method Get-xxx-ValAndTxt / DiagGet-xxx-ValAndTxt: numeric values and text" on page 4635 |
| GetFirstValAndTxtExt | DiagGetFirstVaIAndTxtExt | Get the first (oldest) diagnosis message, numeric, extended numeric values and text ${ }^{4} \boldsymbol{z}$ Chapter 6.9.1.5.3.3.7 "Method Get-xxxValAndTxtExt / DiagGet-xxx-ValAndTxtExt: numeric values, extended numeric values and text" on page 4626 |
| GetNextValAndTxtExt | DiagGetNextVaIAndTxtExt | Get the next diagnosis message, numeric, extended numeric values and text $\stackrel{y}{ } \Rightarrow$ Chapter 6.9.1.5.3.3.7 "Method Get-xxx-ValAndTxtExt / DiagGet-xxx-ValAndTxtExt: numeric values, extended numeric values and text" on page 4626 |
| GetLastValAndTxtExt | DiagGetLastVaIAndTxtExt | Get the last (newest) diagnosis message, extended numeric values and text ${ }^{3} \boldsymbol{z}$ Chapter 6.9.1.5.3.3.7 "Method Get-xxxValAndTxtExt / DiagGet-xxx-ValAndTxtExt: numeric values, extended numeric values and text" on page 4626 |
| GetPrevValAndTxtExt | DiagGetPrevVaIAndTxtExt | Get the previous diagnosis message, extended numeric values and text $\stackrel{\leftrightarrow}{ }$ Chapter 6.9.1.5.3.3.7 "Method Get-xxx-ValAndTxtExt / DiagGet-xxx-ValAndTxtExt: numeric values, extended numeric values and text" on page 4626 |

## Method Ack / DiagAck: acknowledgement

This method can be used to acknowledge a diagnosis alarm previously requested by using any Get. . . / DiagGet . . . method. Alternatively, you can acknowledge an alarm in Automation Builder.

After acknowledgement, the alarm is deleted from the diagnosis system.

| Scope | Name for <br> device <br> diagnosis | Name for <br> device <br> diagnosis | Type | Comment |
| :--- | :--- | :--- | :--- | :--- |
| Return | Ack | DiagAck | AC500_DiagTypes.ERROR_ID |  |
| Input | Data | Data | AC500_Dia- <br> gTypes.DIAG_VAL_TYPE | Variable containing <br> details of diagnosis <br> alarm to be acknowl- <br> edged |


| Example | System diagnosis: acknowledge first diagnosis message |
| :--- | :--- | :--- |

eErrorID_First := fbDiag.Ack(Data := sFirstVal);

## Methods Get... / DiagGet...: get and sort diagnosis messages

All these methods can be used to get the first (oldest), next, last (newest) or previous diagnosis message stored in diagnosis system. The only difference are the details the methods provide. While, e.g., Get-xxx-Val just provides the basic information in numeric format, Get-xxxValExt additionally provides this information by the extended diagnosis data of the entry.

The numeric format provided by these methods can be converted into textual format later on if required ${ }^{\&}$ \& Chapter 6.9.1.5.3.3.8 "Function DiagValToTxt" on page 4627. Alternatively, the methods Get-xxx-ValAndTxt and Get-xxx-ValAndTxtExt can be used for numeric and textual format in parallel ${ }^{\mu}>$ Chapter 6.9.1.5.4.3.6 "Method Get-xxx-ValAndTxt / DiagGet-xxx-ValAndTxt: numeric values and text" on page 4635 " $\stackrel{\text { Chapter 6.9.1.5.4.3.7 "Method Get-xxx-Val- }}{ }$ AndTxtExt / DiagGet-xxx-ValAndTxtExt: numeric values, extended numeric values and text" on page 4636.

All methods may need multiple cycles to process the request. Therefore, they must be called in successive cycles until they return a final result $\stackrel{\text { ® }}{ }$ Chapter 6.9.1.5.2.4 "Enumeration $E R R O R$ _ID" on page 4617.

All diagnosis messages sorted by time, ascending

1. Call any GetFirst. . . method until it indicates a final result.
2. If the result is not "NO_ERROR_NO_DATA": Call any GetNext . . . method as long as its final result is "NO_ERROR".

All diagnosis messages sorted by time, descending

1. Call any GetLast . . . method until it indicates a final result.
2. If the result is not "NO_ERROR_NO_DATA": Call any GetPrev . . . method as long as its final result is "NO_ERROR".

Method Get-xxx-Val / DiagGet-xxx-Val: numeric values
-xxx- = First, Next, Last, Prev. Example: GetFirstVal, DiagGetLastVal.

| Scope | Name for <br> device diag- <br> nosis | Name for <br> device diag- <br> nosis | Type | Comment |
| :--- | :--- | :--- | :--- | :--- |
| Return | Get-xxx-Val | DiagGet-xxx- <br> Val | AC500_DiagTypes.ERROR_ID |  |
| Inout | Data | Data | AC500_Dia- <br> gTypes.DIAG_VAL_TYPE | Variable to write data <br> to |


| Example | System diagnosis: get values for first diagnosis message |  |
| :---: | :---: | :---: |
|  | VAR <br> fbDiag: AC500_Diag.Diag; <br> eErrorID_First: AC500_DiagTypes.ERROR_ID; <br> sFirstVal : AC500_DiagTypes.DIAG_VAL_TYPE; <br> END VAR | (* instance of FB Diag *) (* return value *) (* structure of returned values *) |
|  | eErrorID_First := fbDiag.GetFirstVal(Data $:=$ sFirstVal); |  |

Example Online mode: battery empty or missing

| 盏 PLC_PRG $\times$ |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| PLC_AC500_V3.Application.PLC_PRG |  |  |  |  |  |
| Expression $\pm \text { fbDiag }$ | Type <br> AC500_Diag.Diag | Value | Prepared value | Address | Comment instance of FB Diag |
| - eErrorid_First | ERROR_ID | NO_ERROR |  |  | return value |
| - sFirstVal | AC500_Diag.AC500... |  |  |  | structure of returend values |
| - dtTimestamp | DATE_AND_TIME | DT\#1970-1-1-0:0:15 |  |  | RTC time of event |
| - uiMs | UINT | 356 |  |  | Milliseconds of event |
| - eClass | TECLASS | eDiagClass_4_Warning |  |  | Severity of error event |
| - szDevice | STRING(80) | 'PLC_AC500_V3' |  |  | Name of device |
| - eEvent | TEEVENT | eDiagEvent_Occurred |  |  | Type of event |
| - eHwinterfac... | TEHWID | eDiagHwId_CPU |  |  | Identifier of hardware interface |
| - dwSubSyste... | DWORD | 369098752 |  |  | Any number describin...ail/location within d... |
| - dwadditional | DWORD | 0 |  |  | Additional number des ...ng detail/location wi... |
| - udErrorCode | UDINT | 8 |  |  | Error code |
| - uiSizeExtDiag | UINT | 0 |  |  | Number of bytes of extended diagnosis data |
| $\pm$ hSource | POINTER TO BYTE | 16\#008703D0 |  |  | internal reference needed for text convertion |
| $\pm$ pConn | POINTER TO IoStan... | 16\#B134A1D4 |  |  | internal reference needed for text convertion |

1) eErrorID_First NO_ERROR := fbDiag.GetFirstVal(Data := sFirstVal);

Method Get-xxx-ValExt / DiagGet-xxx-ValExt: numeric values and extended numeric values
-xxx- = First, Next, Last, Prev. Example: GetNextValExt, DiagGetPrevValExt.

| Scop e | Name for device diagnosis | Name for device diagnosis | Type | Comment |
| :---: | :---: | :---: | :---: | :---: |
| Retur <br> n | Get-xxx-ValExt | DiagGet-xxxValExt | AC500 DiagTypes.ERROR_ID |  |
| Inout | Data | Data | AC500 Dia- <br> gTypes.DIAG_VAL_TYP <br> E | Variable to write data to |
| Input | pExt | pExt | POINTER TO BYTE | Address of buffer to copy extended data to |
| Input | Size | Size | WORD | Size of buffer to copy extended data to |
| Inout | Length | Length | WORD | Size of extended data copied to buffer |


| Example | System diagnosis: get numeric values and extended numeric values for first diagnosis message |  |
| :---: | :---: | :---: |
|  | VAR |  |
|  | fbDiag: AC500_Diag.Diag; | (* instance of FB Diag *) |
|  | eErrorID_First : AC500_DiagTypes.ERROR_ID; | (* return value *) |
|  | sFirstVal : AC500_DiagTypes.DIAG_VAL_TYPE; | (* structure of returned values *) |
|  | abyExtData : ARRAY[0..1023] OF BYTE; | (* byte array for extended diagnosis data *) |
|  | wSize : WORD := 1024; | (*Size of buffer to copy extended data to *) |
|  | wLength : WORD; | (*Size of extended data copied to buffer *) |
|  | END_VAR |  |

Method Get-xxx-ValAndTxt / DiagGet-xxx-ValAndTxt: numeric values and text
-xxx- = First, Next, Last, Prev. Example: GetFirstValAndTxt, DiagGetPrevValAndTxt

| Scop <br> e | Name for <br> device diag- <br> nosis | Name for <br> device diag- <br> nosis | Type | Comment |
| :--- | :--- | :--- | :--- | :--- |
| Retur <br> $n$ | Get-xxx-Val- <br> AndTxt | DiagGet-xxx-Val- <br> AndTxt | AC500_Dia- <br> gTypes.ERROR_ID |  |
| Inout | DataVal | DataVal | AC500_Dia- <br> gTypes.DIAG_VAL_TYPE | Variable to write <br> data to |
| Inout | DataTxt | DataTxt | AC500_Dia- <br> gTypes.DIAG_TXT_TYPE | Variable to write <br> text to |


| Example | System diagnosis: get numeric values and text for first diagnosis message |  |
| :---: | :---: | :---: |
|  | VAR <br> fbDiag: AC500_Diag.Diag; eErrorID_First: AC500_DiagTypes.ERROR_ID; sFirstVal : AC500_DiagTypes.DIAG_VAL_TYPE; sFirstTxt:AC500_DiagTypes.DIAG_TXT_TYPE; <br> END VAR | ```(* instance of FB Diag *) (* return value *) (* structure of returned values *) (* structure of returned text *)``` |
|  | eErrorID_First := fbDiag.GetFirstValAndTxt(DataVal := sFirstVal, DataTxt := sFirstTxt); |  |

Example Online mode: battery empty or missing

| 㝣 PLC_PRG_1 $\times$ |  |  |
| :---: | :---: | :---: |
| PLC_AC500_V3.Application.PLC_PRG_1 |  |  |
| Expression $\pm \text { fbDiag }$ | Type AC500_Diag.Diag | Value |
| ( eErrorID_First | ERROR_ID | NO_ERROR |
| - sFirstVal | AC500_Diag.AC500... |  |
| - dtTimestamp | DATE_AND_TIME | DT\#1970-1-1-0:0:15 |
| - uiMs | UINT | 356 |
| - eClass | TECLASS | eDiagClass_4_Warning |
| - szDevice | STRING(80) | 'PLC_AC500_V3' |
| - EEvent | TEEVENT | eDiagEvent_Occurred |
| - eHwinterfac... | TEHWID | eDiagHwid_CPU |
| - dwSubSyste... | DWORD | 369098752 |
| - dwadditional | DWORD | 0 |
| - udErrorCode | UDINT | 8 |
| - uiSizeExtDiag | UINT | 0 |
| $\pm$ hSource | POINTER TO BYTE | 16\#008703D0 |
| $\pm$ pConn | POINTER TO IoStan... | 16\#B134A1D4 |
| - sFirstTxt | AC500_Diag.AC500... |  |
| - szDiag | STRING(512) | '1970-01-01;00:00:15,356;E4;PLC_AC500_V3;8;Battery Device Empty or missing' |
| - szColumn | STRING(1) | ';' |

I| eErrorID_First NO_ERROR := fbDiag.GetFirstValAndTxt(DataVal := sFirstVal, DataTxt := sFirstTxt);

Method Get-xxx-ValAndTxtExt / DiagGet-xxx-ValAndTxtExt: numeric values, extended numeric values and text
-xxx- = First, Next, Last, Prev. Example: GetLastValAndTxtExt, DiagGetFirstValAndTxtExt

| Scop <br> e | Name for device <br> diagnosis | Name for device <br> diagnosis | Type | Initial | Comment |
| :--- | :--- | :--- | :--- | :--- | :--- |
| Retur <br> $n$ | Get-xxx-Val- <br> AndTxtExt | DiagGet-xxx-Val- <br> AndTxtExt | AC500_Dia- <br> gTypes.ERROR_ID |  |  |
| Inout | DataVal | DataVal | AC500_Dia- <br> gTypes.DIAG_VAL_ <br> TYPE | Variable to write <br> data to |  |


| Scop <br> e | Name for device <br> diagnosis | Name for device <br> diagnosis | Type | Initial | Comment |
| :--- | :--- | :--- | :--- | :--- | :--- |
| Inout | DataTxt | DataTxt | AC500_Dia- <br> gTypes_DIAG_TXT_ <br> TYPE |  | Variable to write <br> text to |
| Input | pExt | pExt | POINTER TO BYTE | 0 | Address of buffer <br> to copy extended <br> data to |
| Input | Size | Size | WORD | Size of buffer to <br> copy extended <br> data to |  |
| Inout | Length | Length | WORD | Size of extended <br> data copied to <br> buffer |  |


| Example | System diagnosis: get numeric values, extended numeric values and text of first diagnosis message |  |
| :---: | :---: | :---: |
|  | VAR |  |
|  | fbDiag: AC500_Diag.Diag; | (* instance of FB Diag *) |
|  | eErrorID_First : AC500_DiagTypes.ERROR_ID; | (* return value *) |
|  | sFirstVal: AC500_DiagTypes.DIAG_VAL_TYPE; | (* structure of returned values *) |
|  | sFirstTxt : AC500_DiagTypes.DIAG_TXT_TYPE; | (* structure of returned text *) |
|  | abyExtData : ARRAY[0...1023] OF BYTE; | (* byte array for extended diagnosis data *) |
|  | wSize : WORD := 1024; | (* Size of buffer to copy extended data to *) |
|  | wLength : WORD; | (* Size of extended data copied to buffer *) |
|  | END_VAR |  |

eErrorID_First := fbDiag.GetFirstValExt(DataVal := sFirstVal, DataTxt := sFirstTxt,
pExt := ADR(abyExtData), Size := wSize, Length := wLength);

## Function DiagValToTxt

Call this function to convert a numeric diagnosis message into a textual one at any time, in case this has not yet been done using a method providing both types when requesting this information.

| Scope | Name | Type | Comment |
| :--- | :--- | :--- | :--- |
| Return | DiagValToTxt | AC500_DiagTypes.ERROR_ID |  |
| Inout | DataInVal | AC500_DiagTypes.DIAG_VAL_TYPE | Variable to convert |
| Inout | DataOutTxt | AC500_DiagTypes.DIAG_TXT_TYPE | Variable to write text to |


| Example | System diagnosis: convert first diagnosis message from numeric value to text |  |
| :---: | :---: | :---: |
|  | VAR <br> eErrorID_ValToTxt : AC500_DiagTypes.ERROR_ID; <br> sFirstVal:AC500_DiagTypes.DIAG_VAL_TYPE; <br> szDataOutTxt:AC500_DiagTypes.DIAG_TXT_TYPE; <br> END VAR | (* return value *) <br> (* structure of returned values *) <br> (* string of converted data *) |

eErrorID_ValToTx := DiagValToTxt(DataInVal := sFirstVal, DataOutTxt := szDataOutTxt);


### 6.9.1.6 Structure of error numbers

### 6.9.1.6.1 Error severity

| Error severity | Type | Description | Example |
| :--- | :--- | :--- | :--- |
| 1 | Fatal errors | Safe operation of the <br> operating system is no <br> longer ensured. | Checksum error in <br> system flash, RAM error |
| 2 | Severe error | The operating system <br> works correctly, but the <br> error-free execution of <br> the user program is not <br> ensured. | Checksum error in user <br> flash, task cycle times <br> exceeded |
| 3 | Minor errors | It depends on the appli- <br> cation whether the user <br> program has to be <br> stopped by the operating <br> system or not. The user <br> decides which reaction is <br> to be done. | Flash memory cannot be <br> programmed, I/O module <br> failed |


| Error severity | Type | Description | Example |
| :--- | :--- | :--- | :--- |
| 4 | Warnings | Errors that occur on <br> peripheral devices or <br> that will have an effect <br> only in the future. The <br> user decides which reac- <br> tions are to be done. | Short circuit in an <br> I/O module, battery <br> empty/not installed |
| 11 | Parameter error | Error occurred during <br> parameter setting | Different I/O devices in <br> PLC configuration and <br> hardware installation |

## Errors with error severitiy 1 - fatal errors

Errors with error severity 1 are not entered in the diagnosis system. These errors do not allow normal operation of the PLC. These errors are detected during PLC start-up and stop the PLC immediately.

Examples are RAM errors or checksum errors when starting the firmware.
Such errors are indicated by rapid flashing of the ERR LED.

### 6.9.1.7 Diagnosis history file

The diagnosis history file "DiagHistory.csv" collects diagnosis information. It is stored in the root directory of the user disk and provides a maximum of 2.000 entries. When 2.000 entries are reached, the oldest entry is overwritten. The maximum size of extended data is 32 bytes.

The diagnosis history can be accessed with the Automation Builder or with the IEC application.
Chapter 6.9.1.4.5 "Diagnosis history" on page 4614

The diagnosis history file must be interpreted according to the device and/or the fieldbus. A demonstration of how to read out and receive diagnosis history messages with the IEC application is given in the application example AC500 V3 - Diagnosis History

Example entry - An entry consists of following data:
Diagnosis his-
tory file

| Name | Type | Comment | Example |
| :--- | :--- | :--- | :--- |
| timestamp | ARRAYDT OF <br> BYTE | RTC time of event in milliseconds <br> consists of diTimestamp in DT <br> format and uiMs milliseconds. <br> See « Chapter 6.9.1.5.2.2 <br> "Structure DIAG_VAL_TYPE" <br> on page 4616 STRUCT. | 1603371910177 |
| event | BYTE | Event type (1=comes, 2=gone). <br> See « Chapter 6.9.1.5.2.6 "Enu- <br> meration teEvent" on page 4618. | 1 |
| class | BYTE | Severity of error event. <br> 乡 Chapter 6.9.1.5.2.5 "Enumera- <br> tion teClass" on page 4618. | 4 |
| compID | UDINT | Component ID |  |


| Name | Type | Comment | Example |
| :--- | :--- | :--- | :--- |
| conn | UDINT | Connector | $0 x b 17777$ ac |
| connldx | UDINT | Connector index | 0 |
| sub | DWORD | SubsystemID: Any number <br> describing detail/location within <br> device, device specific | 369098752 |
| addl | DWORD | AdditionalID: Additional number <br> describing detail/location within <br> device, optional, device specific | 0 |
| error | DWORD | Error code | 9 |
| extended data | ARRAYDT OF <br> BYTE | Extended diagnosis data, max. 32 <br> bytes |  |

With the entries compID, conn and connID, the device generating the event is clearly identified in the device tree.
If the PLC configuration is changed, the values of this entries may be changed also.

Therefore, the diagnosis history will be deleted during each download.

### 6.9.2 Online diagnosis in Automation Builder

### 6.9.2.1 Short description and overview

To use the diagnosis system in Automation Builder, login to the online mode is required
Chapter 6.9.2.2 "Entering/leaving the online mode" on page 4640. The online diagnosis in Automation Builder consists of a set of partly animated, mostly read only views. They can be invoked by a double-click on a project tree element which shows a circle indicating that this element is able to show diagnosis messages $\stackrel{y}{ } \Rightarrow$ Chapter 6.9.2.3 "Project tree in online mode" on page 4641.

Available online diagnosis and statistics:

- Diagnosis messages

When the Automation Builder is switched to online mode, incoming diagnosis messages are displayed as plain-text $\stackrel{\leftrightarrow}{ }{ }^{\circ}$ Chapter 6.9.1.4 "Diagnosis in Automation Builder" on page 4611.

- CPU/PLC diagnosis

² Chapter 6.9.2.4 "CPU diagnosis views" on page 4645.

- I/O module diagnosis
y Chapter 6.9.2.5 "Live values in views with I/O components" on page 4649.
- Communication module and fieldbus diagnosis

² Chapter 6.9.2.6 "Communication module and fieldbus diagnosis" on page 4649

- Diagnosis in IEC application
«2 Chapter 6.9.1.5 "Diagnosis in IEC application" on page 4615
For information on the disk status, diagnosis information can be read out with the function blocks PmDiskStatus and PmDiskLifetimeUsed. $\Leftrightarrow$ Chapter 6.8.2.2.1.4 "Health monitoring" on page 4424


### 6.9.2.2 Entering/leaving the online mode

Prerequisite: Set the gateway before entering the online mode. $\Leftrightarrow$ Chapter 6.3.2.15 "Gateway configuration" on page 1593

Enter the online Right-click the "Application" node and select "Login". mode

The Automation Builder project login to online mode updates the latest changes of the project.

The online mode can be entered or left for each PLC in the project separately.

Leave the online Right-click the "Application" node and select Logout. mode

When online mode is active, a thread is running on Automation Builder project which sends cyclically a message to the PLC and expects a response. If the PLC does not respond, the online mode is left programmatically.

### 6.9.2.3 Project tree in online mode

When Automation Builder enters the online mode internally, it shows the state of all configured communication modules.

The connection status can be recognized by a symbol in the device tree:

- Device without diagnosis messages
- 3 Device with diagnosis messages or device diagnosis is disabled $\Leftrightarrow$ Chapter 6.9.2.4.6 "Device diagnosis" on page 4648
- Device without diagnosis messages, but with diagnosis messages on at least one device in the branch below
- (4. Device with diagnosis messages and with diagnosis messages on at least one device in the branch below
- $\boldsymbol{\Delta}$ Device does not respond to identification message and is not available for online connection



## The identification is done in online mode.

- Double-click an element of the device-tree and select "Status" tab. Diagnosis information will be available.



## Diagnosis

 descriptions

The user will be notified in the device tree with an exclamation mark beside the device having diagnosis messages. The diagnosis messages are provided in the "Diagnosis" tab.
Alarms will be presented with a thunderbolt in the first column of the diagnosis grid.
Acknowledging 1. Stop diagnosis refreshment by clicking [Stop refresh].
an alarm
2. Select one or more alarms and click [Acknowledge selected alarms].


Some diagnosis messages contain additional data. Click [View] button to see the additional diagnosis (in hex) for further analysis. If [View] button is not available, no additional data is available for this diagnosis message.
You can copy the additional data to the clipboard with [CTRL] + [C].


When building an IEC application in Automation Builder, diagnosis text lists will be generated and added to the device tree below the diagnosis folder. These text lists contain the device type specific diagnosis texts which are used by the diagnosis functions in the PLC application to show corresponding texts for error numbers.


The diagnosis text lists will only be downloaded to the PLC when a visualization is present in the project.

The text lists will be downloaded automatically to the PLC with the visualization.
If there is a problem with downloading the text lists, make sure that the settings are correct:

1. Right-click on a text list and select "Properties".
2. Open the "Text List" tab. The check box "Download by visualization" has to be selected.


### 6.9.2.4 CPU diagnosis views

### 6.9.2.4.1 Version information

Information on the firmware versions of the processor modules or communication modules, is provided on the "Version information" tab.

Remarks:

- The "Version information" tab displays the version identified on the device and the version provided with Automation Builder.
- The firmware on the devices must match to the Automation Builder supported versions. It might be required to upgrade or downgrade the current firmware for using the PLC with the current Automation Builder.
- The "Required Version" might contain more than one user selectable version (drop-down list), this is indicated by an asterisk after the file name.


State icons

| (8) | Firmware version on device matches version supplied with <br> Automation Builder. |
| :--- | :--- |
|  | Firmware version (or type) on device is different from version supplied <br> with Automation Builder. Upgrade/downgrade to version supplied with <br> Automation Builder is recommended. |
| No icon | Only for communication modules if CPU firmware must be updated <br> first. This happens when CPU firmware has version below 2.5.0.0. <br> Firmware version (or type) on device is different from version supplied <br> with Automation Builder. Upgrade/downgrade to version supplied with <br> Automation Builder is recommended. |
|  | Identified device is different from configured device, thus no firmware <br> update is possible. Happens only for Communication Modules. |
|  | Firmware of device is not updateable or no newer firmware than the <br> initial version is available. |

[Update Firmware] to download the new firmware is only enabled if there is updateable firmware.

### 6.9.2.4.2 Statistics



The "Statistics" tab shows the following information:

- Date and time: The actual date and time of the PLC is shown. It can be set or synchronized with the date/time of the PC via "Set PLC Date \& Time" button.
- Overview resource usage: This tab shows all the required information (it is collected at latest when the command "Generate Code" is executed, some of the information is not available before then.)
For the limitation "User program code and data" a [Details] button will be available. Clicking this button will open a modal window showing a more detailed view of the memory usage.

You can view the PLC log in this tab．It lists the events that were recorded on the target system． This concerns：
－Events during the startup and shutdown of the system（components loaded，with version）
－Application download and loading of the boot application
－Custom entries
－Log entries from I／O drivers
－Log entries from data sources

| Offline logging | $\square$ Default settings <br> The PLC also records actions that are not related to the connection with the controller． |
| :---: | :---: |
| UTC time | Standard setting；the time stamp is converted to the local time on the com－ puter as indicated by the time zone of the operating system． <br> The time stamp of the runtime system is displayed． |
| Severity | There are four categories for the severity of the event： <br> －© ：Message <br> －© ：Warning <br> －© Error <br> －© Debugging <br> You can show or hide each category with the help of the corresponding button in the bar above the list．Each button shows the number of log entries of the category concerned． |
| Time stamp | Date and time（example：12－01－2007 09：48） |
| Description | Description of the event，for example Import function failed of |
| Component | Name of the runtime system component concerned，e．g．CmpApp |
| Drop－down list with component names | The log list displays only events that concern the selected component |
| Logger | Drop－down list with all available recordings．The standard setting is the＜Default Logger＞specified by the target system，at present identical to＇PlcLog＇for the CODESYS runtime system |
| 6 | Refreshes the log list |
| 10 | Exports the list contents to an XML file．You can select the file name and storage directory． |
| 第 | Imports a log list from an XML file．The list is then displayed in a separate window． |
| $x$ | The displayed log list is emptied，i．e．all entries are deleted． |

## 6．9．2．4．4 PLC shell commands

The PLC shell is used for requesting specific information from the controller．By entering a device－specific command the response is returned in a result window．The PLC shell can be issued without login．
Proceed as fol－1．Ensure the gateway is configured properly and a connection to the controller can be lows： established．

2．In Automation Builder double－click the PLC node and open the tab＂PLC Shell＂．
3．Enter＂？＂in the command line of the tab window．All available PLC commands are listed．

If the gateway is able to establish a connection to the controller, an online connection to the PLC is opened automatically.

```
The commands listed in online mode can differ from the commands shown when pressing the button [...] as Automation Builder version and firmware version can differ.
ⓨ Chapter 6.4.1.21.2.8.11 "Tab 'PLC Shell"" on page 2441
« Chapter 6.2.6 "Version information" on page 1384
Chapter 6.3.1.4 "Firmware identification and update" on page 1419.
```


### 6.9.2.4.5 Status

This tab displays status information, for example 'Running' or 'Stopped', and specific diagnosis messages from the respective device, also information about the card used and the internal bus system.

| Communication modules | n/a |
| :---: | :---: |
| Interfaces | n/a |
| //0-Bus | n/a |
| CPU-Parameters | n/a |
| PM5630-2ETH | n/a |

### 6.9.2.4.6 Device diagnosis

Each node in the device tree has a diagnosis view, which displays the diagnosis messages for this device only.
The message consists of:

- Type $\Perp^{\circledR}$ Chapter 6.9.1.4.5 "Diagnosis history" on page 4614
- Timestamp in date and time YYYY-MM-DD hh:mm:ss.ms
- Error severity
- Error code
- Diagnosis description
- Additional data

To view the diagnosis message:

1. Double-click on a device.
2. Select the tab "Diagnosis".

## Example $\quad$ Battery empty or missing.

| us 圌 PLC_AC500_V3 x 屏 DI524 |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Start refresh Stop refresh Acknowledge Selected Alarms |  |  |  |  |  |
| Type | Device | Timestamp | Severity | Error Code | Description |
| + | PLC_AC500_V3 | 1970-01-01; 12:00:13.916 | 4 | 8 | Battery, Device, Empty or missing |

## Example Wrong module configured on I/O bus.

| [7] PLC_AC500_V3 四 DI524 x |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Start refresh Stop refresh Acknowledge Selected Alarms |  |  |  |  |  |  |
| Type | Device | Timestamp | Severity | Error Code | Description | Additi |
| + | DI524 | 1970-01-01; 12:16:30.633 | 11 | 16158 | Module 1, Ty |  |

Device diagnosis is disabled by default.
To enable/disable device diagnosis:

1. Double-click on the PLC.
2. Select the tab "PLC Settings".
3. Under "Additional Settings" enable/disable "Diagnosis for devices".
$\Rightarrow$ When the device diganosis is disabled, this symbol will be displayed in the device tree and no diagnosis messages will be shown.

### 6.9.2.5 Live values in views with I/O components

"//O mapping list" tab: In online mode, all Automation Builder views, which contain I/O component mapping tables, show animated live values which are updated every second.

| D0573 Parameters <br> D0573 1/O Mapping | $\checkmark$ Tool Bar |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | $\begin{aligned} & \text { Object Name } \\ & \text { D0573 } \end{aligned}$ | Variable | Channel | Address | Current Value |  | Description | Termina |
|  |  |  | Relay outputs NOO - NO\%QWO |  | 65280 | WORD |  |  |
| I/O mapping list | $\begin{aligned} & \mathrm{DO} 573 \\ & \mathrm{DO573} \end{aligned}$ | test1 | Relay outputs NO0 - NO\%QB0 |  | 255 | BYTE |  |  |
|  |  |  | Relay output NOO | \%QX0.0 | TRUE | B00L |  | 1 |
| Information | D0573 | test2 | Relay output NO1 | \%Q×0.1 | IRUE | BOOL |  | 2 |
|  | 00573 | test3 | Relay output NO2 | \%QX0.2 | IRUE | BOOL |  | 3 |
|  | D0573 | test4 | Relay output NO3 | \%Q×0.3 | IRUE | BOOL |  | 4 |
|  | 00573 | test5 | Relay output NO4 | \%QX0.4 | IRUE | B00L |  | 5 |
|  | D0573 | test6 | Relay output NO5 | \%QX0.5 | IRUE | B00L |  | 6 |
|  | 00573 | test7 | Relay output NO6 | \%QX0.6 | IRUE | B00L |  | 7 |
|  | 00573 | test8 | Relay output NO7 | \%QX0.7 | IRUE | BOOL |  | 8 |
|  | 00573 |  | Relay outputs NO8 - NO | \%QB1 | 0 | BYTE |  |  |
|  | D0573 |  | Relay output NO8 | $\%$ QX1.0 | FALSE | BOOL |  | 10 |
|  | 00573 |  | Relay output NO9 | \%QX1.1 | FALSE | B00L |  | 11 |
|  | 00573 |  | Relay output NO10 | $\% \mathrm{QX1.2}$ | FALSE | B00L |  | 12 |
|  | 00573 |  | Relay output NO11 | \%QX1.3 | FALSE | B00L |  | 13 |
|  | 00573 |  | Relay output NO12 | $\%$ \%X1.4 | FALSE | B00L |  | 14 |
|  | 00573 |  | Relay output NO13 | $\% \mathrm{Q} \times 1.5$ | FALSE | B00L |  | 15 |
|  | 00573 |  | Relay output NO14 | \%QX1.6 | FALSE | B00L |  | 16 |
|  | D0573 |  | Relay output NO15 | \%QX1.7 | FALSE | B00L |  | 17 |

### 6.9.2.6 Communication module and fieldbus diagnosis

### 6.9.2.6.1 Fieldbus commissioning

## General

Common online diagnosis views for all netX-based communication modules (e. g. CM579ETHCAT, CM579-PNIO) can be accessed whenever the related PLC is in online mode.
${ }^{\text {² }}$ ) Chapter 6.9.2.2 "Entering/leaving the online mode" on page 4640

## Master/controller modules

Master/controller modules like CM579-ETHCAT or CM579-PNIO, provide the following diagnosis views:

- "Diagnostics main": provides diagnosis messages which are common for all protocols (e.g., protocol state and error)
- "Diagnostics live list": provides a list of connected slaves/devices and their state " ${ }^{4}$ Chapter 6.9.2.6.1.4 "PROFINET scan and comparison view" on page 4651
- "Diagnostics eventlog": provides diagnosis messages from the master/controller and its connected slaves/devices



## Slave/device communication modules

Diagnosis views for slave/device communication modules like CM589-PNIO:

- "Diagnostics main": provides diagnosis messages which are common for all protocols
- "Diagnostics details": provides protocol specific diagnosis messages



## PROFINET scan and comparison view

## PNIO_Controller

1. After going online, double-click on "PNIO_Controller (PROFINET-IO-Controller)" in the device tree.
$\Rightarrow$ The editor "PNIO_Controller" is displayed.
2. Select tab "Diagnostics live list" and click [Scan] to find all hardware devices that exist. $\Rightarrow$ The found devices are listed in a table.
3. Click [Compare] to compare the found hardware I/O devices with the current project configuration.

Unknown hard- If any I/O hardware device is unknown:

- The devices will be marked with a red exclamation mark.
- A message box will be appear for each unknown device.
- Automation Builder generates a message with information about its vendor ID and device ID.



## Comparison view

1. To display the comparison view, install the device description for the unknown device.
2. After installing the device description, click [Scan] and click [Compare].

$\Rightarrow$ The message box informs you, that the application will go offline to display the comparison view.
3. Click [Yes].
$\Rightarrow$ The "Project Comparison - Differences" tab displays the difference between the PROFINET configuration in Automation Builder (left side) and the real hardware configuration (right side).

4. Click [Accept Single] to accept only a part of the differences or [Accept Block] to accept all differences.
$\Rightarrow$ After clicking on the Button [Accept Single] or [Accept Block] the found devices will be moved from the right side to the left side.
5. Close tab "Project Comparison - Differences".
$\Rightarrow$ A message will be displayed to ask if you want to commit the new changes into project.


6．Click［Yes］．
$\Rightarrow$ The changes will be saved and the devices will be added to the project．

```
Devices
```



```
    -T PLC_AC500_V3 (PM5630-2ETH - TB5620-2ETH)
    - -1,\ll PLC Logic
        - Application
            +
            Diagnosis
                            Library Manager
                i] PLC_PRG (PRG)
                Task Configuration
                Task
                    果 PLC_PRG
```



```
    +. . Interfaces
    = 链 Extension_Bus
        的目 CM579_PNIO (CM579-PNIO)
            - 旣 PNIO_Controller (PROFINET-IO-Controller)
            f- 帽 CI501_PNIO_Device (CI501-PNIO-Device)
            + CI502_PNIO_Device (CI502PNIO-Device)
            # + CI502_PNIO_Device_1 (CI502-NNIO-Device)
            = 的自 CM589_PNIO (CM589-NIO)
            + 的自 PNIO_Device (PROFINET-IO-Device)
```


## 6．9．2．6．2 CI52x Modbus diagnosis

1．Double－click node＂CI52x＿MODTCP＂in the device tree．
2．Select＂CI52x Diagnosis＂tab．
$\Rightarrow$ The button［Get Diagnosis］appears in the tab view．
3．Click on the button［Get Diagnosis］．
$\Rightarrow$ One of the following use cases will be displayed：
－Device not connected $\Leftrightarrow$＂Device not connected＂on page 4654
－No Errors on the device ${ }^{4}$＂No errors on the device＂on page 4654
－Diagnosis list $\xi^{4}$＂Diagnosis list＂on page 4654

Device not con- If there is no device connected to the project, the following dialog will be displayed:

## nected



1. Select tab "Connection Settings" and enter the IP address for the device.
2. Click again button [Get Diagnosis].

No errors on the If there are no errors on the device the following dialog will be displayed:

## device

Automation Builder

OK

Diagnosis list If the device is not correctly configured the errors will be displayed with "Error Code" and "Code Description".

|  | Error <br> Code | Code Description |
| :--- | :--- | :--- |
| - | 234946507 | Process voitage too low |
| 234946541 | Process voitage switched off |  |
| 234881072 | Module number: 1: Measurement overflow or cut wire at the analog input |  |
|  |  |  |

### 6.9.2.6.3 EtherCAT diagnosis

Due to the nature of the EtherCAT bus, diagnostics cannot be standardized and must be individually adapted to the requirements of the PLC system.

A guideline for EtherCAT diagnosis during commissioning and operation of the AC500 V3 series in combination with Automation Builder diagnosis and the IEC programming is given in the application note EtherCAT diagnosis guideline.

### 6.9.3 Diagnosis messages

### 6.9.3.1 Read out the diagnosis messages via display

If the processor module has a display, you can access the diagnosis system via the display and the buttons.
Diagnosis messages can also be accessed within the engineering suite Automation Builder or with the IEC application.
« Chapter 6.9.1.4 "Diagnosis in Automation Builder" on page 4611
*s Chapter 6.9.1.5 "Diagnosis in IEC application" on page 4615


Fig. 365: Example "battery is empty or missing"

Err 1 E4 bat Id 1 8, d1, d2, d3, d4

Total number of active diagnosis messages, here: 1 Error severity, here: 4, and location of error, here: battery In the list of all errors, the position of the error, here: first error message Error code, here: 8, and system information to identify the error with the help of the diagnosis lists
« E2 diagnosis messages
« E3 diagnosis messages
を E4 diagnosis messages
« E11 diagnosis messages

## 6．9．3．2 Error severity E2：Severe errors

Table 860：Error severity E2：Severe errors

| Error severity | Location | Error code | d1 | d2 | d3 | d4 | Description | Remedy |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | 星：晛 | 或d1 22 |  |  | 島：04 0⿴囗木 |  |  |
| E2 | $\begin{aligned} & \hline \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 2 | 14 | 0 | 0 | 0 | Resource failure |  |
| E2 | $\begin{aligned} & \hline \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 3 | 14 | 0 | 0 | 0 | Timeout |  |
| E2 | CPU | 11 | 0 | 0 | 0 | 0 | Type scaling on PLC startup failed |  |
| E2 | FLd （flash－ disk） | 12 | 19 | 0 | 0．．n | 0．．n | Flash memory is temporary protected （read only，panic mode） |  |
| E2 | iob （I／O bus） | 17 | 14 | 0 | 0 | 0 | Error setting I／O bus master parameter |  |
| E2 | $\begin{aligned} & \text { diS } \\ & \text { (display) } \end{aligned}$ | 23 | 21 | 0 | 0 | 0 | Wrong version of display firmware |  |
| E2 | CPU | 27 | 0 | 2 | 0 | 0 | Failed to set param－ eter＂Reboot at pow－ erfail＂ |  |
| E2 | CPU | 29 | 0 | 0 | 0 | 0 | Combination of plugged option boards is invalid |  |
| E2 | CPU | 49 | 0 | 0 | 0 | 0 | Restore failed |  |
| E2 | $\begin{aligned} & \hline \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 8482 | Module number | 0 | － | 0 | Timeout，while initi－ alizing |  |
| E2 | iob （I／O bus） | 9249 | Module number | 0 | － | 0 | Timeout while waiting for Reset |  |
| E2 | $\begin{aligned} & \hline \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 9258 | Module number | 0 | － | 0 | Breakdown，commu－ nication lost |  |
| E2 | iob （I／O bus） | 16129 | 0 | 0 | 0 | 0 | Severe error，see log |  |
| E2 | $\begin{aligned} & \hline \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 16130 | 0 | 0 | 0 | 0 | Fatal error，see log |  |
| E2 | $\begin{aligned} & \hline \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 16194 | 0 | 0 | 0 | 0 | Fatal error，not run－ ning any more |  |

6．9．3．3 Error severity E3：Minor errors
Table 861：Error severity E3：Minor errors

| Error severity | Location | Error code | d1 | d2 | d3 | d4 | Description | Remedy |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | Fe，E］［PU］ | 氟：8 |  |  | 或d3 0 0 目 |  |  |  |
| E3 | iob （I／O bus） | 3 | Module number | 1 | Channel number | 0 | Discrepancy time expired | Check dis－ crepancy time value， channel wiring and sensor． |
| E3 | FLd （flash－ disk） | 10 | 19 | 5 | 0．．．n | 0 | Medium has almost used its complete spare capacity or is already dead，action required |  |
| E3 | iob （I／O bus） | 12 | Module number | 1 | Channel number | 0 | Test pulse error | Check wiring and sensor |
| E3 | $\begin{aligned} & \hline \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 13 | Module number | 1 | Channel number | 0 | Test pulse cross－talk error | Check wiring and sensor．If this error persists， replace I／O module． Contact ABB tech－ nical sup－ port． |
| E3 | iob （I／O bus） | 18 | Module number | 1 | Channel number | 0 | Test error |  |
| E3 | diS （display） | 22 | 21 | 0 | 0 | 0 | Error at initialization of display |  |
| E3 | $\begin{aligned} & \hline \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 25 | Module number | 1 | Channel number | 0 | Stuck－at error | Check I／O module wiring． Restart I／O module，if needed．If this error persists， replace I／O module． |
| E3 | iob （I／O bus） | 28 | Module number | 1 | Channel number | 0 | Cross－talk error | Check I／O module wiring． Restart I／O module，if needed．If this error persists， replace I／O module． |
| E3 | CPU | 44 | 0 | 7 | 0 | 0 | Loading boot project failed due to failing load of retain data |  |


| Error severity | Location | Error code | d1 | d2 | d3 | d4 | Description | Remedy |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | 或：8目 |  | 或：d2 0 |  | 戓：84 0⿴囗 |  |  |
| E3 | CPU | 45 | 0 | 7 | 0 | 0 | Loading boot project failed due to mis－ match of retain data |  |
| E3 | CPU | 46 | 0 | 7 | 0 | 0 | Loading boot project failed due to insuf－ ficient license for application |  |
| E3 | CPU | 47 | 0 | 7 | 0 | 0 | Loading boot project failed due to unspe－ cified reason |  |
| E3 | CPU | 48 | 0 | 0 | 0 | 0 | Backup failed |  |
| E3 | FLd （flash－ disk） | 54 | 19 | 5 | 0 | 0 | Root partition has 1\％or less free space left |  |
| E3 | $\begin{aligned} & \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 273 | Module number | 1 | Channel number | 0 | Test error |  |
| E3 | $\begin{aligned} & \text { iob } \\ & \text { (//O bus) } \end{aligned}$ | 311 | Module number | 1 | Channel number | 0 | Value difference too high | Adjust toler－ ance window for channels． Check channel wiring and sensor con－ figuration． |
| E3 | $\begin{aligned} & \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 524 | Module number | 1 | Channel number | 0 | Stuck－at error |  |
| E3 | $\begin{aligned} & \hline \text { iob } \\ & (I / O \text { bus }) \end{aligned}$ | 525 | Module number | 1 | Channel number | 0 | Readback error | Check I／O module wiring． Restart I／O module，if needed．If this error persists， replace I／O module． |
| E3 | $\begin{aligned} & \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 530 | Module number | 1 | Channel number | 0 | Cross－talk error | Check I／O module wiring． Restart I／O module，if needed．If this error persists， replace I／O module． |
| E3 | $\begin{aligned} & \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 540 | Module number | 1 | Channel number | 0 | Test error |  |
| E3 | $\begin{aligned} & \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 555 | Module number | 1 | Channel number | 0 | Internal error |  |
| E3 | iob （I／O bus） | 1037 | Module number | 1 | Channel number | 0 | Test error |  |


| Error severity | Location | Error code | d1 | d2 | d3 | d4 | Description | Remedy |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | 星: 8 ${ }^{\text {a }}$ |  | 氟:d2 0 ${ }^{\text {a }}$ |  |  |  |  |
| E3 | iob (I/O bus) | 8480 | Module number | 0 | - | 0 | Not supported protocol variant |  |
| E3 | $\begin{aligned} & \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 8707 | Module number | 0 | - | 0 | PROFIsafe watchdog timed out |  |
| E3 | $\begin{aligned} & \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 8708 | Module number | 0 | - | 0 | Overvoltage |  |
| E3 | $\begin{aligned} & \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 8711 | Module number | 0 | - | 0 | Undervoltage |  |
| E3 | $\begin{aligned} & \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 8722 | Module number | 0 | - | 0 | Internal error |  |
| E3 | $\begin{aligned} & \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 8723 | Module number | 0 | - | 0 | Checksum error has occured in iParameters |  |
| E3 | $\begin{aligned} & \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 8724 | Module number | 0 | - | 0 | PROFIsafe communication error |  |
| E3 | $\begin{aligned} & \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 8732 | Module number | 0 | - | 0 | Internal error |  |
| E3 | $\begin{aligned} & \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 8747 | Module number | 0 | - | 0 | Internal runtime error |  |
| E3 | $\begin{aligned} & \text { iob } \\ & \text { (//O bus) } \end{aligned}$ | 8961 | Module number | 0 | - | 0 | Wrong parameter value, check configuration |  |
| E3 | $\begin{aligned} & \text { iob } \\ & \text { (//O bus) } \end{aligned}$ | 8979 | Module number | 0 | - | 0 | Checksum error has occured in iParameter or F-Parameters |  |
| E3 | $\begin{aligned} & \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 8986 | Module number | 0 | - | 0 | Invalid configuration | Check modules and parameterization |
| E3 | $\begin{aligned} & \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 8988 | Module number | 0 | - | 0 | F-Parameter configuration and address switch value do not match. | Check I/O module FParameter configuration and module address switch value. |
| E3 | iob (I/O bus) | 16131 | Module number | 0 | - | 0 | Timeout | Replace I/O module |
| E3 | $\begin{aligned} & \hline \text { iob } \\ & (I / O \text { bus }) \end{aligned}$ | 16137 | Module number | 0 | - | 0 | Overflow diagnosis buffer | Restart |
| E3 | $\begin{aligned} & \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 16138 | Module number | 0 | - | 0 | Non-safety I/O: Voltage overflow on outputs (above UP3 level), <br> Safety I/O: Process voltage too high | Check terminals/ process voltage |


| Error severity | Location | Error code | d1 | d2 | d3 | d4 | Description | Remedy |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Al，E］［PU |  | 氞：8 | 5：d 25 目 |  | 包d3 0 |  |  |  |
| E3 | $\begin{aligned} & \hline \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 16139 | Module number | 0 | － | 0 | Process voltage UP or UP3 too low | Check process voltage |
| E3 | $\begin{aligned} & \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 16146 | Module number | 0 | － | 0 | Plausibility check failed（iParameter） | Check con－ figuration |
| E3 | $\begin{aligned} & \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 16147 | Module number | 0 | － | 0 | Checksum error | Non－safety I／O：Replace I／O module <br> Safety I／O： <br> Check safety con－ figuration and CRCs for I －and F － Parameters． |
| E3 | $\begin{aligned} & \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 16148 | Module number | 0 | － | 0 | PROFIsafe commu－ nication error | Restart I／O module．If this error persists， contact ABB technical support． |
| E3 | iob （I／O bus） | 16153 | Module number | 0 | － | 0 | PROFIsafe watchdog timed out | Restart I／O module．If this error persists， increase PROFIsafe watchdog time． |
| E3 | $\begin{aligned} & \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 16154 | Module number | 0 | － | 0 | Parameter error | Check con－ figuration． |
| E3 | iob （I／O bus） | 16156 | Module number | 0 | － | 0 | F－Parameter config－ uration and address switch value do not match． | Check I／O module F－ Parameter configura－ tion and module address switch value． |
| E3 | $\begin{aligned} & \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 16164 | Module number | 0 | － | 0 | Internal data inter－ change failure | Replace I／O module |
| E3 | iob （I／O bus） | 16168 | Module number | 0 | － | 0 | Different hard－／firm－ ware versions in the module | Replace I／O module |
| E3 | $\begin{aligned} & \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 16171 | Module number | 0 | － | 0 | Internal error | Replace I／O module |
| E3 | $\begin{aligned} & \hline \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 16175 | Module number | 0 | － | 0 | Sensor voltage too low |  |
| E3 | $\begin{aligned} & \hline \text { iob } \\ & \text { (l/O bus) } \end{aligned}$ | 16128 | 0 | 0 | 0 | 0 | Failed Max Wait Run |  |

6．9．3．4 Error severity E4：Warnings
Table 862：Error severity E4：Warnings

| Error severity | Location | Error code | d1 | d2 | d3 | d4 | Description | Remedy |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | Fe，E］［PU］ | 易：8 | 或d1 22 目 | 気：d2 0⿴囗 | 或d3 0 0 目 |  |  |  |
| E4 | bAt （battery） | 8 | 22 | 0 | 0 | 0 | Empty or missing |  |
| E4 | FLd （flash－ disk） | 9 | 19 | 5 | 0．．．n | 0 | Medium has used $80 \%$ of its spare capacity |  |
| E4 | FLd （flash－ disk） | 53 | 19 | 5 | 0 | 0 | Root partition has $15 \%$ or less free space left |  |
| E4 | iob （I／O bus） | 257 | Module number | 1 | Channel number | 0 | Wrong measure－ ment，false tempera－ ture at the compen－ sation channel |  |
| E4 | $\begin{aligned} & \hline \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 258 | Module number | 1 | Channel number | 0 | AI531：wrong meas－ urement，potential difference is too high； <br> CD522：PWM duty cycle out of duty area |  |
| E4 | iob （I／O bus） | 260 | Module number | 1 | Channel number | 0 | Measurement over－ flow | Check channel wiring and sensor power supply． |
| E4 | $\begin{aligned} & \hline \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 263 | Module number | 1 | Channel number | 0 | Measurement under－ flow at analog input | Check channel wiring and sensor power supply． |
| E4 | iob （I／O bus） | 266 | Module number | 1 | Channel number | 0 | Short circuit and cut wire or＂out of range＂ |  |
| E4 |  | 267 | Module number | 1 | Channel number | 0 | Output／process voltage to small／low． |  |
| E4 | iob （I／O bus） | 302 | Module number | 1 | Channel number | 0 | External voltage feeded at the output clamps． |  |
| E4 | iob （I／O bus） | 303 | Module number | 1 | Channel number | 0 | Short circuit at the analog input | Check channel wiring |
| E4 | iob （I／O bus） | 304 | Module number | 1 | Channel number | 0 | Analog value over－ flow or broken wire at an analog input． |  |
| E4 | iob （I／O bus） | 530 | Module number | 1 | Channel number | 0 | Internal fuse at 0 V is defect． 0 V not connected with GND． |  |


| Error severity | Location | Error code | d1 | d2 | d3 | d4 | Description | Remedy |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | 風 明 | \＃idi 22 目 | 島，d2 0⿴囗⿱一一口1 | 島：d3 0⿴囗木大 |  |  |  |
| E4 | iob （I／O bus） | 559 | Module number | 1 | Channel number | 0 | Short circuit at the digital output | Check channel wiring |
| E4 | $\begin{aligned} & \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 772 | Module number | 1 | Channel number | 0 | Measurement over－ flow at an analog output | Check channel wiring |
| E4 | $\begin{aligned} & \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 775 | Module number | 1 | Channel number | 0 | Measurement under－ flow at analog output | Check channel wiring |
| E4 | $\begin{array}{\|l\|l} \hline \text { iob } \\ \text { (I/O bus) } \end{array}$ | 796 | Module number | 1 | 0 | 0 | Different configura－ tion |  |
| E4 | $\begin{aligned} & \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 8482 | Module number | 0 | － | 0 | Timeout while waiting for ready status |  |
| E4 | iob （I／O bus） | 8483 | Module number | 0 | － | 0 | Timeout during parameterization |  |
| E4 | $\begin{aligned} & \hline \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 9480 | Module number | 0 | － | 0 | I／O module removed from hot swap ter－ minal unit or defec－ tive module on hot swap terminal unit． | Plug I／O module， replace I／O module |
| E4 | iob （I／O bus） | 9480 | Module number | 0 | 0 | 0 | Module removed from hot swap ter－ minal unit |  |
| E4 | iob （I／O bus） | 9500 | Module number | 0 | － | 0 | Wrong I／O module replugged on hot swap terminal unit | Remove wrong I／O module and plug pro－ jected I／O module |
| E4 | iob （I／O bus） | 9514 | Module number | 0 | － | 0 | No communication with I／O module on hot swap terminal unit | Replace I／O module |
| E4 | iob （I／O bus） | 9526 | Module number | 0 | － | 0 | I／O module does not support hot swap | Power off system and replace I／O module |
| E4 | iob （I／O bus） | 9526 | Module number | 0 | 0 | 0 | Module on hot swap terminal unit does not support hot swap functionality |  |
| E4 | iob （I／O bus） | 9736 | Module number | 0 | － | 0 | Hot swap terminal unit required，but not found |  |
| E4 | iob （I／O bus） | 9764 | Module number | 0 | 0 | 0 | Defective hot swap terminal unit |  |
| E4 | iob （I／O bus） | 9770 | Module number | 0 | － | 0 | No communication with hot swap ter－ minal unit | Restart，if error per－ sists replace terminal unit |


| Error severity | Location | Error code | d1 | d2 | d3 | d4 | Description | Remedy |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Heg［PU | H，E3［PU ${ }^{\text {a }}$ | 気：8 | A．dl 22 | 弐：d2 0 | 嵒：dヨ 0 | 盏：14 0 |  |  |
| E4 | iob （I／O bus） | 16172 | Module number | 0 | － | 0 | Has not passed fac－ tory test |  |
| E4 | iob <br> （I／O bus） | 16173 | Module number | 0 | － | 0 | No process voltage UP or UP3 | Check process voltage |

## 6．9．3．5 Error severity E11：Parameter errors

Table 863：Error severity E11：Parameter errors

| Error severity | Location | Error code | d1 | d2 | d3 | d4 | Description | Remedy |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | 武：日目 | ： | －${ }_{\text {－d2 }}$ | ㅁ．0．dヨ 0 | 戓：d4 0 |  |  |
| E11 | CPU | 18 | 0 | 2 | 0 | 0 | At least one param－ eter not found |  |
| E11 | CPU | 19 | 0 | 2 | 0 | 0 | Unable to read at least one parameter value |  |
| E11 | CPU | 20 | 0 | 2 | 0 | 1 （too big）or 2 （too small） | Invalid value of parameter＂LED＂ |  |
| E11 | CPU | 21 | 0 | 2 | 0 | 1 （too big）or <br> 2 （too small） | Invalid value of parameter＂Battery＂ |  |
| E11 | CPU | 24 | 0 | 2 | 0 | 1 （too big）or 2 （too small） | Invalid value of parameter＂Diag－ nosis，stop on error class＂ |  |
| E11 | CPU | 25 | 0 | 2 | 0 | 1 （too big）or 2 （too small） | Invalid value of parameter＂Diag－ nosis，add Plc name to node name＂ |  |
| E11 | CPU | 26 | 0 | 2 | 0 | 1 （too big）or 2 （too small） | Invalid value of parameter＂Reboot at powerfail＂ |  |
| E11 | CPU | 28 | 0 | 2 | 0 | 1 （too big）or <br> 2 （too small） | Value of parameter ＂Missed Cycle Task behaviour＂is not in range |  |
| E11 | CPU | 30 | 0 | 2 | 0 | 1 （too big）or <br> 2 （too small） | Value of parameter ＂Priority Scheme＂is not in range |  |


| Error severity | Location | Error code | d1 | d2 | d3 | d4 | Description | Remedy |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  | 盏d1 22 目 |  |  | 或，d4 0 |  |  |
| E11 | CPU | 40 | 0 | 2 | 0 | $\begin{aligned} & \hline 1 \text { (too } \\ & \text { big) or } \\ & 2 \text { (too } \\ & \text { small) } \end{aligned}$ | Value of parameter ＂Reboot on E2＂is not in range |  |
| E11 | CPU | 50 | 0 | 2 | 0 | $\begin{aligned} & 1 \text { (too } \\ & \text { big) or } \\ & 2 \text { (too } \\ & \text { small) } \end{aligned}$ | Invalid value of parameter＂Enable diagnosis history＂ |  |
| E11 | CPU | 51 | 0 | 2 | 0 | 1 （too big）or <br> 2 （too small） | Invalid value of parameter＂Max－ imum number of entries in diagnosis history＂ |  |
| E11 | CPU | 52 | 0 | 2 | 0 | $\begin{aligned} & 1 \text { (too } \\ & \text { big) or } \\ & 2 \text { (too } \\ & \text { small) } \end{aligned}$ | Invalid value of parameter＂Max－ imum size of extended diagnosis data in diagnosis history＂ |  |
| E11 | iob （I／O bus） | 16133 | Module number | 0 | 0 | 0 | Output data size mismatch |  |
| E11 | $\begin{aligned} & \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 16134 | Module number | 0 | 0 | 0 | Input data size mis－ match |  |
| E11 | iob （I／O bus） | 16145 | 0 | 0 | 0 | 0 | Error setting I／O bus master parameter |  |
| E11 | iob （I／O bus） | 16146 | 0 | 0 | 0 | 0 | Failed to start the parameterization of modules |  |
| E11 | iob （I／O bus） | 16147 | Module number | 0 | 0 | 0 | Failed setting parameters |  |
| E11 | $\begin{aligned} & \hline \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 16149 | Module number | 0 | 0 | 0 | No module data |  |
| E11 | iob | 16246 | 0 | 0 | 0 | 0 | Failed to start the parameterization of modules |  |
| E11 | $\begin{aligned} & \hline \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 16247 | 1．． 20 | 0 | 0 | 0 | Failed setting parameters |  |
| E11 | iob （I／O bus） | 16248 | Module number | 0 | 0 | 0 | Failed setting expected module |  |
| E11 | iob （I／O bus） | 16254 | Module number | 0 | 0 | 0 | Size of parameters expected by module differs from size pro－ vided by configura－ tion |  |
| E11 | $\begin{aligned} & \hline \text { iob } \\ & \text { (I/O bus) } \end{aligned}$ | 16158 | Module number | 0 | 0 | 0 | Type of present module does not match configuration |  |


| Error severity | Location | Error code | d1 | d2 | d3 | d4 | Description | Remedy |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | 里：日完 |  | 弐：d2 0 |  | 4：040 0 |  |  |
| E11 | － | 16159 | 0 | 0 | － | 0 | Configured number of modules differs from found ones |  |
| E11 | － | 16160 | 0 | 0 | － | 0 | At least one module failed during configu－ ration |  |

## 6．9．3．6 Communication modules diagnosis

## 6．9．3．6．1 Diagnosis display

Table 864：Diagnosis messages of communication modules or fieldbuses

| Error severity | Location | Description |
| :---: | :---: | :---: |
|  |  |  |
| E－ | CMx（communication module） $x=\text { slot number }$ | To view these diagnosis messages use Automation Builder ＊Chapter 6．9．1．4＂Diagnosis in Automation Builder＂on page 4611 or IEC application Chapter 6．9．1．5＂Diagnosis in IEC application＂ on page 4615. |
| Ex | CAn（CAN interface） |  |
| Ex | Et1 or Et2（Ethernet ETH1 or ETH2） |  |

## 6．9．3．6．2 CM579－ETHCAT

Application The principle of the EtherCAT diagnosis system and its use in the IEC application of an AC500 example V3 PLC is described in the application example AC500 EtherCAT－Diagnosis with AC500 V3．

Status codes

| Hexadecimal Value | Definition | Description |
| :--- | :--- | :--- |
| $0 x 00000000$ | TLR＿S＿OK | Status ok |
| $0 x C 0650005$ | TLR＿E＿ETHERCAT＿MASTER＿ERR <br> OR＿BUSSCAN＿FAILED | Existing bus does not match config－ <br> ured bus． |
| $0 x C 0650006$ | TLR＿E＿ETHERCAT＿MASTER＿NOT <br> －ALE＿SLAVES＿AVAIL | Not all slaves are available． |
| $0 x C 065000 \mathrm{~B}$ | TLR＿E＿ETHERCAT＿MASTER＿INV <br> ALID＿BUSCYCLETIME | The requested bus cycle time is <br> invalid． |
| $0 x C 065000 \mathrm{C}$ | TLR＿E＿ETHERCAT＿MASTER＿INV <br> ALID＿BROKEN＿SLAVE＿BEHAV－ <br> IOUR＿PARA | Invalid parameter for broken slave <br> behavior． |
| $0 x C 065000 \mathrm{~F}$ | TLR＿E＿ETHERCAT＿MASTER＿CO <br> E＿INVALID＿SLAVEID | Invalid Slaveld was used for CoE． |
| $0 x C 0650012$ | TLR＿E＿ETHERCAT＿MASTER＿CO <br> E＿INVALID＿INDEX | Invalid Index on slave requested． |


| Hexadecimal Value | Definition | Description |
| :---: | :---: | :---: |
| 0xC0650013 | TLR_E_ETHERCAT_MASTER_CO E INVĀLID COMMŪNICATION_STATE | Invalid bus communication state for CoE-Usage. |
| 0xC0650014 | TLR_E_ETHERCAT_MASTER_CO E_FRAME_LOST | Frame with CoE data is lost. |
| 0xC0650015 | TLR_E_ETHERCAT_MASTER_CO E_TIMEOUT | Timeout during CoE service. |
| 0xC0650016 | TLR_E_ETHERCAT_MASTER_CO E_SL̄AVE_NOT_ADD̄RESSABL̄E | Slave is not addressable (not on bus or power down?). |
| 0xC0650017 | TLR_E_ETHERCAT_MASTER_CO E_INVĀLID_LIST_TYPE | Invalid list type requested (during GetOdList). |
| 0xC0650018 | TLR_E_ETHERCAT_MASTER_CO E_SL̄AVE_RESPONSE_TOO_BIG | Data in slave response is too big for confirmation packet. |
| 0xC0650019 | TLR_E_ETHERCAT_MASTER_CO E_INVĀLID_ACCESS̄BITMASK | Invalid access mask selected (during GetEntryDesc). |
| 0xC065001A | TLR_E_ETHERCAT_MASTER_CO E_WKC_ERROR | Slave Working Counter Error during CoE service. |
| 0xC065001C | TLR_E_ETHERCAT_MASTER_INV ALID_COMMUNICATION_STATE | Command is not usable in the communication state. |
| 0xC065001E | TLR_E_ETHERCAT_MASTER_BUS _SCĀN_CURRENTL̄Y_RUNNIN̄G | The scan is already running. It cannot be started twice at the same time. |
| 0xC065001F | TLR_E_ETHERCAT_MASTER_BUS _SCAN_TIMEOUT | Timeout during bus scan. But at least a link is established. |
| 0xC0650020 | TLR_E_ETHERCAT_MASTER_BUS _SCĀN_NOT_READY_YET | The bus scan was not started before or is not finish yet. |
| 0xC0650021 | TLR_E_ETHERCAT_MASTER_BUS _SCĀN_INVALID_SL̄AVE | The requested slave is invalid. |
| 0xC0650022 | TLR_E_ETHERCAT_MASTER_CO E_INVALIDACCESS | Slave does not allow reading or writing (CoE-Access). |
| 0xC0650023 | TLR_E_ETHERCAT_MASTER_CO E_NO_MBX_SUPPORT | Slave does not support a mailbox. |
| 0xC0650024 | TLR_E_ETHERCAT_MASTER_CO E_NO_COE_SUPPŌRT | Slave does not support CoE. |
| 0xC0650025 | TLR_E_ETHERCAT_MASTER_TAS K_CREATION_FAILED | Task could not be created during run time. |
| 0xC0650026 | TLR_E_ETHERCAT_MASTER_INV ALID_SLAVE_SM_CONFIGURATION | The Sync Manager configuration of a slave is invalid. |
| 0xC0650027 | TLR_E_ETHERCAT_MASTER_SD O_ABORTCODE_TŌGGLE | SDO abort code: Toggle bit not alternated. |
| 0xC0650028 | TLR_E_ETHERCAT_MASTER_SD O_AB̄ORTCODE_TIMEOUT | SDO abort code: SDO protocol timed out. |
| 0xC0650029 | TLR_E_ETHERCAT_MASTER_SD O_ABORTCODE_CC̄S_SCS | SDO abort code: Client/server command specifier not valid or unknown. |
| 0xC065002A | TLR_E_ETHERCAT_MASTER_SD O_ABORTCODE_BLK_SIZE | SDO abort code: Invalid block size (block mode only). |
| 0xC065002B | TLR_E_ETHERCAT_MASTER_SD O_ABORTCODE_SĒQNO | SDO abort code: Invalid sequence number (block mode only). |


| Hexadecimal Value | Definition | Description |
| :--- | :--- | :--- |
| $0 \times C 065002 \mathrm{C}$ | TLR_E_ETHERCAT_MASTER_SD |  |
| O_ABORTCODE_CRC |  |  |$\quad$| SDO abort code: CRC error (block |
| :--- |
| mode only). |


| Hexadecimal Value | Definition | Description |
| :---: | :---: | :---: |
| 0xC0650042 | TLR E ETHERCAT MASTER SD <br>  | SDO abort code: Data cannot be transferred or stored to the application because of the present device state. |
| 0xC0650043 | TLR_E_ETHERCAT_MASTER_SD O_AB̄BRTCODE_DİCTIONARȲ | SDO abort code: Object dictionary dynamic generation fails or no object dictionary is present (e.g. object dictionary is generated from file and generation fails because of an file error). |
| 0xC0650044 | TLR_E_ETHERCAT_MASTER_SD O ABORTCODE UN̄KNOWN | SDO abort code: unknown code. |
| 0xC0CC0001 | ECM_ERROR_LLD_TIMEOUT | LLD: Timeout |
| 0xC0CC0003 | ECM ERROR_LLD_UNSUPPORTED_COMMAND | LLD: Unsupported command |
| 0xC0CC0004 | ECM_ERROR_LLD_DUPLICATE_FIXED_STATION_ADDRESS | LLD: Duplicate fixed station address |
| 0xC0CC0005 | ECM_ERROR_LLD_SII_CHECKSU M_ERROR | LLD: SII Checksum Error |
| 0xC0CC0006 | ECM_ERROR_LLD_SII_EEPROM_ LOAD̄ING ER $\bar{R} O R$ | LLD: SII EEPROM Loading Error |
| 0xC0CC0007 | ECM_ERROR_LLD_SII_MISSING_ ERRŌR ACK | LLD: SII Missing Error Ack |
| 0xC0CC0008 | ECM_ERROR_LLD_STATE_CHAN GE_FAILED | LLD: State Change Failed |
| 0xC0CC0009 | ECM_ERROR_LLD_UNEXPECTED_AL_S̄TATŪS | LLD: Unexpected AL Status |
| 0xC0CC000A | ECM_ERROR_LLD_UNEX- PECTED_WKC | LLD: Unexpected WKC |
| 0xC0CC000B | ECM_ERROR_LLD_MAILBOX_NO T_AVAILABLE | LLD: Mailbox not available |
| 0xC0CC000C | ECM_ERROR_LLD_MAILBOX_ME SSAḠE_TOO_LARḠE | LLD: Mailbox message too large |
| 0xCOCCO00D | ECM_ERROR_LLD_CONFIGURATION_IN_PROGRESS | LLD: Configuration in progress |
| 0xC0CCOOOE | ECM_ERROR_LLD_TOO_MANY_C YCLIC FRAMĒS | LLD: Too many cyclic frames |
| 0xC0CC000F | ECM_ERROR_LLD_CYCLIC_FRA ME EXCEEDS̄ MTŪ | LLD: Cyclic frame exceeds MTU |
| 0xC0CC0010 | ECM_ERROR_LLD_INVALID_CYCL IC_TĒLEGRAM_CONFIG | LLD: Invalid cyclic telegram config |
| 0xC0CC0011 | ECM_ERROR_LLD_BUILDING_CO PY_ROUTINES̄_FAİLED | LLD: Building copy routines failed |
| 0xC0CC0012 | ECM ERROR LLD UNSUP-PORTED_SLAVE_STATION_ADDRESS | LLD: Unsupported slave station address |
| 0xC0CC0013 | ECM ERROR LLD STATION ${ }^{-}$ADDRE $\bar{S} S$ NOT ALLOWED | LLD: Station Address not allowed |
| 0xC0CC0014 | ECM_ERROR_LLD_INVALID_STD_ TX_MBX_PHYS_OF̄FSET | LLD: Invalid Std TxMbx PhysOffset |


| Hexadecimal Value | Definition | Description |
| :---: | :---: | :---: |
| 0xC0CC0015 | ECM_ERROR_LLD_INVALID_STD_ RX_MBX_PHYS_O $\bar{F} F S E T$ | LLD: Invalid Std Rx Mbx PhysOffset |
| 0xC0CC0016 | ECM_ERROR_LLD_INVALID_BOO T_TX_MBX_PHYS_OFFSET | LLD: Invalid BOOT Rx Mbx PhysOffset |
| 0xC0CC0017 | ECM_ERROR_LLD_INVALID_BOO T_RX_MBX_PHYS_OFFSET | LLD: Invalid BOOT Tx Mbx PhysOffset |
| 0xC0CC0018 | ```ECM_ERROR_LLD_INVALID_STD_ TX_MBX_SM_NO``` | LLD: Invalid Std Tx Mbx SmNo |
| 0xC0CC0019 | ECM_ERROR_LLD_INVALID_STD_ RX_MBX_SM_NO | LLD: Invalid Std Rx Mbx SmNo |
| 0xC0CC001A | ECM_ERROR_LLD_INVALID_BOO <br> T_TX_MBX_SM_NO | LLD: Invalid BOOT Tx Mbx SmNo |
| 0xC0CC001B | $\begin{aligned} & \text { ECM_ERROR_LLD_INVALID_BOO } \\ & \text { T_RX_MBX_SM_NO } \end{aligned}$ | LLD: Invalid BOOT Rx Mbx SmNo |
| 0xC0CC001C | ECM ERROR_LLD UNCON-FIGURED_SLĀVE_S̄TATION_ADDERESS | LLD: Unconfigured slave station address |
| 0xC0CC001D | ECM_ERROR_LLD_WRONG_SLAV E_STATE | LLD: Wrong slave state |
| 0xC0CC001E | ECM_ERROR_LLD_CYCLE_TIME_ TOO_SMALL | LLD: Cycle time too small |
| 0xC0CC001F | ECM_ERROR_LLD_REPETITION_COUNT_NOT_SUPPORTED | LLD: Repetition count not supported |
| 0xC0CC0020 | ECM_ERROR_LLD_INVALID_CALL BACK TYPE | LLD: Invalid callback type |
| 0xC0CC0021 | ECM_ERROR_LLD_INVALID_CYCL E MŪLTIPLIER | LLD: Invalid cycle multiplier |
| 0xC0CC0022 | ECM_ERROR_LLD_UNKNOWN_E RROR | LLD: Unknown Error |
| 0xC0CC0023 | ECM_ERROR_LLD_INVALID_REG LENGTH | LLD: Invalid reg length |
| 0xC0CC0024 | ECM_ERROR_LLD_INVALID_PARA METĒR | LLD: Invalid parameter |
| 0xC0CC0025 | ECM_ERROR_LLD_IRQ_NOT_AVA ILABLE | LLD: IRQ not available |
| 0xC0CC0026 | ECM_ERROR_LLD_IOMEM_IRQ_N OT_ĀVAILABLE | LLD: IOMem Irq not available |
| 0xC0CC0027 | ECM_ERROR_LLD_HW_INIT_FAIL ED | LLD: Hardware init failed |
| 0xC0CC0028 | ECM_ERROR_LLD_MUTEX_CREATION_FAILED | LLD: Mutex creation failed |
| 0xC0CC0029 | $\begin{aligned} & \text { ECM_ERROR_LLD_DC_RX_LATC } \\ & \text { H_COM- } \\ & \text { MAND_REQUIRED_FOR_DC } \end{aligned}$ | LLD: DC Rx Latch command is not configured within cyclic frames |
| 0xC0CC002A | ECM_ERROR LLD_TX_PROCESS _IMAGE_EXCĒEDĒD | LLD: Transmit process image is exceeded |
| 0xC0CC002B | ECM_ERROR_LLD_RX_PROCESS _IMĀGE_EXCĒEDĒD | LLD: Receive process image is exceeded |
| 0xC0CC002C | $\begin{aligned} & \text { ECM_ERROR_LLD_MBX_STATE_I } \\ & \text { MAGE_EXCEEDED } \end{aligned}$ | LLD: Mailbox State image is exceeded |


| Hexadecimal Value | Definition | Description |
| :---: | :---: | :---: |
| 0xC0CC002D | ECM_ERROR_LLD_RESULT_DUP LICATE_BWR_RX_LATCH_CMD | LLD: Duplicate BWR Rx DC Latch command detected in cyclic frames |
| 0xC0CC002E | ECM_ERROR_LLD_RESULT_DUP LICAT̄E_EXT_SYYST̄IME_CONTROL_C̄MD | LLD: Duplicate External Sync SysTime Control command detected in cyclic frames |
| 0xC0CC002F | ECM_ERROR_LLD_CC_PROCESS _IMAGE_EXCEEDED | LLD: Cross Communication Process image exceeded |
| 0x40CD0017 | ECM_INFO_EMC_BUS_IS_OFF | Bus is off |
| 0xC0CD0001 | ECM_ERROR_EMC_REQUEST_D ESTINATION PROBLEM | Request Destination Problem |
| 0xC0CD0002 | ECM_ERROR_EMC_INVALID_SLA VE_STATION_ADDRESS | Invalid slave station address |
| 0xC0CD0003 | ECM_ERROR_EMC_CONFIGURATION_BUFFER_IS_OPEN | Configuration buffer is open |
| 0xC0CD0004 | ECM_ERROR_EMC_WRONG_STA TE_FOR_RECONFIḠURATION | Wrong state for reconfiguration |
| 0xC0CD0005 | ECM_ERROR_EMC_CONFIGURATION_BUFFER_IS_NOT_OPEN | Configuration buffer is not open |
| 0xC0CD0006 | ECM_ERROR_EMC_SLAVE_STATION_ADDRES̄S_ALREADY_IN_C ONFIG | Slave station address already in config |
| 0xC0CD0007 | ECM_ERROR_EMC_INVALID_STD _MBX_PARAMETERS | Invalid Std Mbx parameters |
| 0xC0CD0008 | ECM_ERROR_EMC_INVALID_BOO T_MB̄X_PARAMETER̄S | Invalid BOOT Mbx parameters |
| 0xC0CD0009 | ECM_ERROR_EMC_STD_MBX_S M_ARE_OVERLAPPING | Std Mbx SMs are overlapping |
| 0xC0CD000A | ECM_ERROR_EMC_BOOT_MBX SM_ĀRE_OVERLAPPING | BOOT Mbx SMs are overlapping |
| 0xC0CD000B | ECM_ERROR_EMC_SM_PARAMS _ALREADY_ADDED | SM Params already added |
| 0xC0CD000C | ECM_ERROR_EMC_INVALID_SM_ NUMBER | Nvalid SM number |
| 0xC0CD000D | ECM_ERROR_EMC_FMMU_PARA MS_ALREADY_ADDED | FMMU params already added |
| 0xC0CD000E | ECM_ERROR_EMC_INVALID_FMM U_NUMBER | Invalid FMMU number |
| 0xC0CD000F | ECM_ERROR_EMC_INVALID_MIN STATE | Invalid min state |
| 0xC0CD0010 | ECM_ERROR_EMC_CYCLE_FRA ME_ĀMOUNT_EXCEEDED | Cycle frame amount exceeded |
| 0xC0CD0011 | ECM_ERROR_EMC_INVALID_CYC LIC_FRAME_IN_CONFIGURAT̄TION | Invalid cycle frame in configuration |
| 0xC0CD0012 | ECM_ERROR_EMC_CYCLE_FRA ME_INDEX_NŌT_VĀLID | Cycle frame index not valid |
| 0xC0CD0013 | ECM_ERROR_EMC_INVALID_TEL EGRĀM_LENḠTH | Invalid telegram length |
| 0xC0CD0014 | ECM_ERROR_EMC_CYCLE_FRA ME_LENGTH_EXCEEDED | Cycle frame length exceeded |


| Hexadecimal Value | Definition | Description |
| :---: | :---: | :---: |
| 0xC0CD0015 | ```ECM_ERROR_EMC_AMOUNT_OF TELE- ḠRAMS_IN_CYCLIC_FRAME_EXC EEDED``` | Amount of telegrams in cyclic frame exceeded |
| 0xC0CD0016 | ECM_ERROR_EMC_STATE_CHAN GE IN PROGRESS | State change in progress |
| 0xC0CD0018 | ECM_ERROR_EMC_TOO_MANY_ SLAVES GIVEN | Too many slaves given |
| 0xC0CD0019 | ECM_ERROR_EMC_DUPLICATE STA- <br> TION_ADDRESS_IN_LIST | Duplicate station address in list |
| 0xC0CD001A | ECM_ERROR_EMC_COMMAND_TYPE_NOT_ALLOWED_FO R_SLAV̄E_FSM | Command type not allowed for slave FSM |
| 0xC0CD001B | ECM ERROR EMC CONFIGURATION_DATA_INCORRECT | Configuration data incorrect |
| 0xC0CD001C | ECM_ERROR_EMC_VENDORID MISMĀTCH | VendorID mismatch |
| 0xC0CD001D | ECM_ERROR_EMC_PRODUCTCODĒ_MISMĀTCH | ProductCode mismatch |
| 0xC0CD001E | ECM_ERROR_EMC_REVISIONNO_MISMATC̄ | Revision number mismatch |
| 0xC0CD001F | ECM_ERROR_EMC_SERIALNO_MISMATTCH | Serial number mismatch |
| 0xC0CD0020 | ECM_ERROR_EMC_LOST_CONNECTION | Lost connection |
| 0xC0CD0021 | ECM_ERROR_EMC_UNKNOWN_S TATE CHANGE HAPPPENED | Unknown state change happened |
| 0xC0CD0022 | ECM_ERROR_EMC_UNEX-PECTED_STATE_CH̄ANGE_HAPPENED | Unexpected state change happened |
| 0xC0CD0023 | ```ECM_ERROR_EMC_SLAVE_CHAN GED_STATE``` | Slave changed state |
| 0xC0CD0026 | ECM_ERROR_EMC_DC_RX_TIMESTAMP_ERROR | DC Rx Timestamp error |
| 0xC0CD0027 | ECM ERROR EMC DC MASTER _PORT_TIMESTAMP_ERROR | DC master port timestamp error |
| 0xC0CD0028 | ECM_ERROR_EMC_INVALID_SLA VE_INDEX | Invalid slave index |
| 0xC0CD0029 | ECM_ERROR_EMC_WRONG_MAS TER_STATE |  |
| 0xC0CD002A | ECM_ERROR_EMC_INVALID_TRA NSFER_ID | Invalid Transfer Id |
| 0xC0CD002B | ECM_ERROR_EMC_INVALID_SEG MENTATION | Invalid Segmentation |
| 0xC0CD002C | ECM_ERROR_EMC_IP_PARAMS_ ALREADY ADDED | EoE IP Params already added |
| 0xC0CD002D | ECM_ERROR_EMC_EOE_SUPPORT_NOT_AVAILABLE | EoE support not available |


| Hexadecimal Value | Definition | Description |
| :--- | :--- | :--- |
| $0 x C 0 C D 002 \mathrm{E}$ | ECM_ERROR_EMC_END_CON- <br> FIGURATION_IN_PROGRESS | End configuration in progress |
| $0 x C 0 C D 002 \mathrm{~F}$ | ECM_ERROR_EMC_WRONG_STA | Wrong state for reconfiguration (Bus <br> TE_FOR_RECONFIGURA- <br> TION_BUS_IS_ON |
| Is On) |  |  |


| Hexadecimal Value | Definition | Description |
| :---: | :---: | :---: |
| 0xC0CD0046 | ECM ERROR EMC UNEXPECT̄̄ED SLAV̄E AT PORTO | Unexpected slave at port 0 of slave. |
| 0xC0CD0047 | ECM_ERROR_EMC_UNEXPECTED_SLAVE_AT_PORT1 | Unexpected slave at port 1 of slave. |
| 0xC0CD0048 | ECM_ERROR_EMC_UNEXPECTED_SLAV̄E_AT_PORT2 | Unexpected slave at port 2 of slave. |
| 0xC0CD0049 | ECM_ERROR_EMC_UNEXPECTED_SLAV̄E_AT_PORT3 | Unexpected slave at port 3 of slave. |
| 0xC0CD004A | ECM_ERROR_EMC_UNEXPECTED_SLAV̄E_RĒCONNECTED | - |
| 0xC0CD004B | ECM_ERROR_EMC_UNEX- <br> PECTED_MISS̄ING_SLAVE_AT_PO RTO | Missing slave at port 0 of slave. |
| 0xC0CD004C | ECM_ERROR_EMC_UNEX- <br> PECT̄ED_MISS̄ING_SLAVE_AT_PO RT1 | Missing slave at port 1 of slave. |
| 0xC0CD004D | ```ECM_ERROR_EMC_UNEX- PECT̄ED_MISS̄ING_SLAVE_AT_PO RT2``` | Missing slave at port 2 of slave. |
| 0xC0CD004E | ECM_ERROR_EMC_UNEX- <br> PECT̄ED_MISS̄ING_SLAVE_AT_PO RT3 | Missing slave at port 3 of slave. |
| 0xC0CD004F | ECM_ERROR_EMC_SLAVE_NOT_ CHEC̄KED | Slave is not checked. |
| 0xC0CD0050 | ECM_ERROR_EMC_UNEXPECTED_SLAVE_AT_PORT0_1 | Unexpected slave at port 0 and 1 of slave. |
| 0xC0CD0051 | ECM_ERROR_EMC_UNEXPECTED_SLAVE_AT_PORTO_2 | Unexpected slave at port 0 and 2 of slave. |
| 0xC0CD0052 | ECM_ERROR_EMC_UNEXPECTED_SLAV̄E_AT_PORTO_3 | Unexpected slave at port 0 and 3 of slave. |
| 0xC0CD0053 | ECM_ERROR_EMC_UNEXPECT̄̄E_SLAV̄E_AT_PORT1_2 | Unexpected slave at port 1 and 2 of slave. |
| 0xC0CD0054 | ECM_ERROR_EMC_UNEXPECTED_SLAV̄E_AT_PORT1_3 | Unexpected slave at port 1 and 3 of slave. |
| 0xC0CD0055 | ECM_ERROR_EMC_UNEXPECTED_SLAVE_AT_PORT2_3 | Unexpected slave at port 2 and 3 of slave. |
| 0xC0CD0056 | ECM_ERROR_EMC_UNEXPECT̄ED_SLAV̄E_AT_PORTO_1_2 | Unexpected slave at port 0, 1 and 2 of slave. |
| 0xC0CD0057 | ECM ERROR EMC UNEX- <br> PECTED_SLAV̄E_AT_PORT0_1_3 | Unexpected slave at port 0, 1 and 3 of slave. |
| 0xC0CD0058 | ECM ERROR_EMC_UNEXPECT̄ED_SLAV̄E_AT̄_PORT0_2_3 | Unexpected slave at port 0, 2 and 3 of slave. |
| 0xC0CD0059 | ECM_ERROR_EMC_UNEXPECTED_SLAVE_AT_PORT1_2_3 | Unexpected slave at port 1, 2 and 3 of slave. |
| 0xC0CD005A | ECM_ERROR_EMC_MISSING_SL AVE_AT_PORTO_1 | Missing slave at port 0 and 1 of slave. |
| 0xC0CD005B | ECM_ERROR_EMC_MISSING_SL AVE_AT_PORTO_2 | Missing slave at port 0 and 2 of slave. |
| 0xC0CD005C | ECM_ERROR_EMC_MISSING_SL AVE_AT_PORTO_3 | Missing slave at port 0 and 3 of slave. |


| Hexadecimal Value | Definition | Description |
| :--- | :--- | :--- |
| $0 \times C 0 C D 005 \mathrm{D}$ | ECM_ERROR_EMC_MISSING_SL | Missing slave at port 1 and 2 of <br> slave. |
| AVE_AT_PORT1_2 | ECM_ERROR_EMC_MISSING_SL | Missing slave at port 1 and 3 of <br> slave. |
| AVE_AT_PORT1_3 |  |  |


| Hexadecimal Value | Definition | Description |
| :--- | :--- | :--- |
| $0 x C 0 C D 0074$ | ECM_ERROR_EMC_DC_RX_LATC |  |
| H_ERROR_AT_PORT3 |  |  | | DC Latch Error detected at port 3 of |
| :--- |
| slave |


| Hexadecimal Value | Definition | Description |
| :---: | :---: | :---: |
| 0xC0CD0089 | ECM ERROR EMC UNSUP- <br> PORTED_FMM̄U_MĀP- <br> PING_OF_EXT_S̄YNC_OBJ_RECO <br> RD | Unsupported FMMU mapping of Ext Sync object detected |
| 0xC0CD008A | ECM_ERROR_EMC_EXT_SYNC_R EQUİRES_ADJUUST_EXT_SYNC_C MD | Unicast Ext Sync control (APWR/ FPWR $0 \times 910$ ) is required |
| 0xC0CD008B | ECM_ERROR_EMC_EXT_SYNC_C MD_DOES_NŌT_MĀTCH_XRMW_ CMD | Unicast Ext Sync control does not match xRMW command |
| 0xC0CD008C | ECM_ERROR_EMC_EXT_SYNC_R EQUIRES_XRMW_CMD | Ext Sync requires DC configuration (xRMW command to 0x910) |
| 0xC0CD008D | ECM ERROR_EMC_EXPLICIT_DE V_IDĒNT_FAIL̄ED_ĀLSTATUS | Explicit Device identification via ALSTATUS failed |
| 0xC0CD008E | ECM_ERROR_EMC_EXPLICIT_DE V_IDENT_FAILED_REG | Explicit Device identification via register failed |
| 0xC0CD008F | ECM ERROR EMC COPY INFOS FOŪND_AT_ŪNMAPPPED_RECEIV E_DATA | CopyInfos found at unmapped receive data |
| 0xC0CD0090 | ECM_ERROR_EMC_COPY_INFO RECĒIVE_DATA_AREA_NOTT_MAT̄ CHING | CopyInfo receive data area is not matching |
| 0xC0CD0091 | ECM_ERROR_EMC_SDO_UPLOA D_TŌO_LONG | SDO Upload data too long |
| 0xC0CD0092 | ECM_ERROR_EMC_SDO_UPLOA D_TŌO_SHOR̄T | SDO Upload data too short |
| 0xC0CD0093 | ECM_ERROR_EMC_SDO_UPLOA D_COM- <br> PĀRE_DOES_NOT_MATCH_EXPE CTATION | SDO Upload compare does not match expectation |
| 0xC0CD0094 | ```ECM_ERROR_EMC_SOE_READ_T OO_LONG``` | SoE Read IDN data too long |
| 0xC0CD0095 | ECM_ERROR_EMC_SOE_READ_T OO_S̄HORT | SoE Read IDN data too short |
| 0xC0CD0096 | ECM_ERROR_EMC_SOE_READ_ COM- <br> PARE_DOES_NOT_MATCH_EXPE CTATION | SoE Read compare does not match expectation |
| 0xC0CD0097 | ECM_ERROR_EMC_REG_INITCM D_COM- <br> PARE_DOES_NOT_MATCH_EXPE CTATION | Register read compare does not match expectation |
| 0xC0CD0098 | ECM ERROR EMC REDUNDANCY PORT_ONLY_POSSIBLE_ONCE | Redundancy port can only be placed once into configuration |
| 0xC0CD0099 | ECM_ERROR_EMC_STARTUP_SC AN_SII_FAILED | Startup scan of SII failed |
| 0xC0CD009A | ECM_ERROR_EMC_STARTUP_VE RIFY_SII_FAILED | Startup verification of SII failed |
| 0xC0CD009B | ECM_ERROR_EMC_MAIN_PORT_ NOT_CONNEC̄TED | Main port not connected during topology scan |


| Hexadecimal Value | Definition | Description |
| :--- | :--- | :--- |
| 0xC0CD009C | ECM_ERROR_EMC_BUS_SCAN_T | Bus scan detects too many slaves |
| OO_MANY_SLAVES |  |  |


| Hexadecimal Value | Definition | Description |
| :---: | :---: | :---: |
| 0xC0CF0009 | ECM_ERROR_COE_TRANSFER_A BORTED | CoE: Transfer has been aborted. |
| 0xC0CF000A | ECM_ERROR_COE_SDOINFO_INITIALIZATION ERROR | 0xC0CF000B |
| 0xC0CF000C | ECM_ERROR_COE_PROTOCŌL_ERRŌR | CoE Protocol Error |
| 0xC0CF000D | ECM_ERROR_COE_NO_AOE_AVA ILABLE | CoE: No AoE available |
| 0xC0CF000F | ECM_ERROR_COE_INVALID_SLA VE_STATION_ADDRESS | CoE: Invalid slave station address |
| 0xC0CF8000 | ECM_ERROR_COE_ABORT- <br> CODĒ_TOGGĒE_BIT_NOT_ALTER NATED | SDO Abort Code: Toggle Bit not alternated |
| 0xC0CF8001 | ECM ERROR_COE_ABORT-CODĒ_COMMAND_SPECIFIER_ÑOT_VALID | SDO Abort Code: Command specifier not valid |
| 0xC0CF8002 | ECM ERROR COE_ABORTCODĒ_PROTŌCOL_TIMEOUT | SDO Abort Code: Protocol Timeout |
| 0xC0CF8003 | ECM_ERROR_COE_ABORTCODĒ_OUT_ŌF_MEMORY | SDO Abort Code: Out Of Memory |
| 0xC0CF8004 | ECM ERROR COE ABORTCODE_UNSUPPORTED_ACCESS | SDO Abort Code: Unsupported access |
| 0xC0CF8005 | ECM ERROR_COE ABORTCODĒ_OBJECTT_IS_WRITE_ONLY | SDO Abort Code: Object is write only |
| 0xC0CF8006 | ECM_ERROR_COE_ABORTCODĒ_OBJECTT_IS_READ_ONLY | SDO Abort Code: Object is read only |
| 0xC0CF8007 | ```ECM_ERROR_COE_ABORT- COD\overline{E}SUB- INDEX_CANNOT_BE_WRITTEN_SI 0_NZ``` | SDO Abort Code: Subindex cannot be written if subindex 0 is not zero |
| 0xC0CF8008 | ```ECM_ERROR_COE_ABORT- CODE\overline{E}COM- PLETE_ACCESS_NOT_SUP- PORTED``` | SDO Abort Code: Complete access not supported |
| 0xC0CF8009 | ECM_ERROR_COE_ABORTCOD $\bar{E}$ _OBJECTT_LENGTH_EXCEE DS_MĀILBOX_SIZE | SDO Abort Code: Object length exceeds mailbox size |
| 0xC0CF800A | ECM ERROR_COE ABORT- <br> CODĒ_OBJEC̄T_MĀPPED_TO_RX PDO_N̄O_WRITE | SDO Abort Code: Object mapped to RxPDO, SDO Download blocked |
| 0xC0CF800B | ECM ERROR_COE_ABORTCODĒ_OBJEC̄T_DOES_NOT_EXIS T | SDO Abort Code: Object does not exist |
| 0xC0CF800C | ECM_ERROR_COE_ABORTCODĒ_OBJECTT_CANNNOT_BE_PD O_MAPPPED | SDO Abort Code: Object cannot be mapped to PDO |
| 0xC0CF800D | ECM_ERROR_COE_ABORT- <br> CODE_PDO_LENGTH_WOULD_E XCEED | SDO Abort Code: PDO Length would exceed maximum size |
| 0xC0CF800E | ECM_ERROR_COE_ABORT-CODĒ_GEN_PARAM_INCOMPATIBILITY | SDO Abort Code: General parameter incompatibility |


| Hexadecimal Value | Definition | Description |
| :--- | :--- | :--- |
| 0xC0CF800F | ECM_ERROR_COE_ABORT- <br> CODE_ACCESS_FAILED_DUE_TO <br> -HW_ERROR | SDO Abort Code: Access failed due <br> to hardware error |
| 0xC0CF8010 | ECM_ERROR_COE_ABORT- <br> CODE_DATA- <br> TYPE_DOES_NOT_MATCH | SDO Abort Code: Data type does <br> not match |
| 0xC0CF8011 | ECM_ERROR_COE_ABORT- | SDO Abort Code: Data type length <br> too long |
| CODE_DATA- |  |  |
|  | TYPE_LENGTH_TOO_LONG | SDO Abort Code: Data type length <br> 0xC0CF8012 <br> 0xC0CF8013 |
| ECM_ERROR_COE_ABORT- | CODE_DATA- |  |
|  | TYPE_LENGTH_TOO_SHORT | too short |


| Hexadecimal Value | Definition | Description |
| :---: | :---: | :---: |
| 0xCOD00004 | ECM_ERROR_EOE_INVALID_EOE RESPONSE | Invalid EoE Response |
| 0xCOD00005 | ECM_ERROR_EOE_UNKNOWN_E RRŌ | SetIPParam/SetFilterParam: Unknown error |
| 0xC0D00006 | ECM_ERROR_EOE_UNSPECI- FIED_ERROR | SetIPParam/SetFilterParam: Unspecified Error |
| 0xC0D00007 | ECM ERROR EOE UNSUPPORT̄ED_FRAME_TYPE | SetIPParam/SetFilterParam: Unsupported frame type |
| 0xC0D00008 | ECM_ERROR_EOE_NO_IP_SUP- PORT | SetIPParam/SetFilterParam: No IP support |
| 0xC0D00009 | ECM_ERROR_EOE_DHCP_NOT_S UPPŌRTED | SetIPParam/SetFilterParam: DHCP not supported |
| OxCOD0000A | ECM_ERROR_EOE_NO_FILTER_S UPPŌRT | SetIPParam/SetFilterParam: No filter supported |
| 0xCOD0000B | ECM_ERROR_EOE_TIMEOUT | EoE Timeout |
| 0xCOD0000C | ECM_ERROR_EOE_SHUT- TING_DOWN | EoE is shutting down |
| OxCOD0000D | ECM_ERROR_EOE_MASTER_AD DRES̄S_NOT_ALLOWED | EoE: Master address is not allowed to use here |
| 0xCOD0000E | ECM_ERROR_EOE_CONFIGURATION_IS_NOT_OPEN | EoE: Configuration is not open |
| 0xCOD0000F | ECM ERROR EOE CONFIGURATION_IS_ALREADEM_OPEN | EoE: Configuration is already open |
| 0xC0D00010 | ECM ERROR EOE DUPLICATE_IP_ADDRESS | EoE: Duplicate IP address |
| 0xC0D00011 | ECM ERROR EOE DUPLI-CATE_MAC_ADDRĒSS_ON_MULTIPLE_PORTS | EoE: Duplicate MAC address on multiple ports |
| 0xCOD00012 | ECM_ERROR_EOE_FRAME_TOO_ LARGE | EoE: Frame too large |
| 0xCOD00013 | ECM_ERROR_EOE_IF_INITIALIZATIŌN_ERRŌR | EoE: Interface initialization error |
| 0xC0D00014 | ECM_ERROR_EOE_IF_NO_FRAM E_AVAILABLE | EoE: No Frame available |
| 0xCOD00015 | ECM_ERROR_EOE_LINK_DOWN | EoE: Link down |
| 0xC0D10002 | ECM_ERROR_FOE_ERROR_UNK NOWN_ERROR | - |
| 0xC0D10003 | ECM_ERROR_FOE_INVALID_TRA NSFĒR_HANDLE | FoE: Invalid transfer handle |
| 0xC0D10004 | ECM_ERROR_FOE_INVALID_TRA NSFEER_STATE | FoE: Invalid transfer state |
| 0xC0D10005 | ECM_ERROR_FOE_INVALID_SLA VE_STATION_ADDRESS | FoE: Invalid slave station address |
| 0xC0D10006 | ECM_ERROR_FOE_WRONG_SLA VE_STATE | FoE: Wrong slave state |
| 0xC0D10007 | ECM_ERROR_FOE_NO_MAILBOX _AVAİLABLE | FoE: No mailbox available |
| 0xC0D10008 | ECM_ERROR_FOE_TRANSFER_A BORTED | FoE: Transfer has been aborted |


| Hexadecimal Value | Definition | Description |
| :---: | :---: | :---: |
| 0xC0D10009 | ECM_ERROR_FOE_PROTOCŌL_TIMEŌUT | FoE: Protocol Timeout |
| 0xC0D1000A | ECM_ERROR_FOE_TRANSFER_S EGMĒNT_TOŌ_LON̄G | FoE: Transfer segment is too long |
| 0xC0D1000B | ECM_ERROR_FOE_MAILBOX_TR ANSMIT_FAILED | FoE: Mailbox transmit failed |
| 0xC0D1000C | ECM ERROR_FOE_FILENAME TOO LONG | FoE: Filename is too long |
| 0xC0D1000D | ECM_ERROR_FOE_BUFFER_EXC EEDED | FoE: Buffer is exceeded |
| 0xC0D1000E | ECM ERROR FOE FIRST SEGMENT_SHOUL̄D_NŌT_BE_EMPTY | FoE: First segment should not be empty |
| 0xC0D1000F | ECM ERROR FOE SEGMENT SHOUL̄D BE EMPTY | FoE: Segment should be empty |
| 0xC0D18000 | ECM_ERROR_FOE_ERROR_NOT DEFINED | FoE: Error Response: not defined |
| 0xC0D18001 | ECM_ERROR_FOE_ERROR_NOT _FOŪND | FoE: Error Response: Not Found |
| 0xC0D18002 | ECM_ERROR_FOE_ACCESS_DEN | FoE: Error Response: Access Denied |
| 0xC0D18003 | ECM_ERROR_FOE_ERROR_DISK _FUL̄̄ | FoE: Error Response: Disk full |
| 0xC0D18004 | ECM_ERROR_FOE_ERROR_ILLE GAL | FoE: Error Response: Illegal |
| 0xC0D18005 | ECM_ERROR_FOE_ERROR_PACK ET_NUMBER_WRONG | FoE: Error Response: Packet number is wrong |
| 0xC0D18006 | ECM_ERROR_FOE_ERROR_ALRE ADY_EXISTS | FoE: Error Response: Already exists |
| 0xC0D18007 | ECM_ERROR_FOE_ERROR_NO_ USER | FoE: Error Response: No User |
| 0xC0D18008 | ECM_ERROR_FOE_ERROR_BOO TSTRAP_ONLY | FoE: Acces to specified file is only allowed in BOOT state |
| 0xC0D18009 | ECM_ERROR_FOE_ERROR_NOT _BOŌTSTRAP | FoE: Access to specified file is only allowed when in PREOP, SAFEOP or OP |
| 0xC0D1800A | ECM_ERROR_FOE_ERROR_NO_ RIGHTTS | FoE: No Rights |
| 0xC0D1800B | ECM_ERROR_FOE_ERROR_PROGRAM ERRO $\bar{R}$ | FoE: Program Error |
| 0xC0D20001 | ECM_ERROR_SOE_UNKNOWN_S OE_ERROR | SoE: Unknown SoE Error |
| 0xC0D20002 | ECM_ERROR_SOE_INITIALIZATION ERROR | SoE: Initialization error |
| 0xC0D20003 | ECM_ERROR_SOE_INVALID_TRA NSFĒR_HANDLE | SoE: Invalid transfer handle |
| 0xC0D20004 | ECM_ERROR_SOE_NO_MAILBOX AVAILABLE | SoE: No Mailbox available |
| 0xC0D20005 | ECM_ERROR_SOE_INVALID_TRA NSFĒR_STATE | SoE: Invalid transfer state |


| Hexadecimal Value | Definition | Description |
| :---: | :---: | :---: |
| 0xC0D20006 | ECM_ERROR_SOE_TRANSFER_S EGMĒNT_TOŌ_LONG | SoE: Transfer segment is too long |
| 0xC0D20007 | ```ECM_ERROR_SOE_SHUT- TING DOWN``` | SoE is shutting down |
| 0xC0D20008 | ECM_ERROR_SOE_MAX_TOTAL BYTĒS_SMALLER_THAN_ACTUĀL _TOTAL_BYTES | SoE: Maximum total bytes is smaller than actual total bytes |
| 0xC0D20009 | ECM_ERROR_SOE_MAILBOX_TR ANSMIT_FAILED | SoE: Mailbox transmit failed |
| 0xC0D2000A | ECM_ERROR_SOE_INVALID_SOE _HEADER | SoE: Invalid SoE header |
| 0xC0D2000B | ECM_ERROR_SOE_PROTOCŌL_TIMEŌUT | SoE: Protocol Timeout |
| 0xC0D2000C | ECM_ERROR_SOE_PROTOCŌL ERRŌR | SoE: Protocol Error |
| 0xC0D2000D | ECM_ERROR_SOE_TRANSFER_A BORTED | SoE: Transfer has been aborted |
| 0xC0D2000E | ECM_ERROR_SOE_WRONG_SLA VE_STATE | SoE: Wrong slave state |
| 0xC0D2000F | $\begin{aligned} & \text { ECM_ERROR_SOE_NO_AOE_AVA } \\ & \text { ILABLE } \end{aligned}$ | SoE: No AoE available |
| 0xC0D20010 | ECM_ERROR_SOE_INVALID_SLA VE STATION -ADDRESS | SoE: Invalid slave station address |
| 0xC0D21001 | ECM_ERROR_SOE_SSC_NO_IDN | SoE: No IDN |
| 0xC0D21009 | ECM_ERROR_SOE_SSC_INVALID _ACCESS_TO_ELEMENT_1 | SoE: Invalid access to element 1 |
| 0xC0D22001 | $\begin{aligned} & \text { ECM_ERROR_SOE_SCC_NO_NA } \\ & \mathrm{ME} \end{aligned}$ | SoE: IDN has no name |
| 0xC0D22002 | ECM_ERROR_SOE_SSC_NAME_T RANS̄MISSION_IS_TOO_S | SoE: Name transmission is too short |
| 0xC0D22003 | ECM_ERROR_SOE_SSC_NAME_T RANS̄MISSION_IS_TOO_LONG | SoE: Name transmission is too long |
| 0xC0D22004 | ECM_ERROR_SOE_SSC_NAME_ CANN̄OT BE CHANGED | SoE: Name cannot be changed |
| 0xC0D22005 | ECM_ERROR_SOE_SSC_NAME_I S_WRITE_PRO- <br> TĒCTED_ĀT_THIS_TIME | SoE: Name is write protected at this time |
| 0xC0D23002 | ECM_ERROR_SOE_SSC_ATTRIB UTE_TRANSMISSION_IS_TOO_SHORT | SoE: Attribute transmission is too short |
| 0xC0D23003 | ECM_ERROR_SOE_SSC_ATTRIB UTE-TRANSMIS- <br> SION_IS_TOO_LONG | SoE: Attribute transmission is too long |
| 0xC0D23004 | ECM_ERROR_SOE_SSC_ATTRIB UTE CANNOT BE CHANGED | SoE: Attribute cannot be changed |
| 0xC0D23005 | ECM_ERROR_SOE_SSC_ATTRIB UTE_IS_WRITE_PROTECTED_AT_THIS_TIME | SoE: Attribute is write protected at this time |
| 0xC0D24001 | $\begin{aligned} & \text { ECM_ERROR_SOE_SSC_NO_UNI } \\ & \mathrm{T} \end{aligned}$ | SoE: IDN has no unit |


| Hexadecimal Value | Definition | Description |
| :---: | :---: | :---: |
| 0xC0D24002 | ECM_ERROR_SOE_SSC_UNIT_T RANS̄MISSION_IS_TOO_S̄HORT | SoE: Unit transmission is too short |
| 0xC0D24003 | ECM_ERROR_SOE_SSC_UNIT_T RANS̄MISSION IS TOO $\overline{\text { LONG }}$ | SoE: Unit transmission is too long |
| 0xC0D24004 | ECM_ERROR_SOE_SSC_UNIT_C ANNŌT BE CHANGED | SoE: Unit cannot be changed |
| 0xC0D24005 | ECM_ERROR_SOE_SSC_UNIT_IS WRITTE_PRO- <br> T̄ECTED_AT_THIS_TIME | SoE: Unit is write protected at this time |
| 0xC0D25001 | ECM_ERROR_SOE_SSC_NO_MA XIMŪM_VALUĒ | SoE: IDN has no maximum value |
| 0xC0D25002 | ECM_ERROR_SOE_SSC_MINIMUM_VALUE_TRA $\bar{N} S M I \bar{S}-$ SION_IS_TOO_SHORT | SoE: Minimum value transmission is too short |
| 0xC0D25003 | ECM_ERROR_SOE_SSC_MIN-IMUM_VALUE_TRANSSMISSION_IS_TOO_LONG | SoE: Minimum value transmission is too long |
| 0xC0D25004 | ECM_ERROR_SOE_SSC_MINIMUM_VALUE_CANN̄OT_BE_CHA NGED | SoE: Minimum value cannot be changed |
| 0xC0D25005 | ECM_ERROR_SOE_SSC_MINIMUM VALUE IS WRITE_PROTECTED_AT_THIS_TIME | SoE: Mimum value is write protected at this time |
| 0xC0D26001 | ECM_ERROR_SOE_SSC_NO_MA XIMUM_VALUE | SoE: IDN has no maximum value |
| 0xC0D26002 | ECM_ERROR_SOE_SSC_MAX-IMUM_VALUE_TRANSSMISSION_IS_TOO_SHORT | SoE: Maximum value transmission is too short |
| 0xC0D26003 | ECM_ERROR_SOE_SSC_MAXIMUM VALUE_TRAN̄SMISSION_IS_TOO_LONG | SoE: Maximum value transmission is too long |
| 0xC0D26004 | ECM_ERROR_SOE_SSC_MAXIMUM_VALUE_CANNOT_BE_CHA NGED | SoE: Maximum value cannot be changed |
| 0xC0D26005 | ECM ERROR SOE SSC MAXIMUM VALUE IS WRITE_PROTECTED_AT_THIS_TIME | SoE: Maximum value is write protected at this time |
| 0xC0D27002 | ECM_ERROR_SOE_SSC_OPDATA TRANSMISS̄ION_IS_TOO_SHORT | SoE: OpData transmission is too short |
| 0xC0D27003 | ECM ERROR SOE SSC_OPDATA _TRĀNSMISSİON_IS_TOŌ_LONG | SoE: OpData transmission is too long |
| 0xC0D27004 | ECM ERROR SOE SSC OPDATA _CAN̄NOT_BE__CHĀNGED̄ | SoE: OpData cannot be changed |
| 0xC0D27005 | ECM_ERROR_SOE_SSC_OPDATA IS WRITE PROTECTED_AT_THIS_TIME | SoE: OpData is write protected at this time |
| 0xC0D27006 | ECM_ERROR_SOE_SSC_OPDATA IS_LOWER_THAN_MINİMUM_VALUE | SoE: OpData is lower than minimum value |


| Hexadecimal Value | Definition | Description |
| :---: | :---: | :---: |
| 0xC0D27007 | ECM ERROR_SOE_SSC_OPDATA IS_HIGHER_THAN_MAXİMUM_VALUE | SoE: OpData is higher than maximum value |
| 0xC0D27008 | $\begin{aligned} & \text { ECM_ERROR_SOE_SSC_OPDATA } \\ & \text { _IS_INVALID } \end{aligned}$ | SoE: OpData is invalid |
| 0xC0D27009 | ECM_ERROR_SOE_SSC_OPDATA IS_WRITE_PRO- <br> TEC̄TED_BȲ_PASSWORD | SoE: OpData is write protected by password |
| 0xC0D2700A | ECM_ERROR_SOE_SSC_OPDATA IS_WRITE_PRO- <br> T̄EC̄TED_DŪE_CYCLICALLY_CONFIGURED | SoE: OpData is write protected due to being cyclically configured |
| 0xC0D2700B | ECM ERROR SOE SSC OPDATA _INVĀLID_DIRECT_ADDRESSING | SoE: Invalid direct addressing |
| 0xC0D2700C | ECM ERROR SOE SSC OPDATA IS WRITE PRO- <br> TECTED_DUE_OTHER_SETTINGS | SoE: OpData is write protected due to other settings. |
| 0xC0D2700D | ECM_ERROR_SOE_SSC_OPDATA INVĀLID_FLOATING_POINT_NUM BER | SoE: Invalid floating point number |
| 0xC0D2700E | ECM_ERROR_SOE_SSC_OPDATA IS WRITE PRO-TECTTED_AT_PARAMETERIZATION_LEVEL | SoE: OpData is write protected at parameterization level |
| 0xC0D2700F | ECM_ERROR_SOE_SSC_OPDATA IS_WRITE_PRO- <br> T̄EC̄TED_AT_OPERATION_LEVEL | SoE: OpData is write protected at operation level |
| 0xC0D27010 | ECM_ERROR_SOE_SSC_OPDATA PROCCEDURE COMMAND_ALREADYY_ACTIVE | SoE: Procedure command already active |
| 0xC0D27011 | ECM_ERROR_SOE_SSC_OPDATA PRŌCEDURE_COMMAND_NOT_INTERRUPTIBLE | SoE: Procedure command not interruptible |
| 0xC0D27012 | ECM_ERROR_SOE_SSC_OPDATA PRŌCEDURE_COM-MAND_NOT_EXECUTABLE_AT_THIS_TIME | SoE: Procedure command is not executable at this time |
| 0xC0D27013 | ECM_ERROR_SOE_SSC_OPDATA PRŌCEDURE COMMAND NOT EXECUTABLE_INVALID_PARAM | SoE: Procedure command is not executable due to invalid parameter |
| 0xC0D4005C | ECM_ERROR_ENI_NO_SLAVES_I N_ENI | ENI does not contain any slaves |
| 0xC0D50001 | ECM ERROR ALSTATCODĒ_UNSPĒCIFIED_ERROR | ALStatusCode: Unspecified error |
| 0xC0D50002 | ECM_ERROR_ALSTATCODĒ_NO_MEMORY | ALStatusCode: No memory |
| 0xC0D50003 | ECM_ERROR_ALSTATCODE_INVALID_DEVICE_SETUP | ALStatusCode: Invalid Device Setup |
| 0xC0D50011 | ECM_ERROR_ALSTATCOD $\bar{E}$ _INVALID_REQUESTED_ST ATE_CHANGE | ALStatusCode: Invalid requested state change |


| Hexadecimal Value | Definition | Description |
| :--- | :--- | :--- |
| 0xC0D50012 | ECM_ERROR_ALSTAT- <br> CODE_UNKNOWN_REQUESTED_- <br> STATE | ALStatusCode: Unknown requested <br> state |
| 0xC0D50013 | ECM_ERROR_ALSTAT- <br> CODE_BOOTSTRAP_NOT_SUP- <br> PORTED | ALStatusCode: <br> pootstrap not sup- |
| 0xC0D50014 | ECM_ERROR_ALSTAT- <br> CODE_NO_VALID_FIRMWARE | ALStatusCode: No valid firmware |
| 0xC0D50015 | ECM_ERROR_ALSTAT- <br> CODE_INVALID_BOOT_MAILBOX_- <br> CONFIGURATION | ALStatusCode: Invalid BOOT <br> mailbox configuration |
| 0xC0D50016 | ECM_ERROR_ALSTAT- | ALStatusCode: Invalid PREOP <br> CODE_INVALID_PREOP_MAILBOX |
| mailbox configuration |  |  |


| Hexadecimal Value | Definition | Description |
| :--- | :--- | :--- |
| 0xC0D50025 | ECM_ERROR_ALSTAT- <br> CODE_INVALID_OUTPUT_MAP- <br> PING | ALStatusCode: Invalid Output Map- <br> ping |
| 0xC0D50026 | ECM_ERROR_ALSTAT- <br> CODE_INCONSISTENT_SET- | ALStatusCode: Inconsistent settings |
|  | TINGS |  |
| 0xC0D50027 | ECM_ERROR_ALSTAT- <br> CODE_FREERUN_NOT_SUP- <br> PORTED | ALStatusCode: FreeRun not sup- <br> ported |
| 0xC0D50028 | ECM_ERROR_ALSTAT- <br> CODE_SYNCMODE_NOT_SUP- <br> PORTED | ALStatusCode: SyncMode not sup- <br> ported |
| 0xC0D50029 | ECM_ERROR_ALSTAT- | ALStatusCode: FreeRun needs <br> CODE_FREERUN_NEEDS_3BUFF |
| 3Buffer mode |  |  |


| Hexadecimal Value | Definition | Description |
| :---: | :---: | :---: |
| 0xC0D50044 | ECM ERROR_ALSTATCODE_MBX_FOE | ALStatusCode: MBX_FOE |
| 0xC0D50045 | ECM_ERROR_ALSTATCODE MBX S̄OE | ALStatusCode: MBX_SOE |
| 0xC0D5004F | ECM ERROR_ALSTATCODĒ_MBX_VOE | ALStatusCode: MBX_VOE |
| 0xC0D50050 | ECM_ERROR_ALSTATCODĒ_EEPRŌM_NO_ACCESS | ALStatusCode: EEPROM no access |
| 0xC0D50051 | ECM ERROR ALSTATCODĒ_EEPRŌM_ERROR | ALStatusCode: EEPROM error |
| 0xC0D50060 | ECM_ERROR_ALSTATCODĒ_SLAVE_RESTARTED_LOC ALLY | ALStatusCode: Slave restarted locally |
| 0xC0D50061 | ECM ERROR ALSTATCODE DEVICE IDENTIFICATION_VILUE_UPDATED | ALStatusCode: Device identificatin value updated |
| 0xC0D500F0 | ECM_ERROR_ALSTATCODĒ APPLICATION CONTROLLER_AVAILABLE | ALStatusCode: Application controller available |
| 0xC0D58000 | ECM ERROR_ALSTATCODĒ VENDŌR_SPECIFIC_CODE_START | Begin of vendor-specific ALStatusCode mapping |
| 0xC0D5FFFF | ECM_ERROR_ALSTAT-CODĒ_VENDŌR_SPECIFIC_CODE_END | End of vendor-specific ALStatusCode mapping |
| 0xC0D60001 | ECM ERROR_IF_COE_SUPPORT_NOT_AVAIILABLE | CoE support is not configured |
| 0xC0D60002 | ECM ERROR_IF_SOE_SUPPORT_NOT_AVAIILABLE | SoE support is not configured |
| 0xC0D60003 | ECM_ERROR_IF_FOE_SUPPORT_NOT_AVAILABLĒ | FoE support is not configured |
| 0xC0D60004 | ECM ERROR IF AOE SUPPORT_NOT_AVAIILABLE | AoE support is not configured |
| 0xC0D60005 | ECM_ERROR_IF_INVALID_TRANS PORT̄_TYPE | Invalid transfer type |
| 0xC0D60006 |  | SoE: Invalid drive number |
| 0xC0D60007 | ECM_ERROR_IF_SOE_INVALID_E LEMĒNT_FLAḠS | SoE: invalid element flags |
| 0xC0D60008 | ECM_ERROR_IF_INVALID_SOE_T RANSFER_ID | SoE: Invalid transfer ID |
| 0xC0D60009 | ECM_ERROR_IF_TRANSFER_AB ORTĒD | Transfer aborted |
| 0xC0D6000A | $\begin{aligned} & \text { ECM_ERROR_IF_OUT_OF_PACKE } \\ & \text { TS } \end{aligned}$ | Out of packets |
| 0xC0D6000B | ECM_ERROR_IF_OUT_OF_TRAN SFER_CONTEXTS | Out of transfer contexts |
| 0xC0D6000C | ECM_ERROR_IF_INVALID_SUBINDEX_FOR_COMPLETE_ACCESs | CoE: Invalid subindex for Complete Access |


| Hexadecimal Value | Definition | Description |
| :--- | :--- | :--- |
| $0 x C 0 D 6000 \mathrm{D}$ | ECM_ERROR_IF_INVALID_COE_T <br> RANSFER_ID | CoE: Invalid transfer ID |
| $0 x$ C0D6000E | ECM_ERROR_IF_INVALID_COE_S <br> DOINFO_LISTTYPE | CoE: Invalid SDOINFO ListType |
| $0 x C 0 D 6000 \mathrm{~F}$ | ECM_ERROR_IF_FILE_READ_ER <br> ROR | File Read Error |
| $0 x C 0 D 60010$ | ECM_ERROR_IF_COULD_NOT_O <br> PEN_FILE | Could not open file |
| $0 x C 0 D 60011$ | ECM_ERROR_IF_INVALID_CONFI <br> G_NXD | Invalid config.nxd detected |
| $0 x C 0 D 60012$ | ECM_ERROR_IF_CONFIG_NXD_- <br> WITHOUT_SLAVES | Config.nxd does not contain any <br> slaves |
| $0 x C 0 D 60013$ | ECM_ERROR_IF_INVALID_FILE_N <br> AME | Invalid file name |
| $0 x C 0 D 60014$ | ECM_ERROR_IF_INVALID_FOE_T <br> RANSFER_ID | Invalid FoE transfer id |
| $0 x C 0 D 60015$ | ECM_ERROR_IF_INVALID_GET_T <br> OPOLOGY_TRANSFER_ID | Invalid GetTopology transfer id |

### 6.9.3.6.3 CM589-PNIO(-4) errors

In PLC display the diagnosis messages of "CM589-PNIO" are not shown.
The following diagnosis messages are signaled by "CM589-PNIO":

| Error <br> severit <br> y | SubSyste- <br> minfo | Addi- <br> tional | Error code | Meaning | Remedy |
| :--- | :--- | :--- | :--- | :--- | :--- |
| 3 | 0 | 0 | 1000 | No communication <br> module or wrong <br> type found. | Plug the correct com- <br> munication module. |
| 3 | 0 | 0 | 1001 | Type of CM589- <br> PNIO not supported. | Exchange the com- <br> munication module. |
| 3 | 0 | 0 | 1002 | Firmware version of <br> CM589-PNIO not <br> supported. | Update firmware of <br> CM589-PNIO. |
| 3 | 0 | 0 | 1003 | Identification of com- <br> munication module <br> failed. | Exchange the com- <br> munication module or <br> plug the correct com- <br> munication module. |
| 3 | 0 | 0 | 2000 | Watchdog error |  |
| 3 | 0 | 0 | 2001 | CM589-PNIO is not <br> communicating. | Check bus connection <br> and configuration. |
| 3 | 0 | 0 | 2002 | CM589-PNIO signals <br> communication error. | Check bus connection <br> and configuration. |
| 3 | 0 | 0 | 2003 | Starting of CM589- <br> PNIO's protocol <br> stack failed. | Check bus connection <br> and configuration. |


| Error <br> severit <br> y | SubSyste- <br> minfo | Addi- <br> tional | Error code | Meaning | Remedy |
| :--- | :--- | :--- | :--- | :--- | :--- |
| 3 | 0 | 0 | 2004 | Stopping of CM589- <br> PNIO 's protocol <br> stack failed. |  |
| 3 | 0 | 0 | 2005 | PLC cannot be set to <br> run due to an error of <br> CM589-PNIO. | Check error log and <br> correct errors. |
| 3 | 0 | 0 | 3000 | Configuration error | Check configuration <br> and correct errors. |
| 3 | 0 | 0 | 3001 | Configuration version <br> mismatch | Use matching CPU <br> firmware version. |
| 3 | 0 | 0 | 4000 | Ethernet link down | Check Ethernet cable <br> connection. |
| 3 | 0 | 0 | 10000 | Wrong module is <br> configured. | Check configuration <br> and correct errors. |
| 3 | 0 | 0 | 10001 | Correct module <br> is configured but <br> locked by another <br> I/O controller. | Check configuration <br> and correct errors. |
| 3 | 0 | 0 | 10002 | Module is configured <br> but not used by I/O <br> controller. | Check configuration <br> and correct errors. |
| 3 | Slot/subslot | 0 | 65636 | Slot/subslot: No <br> module configured. | Check configuration <br> and correct errors. |

### 6.9.3.6.4 CM598-CN CAN communication module diagnosis

In Automation Builder, diagnosis messages of communication module CM598-CN are displayed at device tree node "CM598-CN" and all nodes below,CANopen remote devices and I/O modules (for CI58x modules only).
Click tab "Diagnosis".
Within PLC application, diagnosis messages can be read by diagnosis related methods of function block type "Diag", provided in library "Diag".
In PLC display, diagnosis messages of CM598-CN are not shown.
Following diagnosis messages are signaled by CM598-CN.
CM598-CN communication module specific diagnosis messages:

| Severity | SubSyste- <br> minfo | Additional | Error code | Meaning | Remedy |
| :--- | :--- | :--- | :--- | :--- | :--- |
| 3 | 0 | 0 | 6554600 | CM598-CN <br> not found | Plug the cor- <br> rect communi- <br> cation module |
| 3 | 0 | 0 | 6554601 | Type of <br> CM598-CN <br> not supported | Exchange the <br> communica- <br> tion module |
| 3 | 0 | 0 | 6554602 | Firmware ver- <br> sion of <br> CM598-CN <br> not supported | Update firm- <br> ware of <br> CM598-CN |


| Severity | SubSysteminfo | Additional | Error code | Meaning | Remedy |
| :---: | :---: | :---: | :---: | :---: | :---: |
| 3 | 0 | 0 | 6554603 | Identification <br> of CM598-CN <br> failed | Exchange the communication module |
| 3 | 0 | 0 | 6555600 | Watchdog error |  |
| 4 | 0 | 0 | 6555601 | CM598-CN is not communicating | Check bus connection and configuration |
| 4 | 0 | 0 | 6555602 | CM598-CN signals communication error | Check bus connection and configuration |
| 3 | 0 | 0 | 6555603 | Starting of CM598-CN's protocol stack failed | Check bus connection and configuration |
| 3 | 0 | 0 | 6555604 | Stopping of CM598-CN's protocol stack failed |  |
| 3 | 0 | 0 | 6555605 | PLC cannot be set to run due to an error of CM598-CN | Check error log and correct errors |
| 3 | 0 | 0 | 6556600 | Configuration error | Check configuration and correct errors |
| 3 | 0 | 0 | 6556601 | Configuration version mismatch | Use matching CPU firmware version |
| 3 | 0 | 0 | 6557600 | Starting of CM598-CN's SYNC task failed |  |
| 3 | 0 | 0 | 6557600 | Stopping of CM598-CN's SYNC task failed |  |
| 3 | 0 | 0 | 6558600 | Starting CAN 2A/2B failed |  |
| 3 | 0 | 0 | 6558601 | Stopping CAN 2A/2B failed |  |

CANopen common diagnosis, shown at CANopen remote device.

| Severity | SubSysteminfo | Additional | Error code | Meaning | Remedy |
| :---: | :---: | :---: | :---: | :---: | :---: |
| 3 | 0 | 0 | 65536 | SDO timeout |  |
| 3 | 0 | 0 | 65537 | Error during SDO transfer |  |
| 3 | 0 | 0 | 65538 | Configuration fault | Check configuration and correct errors |
| 3 | 0 | NMT_state <br> 0=unknown <br> 1=initializing <br> 2=stopped <br> 3=operational <br> 4=pre-operational <br> 5=reset_application <br> 6=reset_com m | 65543 | Node is in unexpected NMT state <NMT state> |  |
| 3 | 0 | 0 | 65545 | Emergency buffer overflow |  |
| 3 | 0 | 0 | 65547 | Unexpected boot up message from node received |  |
| 3 | 0 | 0 | 65548 | Parameter set of node is invalid | Check parameter set and correct errors |
| 3 | 0 | 0 | 131073 | Node configuration error | Check configuration and correct errors |
| 3 | 0 | 0 | 131074 | Node not communicating | Check bus connection and configuration |

CANopen EMCY messages, shown at CANopen remote devices.
Manufacturer specific data can be viewed as raw bytes.

| Severity | SubSyste- <br> minfo | Additional | Error code | Meaning | Remedy |
| :--- | :--- | :--- | :--- | :--- | :--- |
| 3 | 0 | Error register | 200704 | Generic error; <br> error register: <br> <error reg- <br> ister> |  |
| 3 | 0 | Error register | 204800 | Current - <br> generic error; <br> error register: <br> <error reg- <br> ister> |  |
| 3 | 0 | Error register | 205056 | Current, <br> CANopen <br> device input <br> side - generic; <br> error register: <br> <error reg- <br> ister> |  |
| 3 | 0 | Error register | 205312 | Current inside <br> the CANopen <br> device - <br> generic; error <br> register: <br> <error reg- <br> ister> |  |
| 3 | 0 | Error register | 209408 | Voltage inside <br> the CANopen <br> device - <br> generic; error <br> register: <br> <error reg- <br> ister> |  |
| 3 | 0 | Error register | 205568 | Current, <br> CANopen <br> device output <br> side - generic; <br> error register: <br> <error reg- <br> ister> |  |
|  | 0 | 208896 | Voltage - <br> generic error; <br> error register: <br> <error reg- <br> ister> |  |  |
| 3 | 0 | Error register | 209152 | Mains voltage <br> - generic; <br> error register: <br> <error reg- <br> ister> |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |


| Severity | SubSyste- <br> minfo | Additional | Error code | Meaning | Remedy |
| :--- | :--- | :--- | :--- | :--- | :--- |
| 3 | 0 | Error register | 209664 | Ouput voltage <br> -generic; <br> error register: <br> <error reg- <br> ister> |  |
| 3 | 0 | Error register | 212992 | Temperatur - <br> generic error; <br> error register: <br> <error reg- <br> ister> |  |
| 3 | 0 | Error register | 213248 | Ambient tem- <br> perature - <br> generic error; <br> error register: |  |
| <error reg- |  |  |  |  |  |
| ister> |  |  |  |  |  |$|$


| Severity | SubSysteminfo | Additional | Error code | Meaning | Remedy |
| :---: | :---: | :---: | :---: | :---: | :---: |
| 3 | 0 | Error register | 225280 | Additional modules generic error; error register: <error register> |  |
| 3 | 0 | Error register | 229376 | Monitoringgeneric error error register: <error register> |  |
| 3 | 0 | Error register | 229632 | Communication - generic; error register: <error register> |  |
| 3 | 0 | Error register | 229684 | CAN overrun (objects lost); error register: <error register> |  |
| 3 | 0 | Error register | 229664 | CAN in error passive mode; error register: <error register> |  |
| 3 | 0 | Error register | 229680 | Life guard error or heartbeat error; error register: <error register> |  |
| 3 | 0 | Error register | 229696 | Recovered from bus off; error register: <error register> |  |
| 3 | 0 | Error register | 229712 | CAN-ID colli- <br> sion; error register: <error register> |  |
| 3 | 0 | Error register | 229888 | Protocol error - generic; error register: <error register> |  |
| 3 | 0 | Error register | 229904 | PDO not processed due to length error; error register: <error register> |  |


| Severity | SubSysteminfo | Additional | Error code | Meaning | Remedy |
| :---: | :---: | :---: | :---: | :---: | :---: |
| 3 | 0 | Error register | 229920 | PDO length exceeded; error register: <error register> |  |
| 3 | 0 | Error register | 229936 | DAM MPDO not processed, destination object not available; error register: <error register> |  |
| 3 | 0 | Error register | 229952 | Unexpected SYNC data length; error register: <error register> |  |
| 3 | 0 | Error register | 229968 | RPDO timeout; error register: <error register> |  |
| 3 | 0 | Error register | 233472 | External error <br> - generic <br> error; error register: <error register> |  |
| 3 | 0 | Error register | 258084 | Additional functions generic error; error register: <error register> |  |
| 3 | 0 | Error register | 261888 | Device specific - generic error; error register: <error register> |  |

ABB Communication Interface Module (CI58x) specific diagnosis messages, shown at I/O modules directly instead of CANopen remote device.

| Severity | SubSyste- <br> minfo | Additional | Error code | Meaning | Remedy |
| :--- | :--- | :--- | :--- | :--- | :--- |
| 3 | 255 | 0 | 8722 | Internal error |  |
| 3 | 255 | 0 | 8732 | Internal error |  |


| Severity | SubSysteminfo | Additional | Error code | Meaning | Remedy |
| :---: | :---: | :---: | :---: | :---: | :---: |
| 4 | 255 | 0 | 9480 | I/O module removed from hot swap terminal unit or defective module on hot swap terminal unit | Plug I/O module, replace I/O module |
| 4 | 255 | 0 | 9500 | Wrong I/O module plugged on hot swap terminal unit | Remove wrong I/O module and plug projected I/O module |
| 4 | 255 | 0 | 9514 | No communication with I/O module on hot swap terminal unit | Replace I/O module |
| 4 | 255 | 0 | 9526 | I/O module does not support hot swap | Power off system and replace I/O module |
| 4 | 255 | 0 | 9736 | Hot swap terminal unit required but not found | Plug hot swap terminal unit |
| 4 | 255 | 0 | 9764 | Defective hot swap terminal unit |  |
| 4 | 255 | 0 | 9770 | No communication with hot swap terminal unit | Restart, if error persists replace terminal unit |
| 3 | 255 | 0 | 16131 | Timeout in the I/O module | Replace I/O module |
| 3 | 255 | 0 | 16137 | Overflow diagnosis buffer | Restart |
| 4 | 255 | 0 | 16138 | Voltage overflow at outputs (above UP3 level) | Check terminals/check process supply voltage |
| 3/4 | 255 | 0 | 16139 | Process voltage UP or UP3 too low | Check process supply voltage |
| 3 | 255 | 0 | 16145 | No communication with I/O module | Replace I/O module |
| 3 | 255 | 0 | 16147 | Checksum error in the I/O module | Replace I/O module |
| 3 | 255 | 0 | 16154 | Parameter error | Check master |


| Severity | SubSysteminfo | Additional | Error code | Meaning | Remedy |
| :---: | :---: | :---: | :---: | :---: | :---: |
| 4 | 255 | 0 | 16159 | At least one module does not support failsafe function | Check modules and parameterization |
| 3 | 255 | 0 | 16160 | Wrong I/O module type on socket | Replace I/O module / check configuration |
| 4 | 255 | 0 | 16162 | No response during initialization of the I/O module | Replace I/O module |
| 3 | 255 | 0 | 16164 | Internal data exchange failure | Replace I/O module |
| 3 | 255 | 0 | 16168 | Different hard-/ firmware versions in the module | Replace I/O module |
| 3 | 255 | 0 | 16171 | Internal error in the module | Replace I/O module |
| 3/4 | 255 | 0 | 16173 | No process voltage UP or UP3 | Check process voltage |
| 4 | 255 | 0 | 16174 | Voltage feedback on activated digital outputs DO0...DO7 on UP3 | Check terminals |
| 3 | 0-31 | 0 | 18 | Test error |  |
| 4 | 0-31 | 0 | 257 | Wrong measurement, false temperature at the compensation channel |  |
| 4 | 0-31 | 0 | 258 | Al531: Wrong measurement; potential difference is too high; CD522: PWM duty cycle out of duty area |  |
| 4 | 0-31 | 0 | 260 | Measurement overflow | Check channel wiring and sensor power supply |
| 4 | 0-31 | 0 | 263 | Measurement underflow at analog input | Check channel wiring and sensor power supply |


| Severity | SubSysteminfo | Additional | Error code | Meaning | Remedy |
| :---: | :---: | :---: | :---: | :---: | :---: |
| 4 | 0-31 | 0 | 266 | Short circuit and cut wire or "out of range" |  |
| 4 | 0-31 | 0 | 267 | Output/ process voltage to small/low |  |
| 3 | 0-31 | 0 | 273 | Test error |  |
| 4 | 0-31 | 0 | 303 | Short circuit at an analog input | Check channel wiring |
| 4 | 0-31 | 0 | 304 | Analog value overflow or broken wire at an analog input | Check value or check terminals |
| 4 | 0-31 | 0 | 530 | Internal fuse at 0 V is defect. OV not connected with GND | Check I/O module wiring. Restart I/O module, if needed. If this error persists, replace I/O module |
| 3 | 0-31 | 0 | 540 | Test error |  |
| 3 | 0-31 | 0 | 555 | Internal error |  |
| 4 | 0-31 | 0 | 558 | Externally voltage detected on digital output DO0...DO7 | Check terminals |
| 4 | 0-31 | 0 | 559 | Short circuit at digital output | Check channel wiring |
| 4 | 0-31 | 0 | 772 | Analog value overflow at an analog output | Check output value |
| 4 | 0-31 | 0 | 775 | Analog value underflow at an analog output | Check output value |
| 4 | 0-31 | 0 | 796 | Different configuration |  |
| 3 | 0-31 | 0 | 1037 | Test error |  |
| 4 | 0-31 | 0 | 1070 | Externally voltage detected on digital output DC0...DC7 | Check terminals |
| 4 | 0-31 | 0 | 1071 | Short circuit at digital output | Check terminals |

### 6.9.3.6.5 CM582-DP PROFIBUS DP slave diagnosis

The diagnosis messages of the communication module CM582-DP are displayed in the tab "Diagnosis" of node "CM582-DP" in the device tree of the Automation Builder. Within PLC application they can be read with the diagnosis methods of IO driver or function block "Diag".

In the PLC display the diagnosis messages of CM582-DP are not shown.
The following diagnosis messages are signaled by CM582-DP:
\(\left.$$
\begin{array}{|l|l|l|l|l|l|}\hline \begin{array}{l}\text { Error } \\
\text { severity }\end{array} & \text { SubSysteminfo } & \text { Additional } & \text { Error code } & \text { Meaning } & \text { Remedy } \\
\hline 3 & 0 & 0 & 1000 \mathrm{x} & \begin{array}{l}\text { No communication module } \\
\text { or wrong type found }\end{array} & \begin{array}{l}\text { Plug the correct com- } \\
\text { munication module }\end{array} \\
\hline 3 & 0 & 0 & 1001 & \begin{array}{l}\text { Type of CM582-DP not sup- } \\
\text { ported }\end{array} & \begin{array}{l}\text { Exchange the com- } \\
\text { munication module }\end{array} \\
\hline 3 & 0 & 0 & 1002 & \begin{array}{l}\text { Firmware version of CM582- } \\
\text { DP not supported }\end{array} & \begin{array}{l}\text { Update firmware of } \\
\text { CM582-DP }\end{array} \\
\hline 3 & 0 & 0 & 1003 & \begin{array}{l}\text { Identification of communica- } \\
\text { tion module failed }\end{array} & \begin{array}{l}\text { Exchange the com- } \\
\text { munication module or } \\
\text { plug the correct com- } \\
\text { munication module }\end{array} \\
\hline 3 & 0 & 0 & 2000 & \text { Watchdog error } & \text { - } \\
\hline 3 & 0 & 0 & 2001 & \begin{array}{l}\text { CM582-DP is not communi- } \\
\text { cating }\end{array} & \begin{array}{l}\text { Check bus connection } \\
\text { and configuration }\end{array} \\
\hline 3 & 0 & 0 & 2003 & \begin{array}{l}\text { CM582-DP signals commu- } \\
\text { nication error }\end{array} & \begin{array}{l}\text { Check bus connection } \\
\text { and configuration }\end{array} \\
\hline 3 & 0 & 0 & \begin{array}{l}\text { Starting of CM582-DP's pro- } \\
\text { tocol stack failed }\end{array} & \begin{array}{l}\text { Check bus connection } \\
\text { and configuration }\end{array} \\
\hline 3 & 0 & 0 & 2004 & \begin{array}{l}\text { Stopping of CM582-DP's } \\
\text { protocol stack failed }\end{array} & \begin{array}{l}\text { - }\end{array} \\
\hline 3 & 0 & 0 & 3005 & \begin{array}{l}\text { PLC cannot be set to run } \\
\text { due to an error of CM582- } \\
\text { DP }\end{array} & \begin{array}{l}\text { Check error log and } \\
\text { correct errors }\end{array} \\
\hline 3 & 0 & 0 & 3002 & \begin{array}{l}\text { Configuration error }\end{array} & \begin{array}{l}\text { Check configuration } \\
\text { and correct errors }\end{array} \\
\hline 3 & 0 & 0 & 3004 & \begin{array}{l}\text { Configuration version mis- } \\
\text { match }\end{array} & \begin{array}{l}\text { Use matching CPU } \\
\text { firmware version }\end{array} \\
\hline 3 & 0 & 0 & \begin{array}{l}\text { Writing parameters to } \\
\text { CM582-DP failed }\end{array} & \begin{array}{l}\text { Check configuration } \\
\text { and correct errors }\end{array}
$$ <br>

\hline ported or out of limits\end{array}\right\}\)| Configuration of IM0 data |
| :--- |
| failed |

### 6.9.3.6.6 AC500-S: errors from safety CPU and safety I/O modules

Table 865: Error messages for safety CPU

| Severity | Error code | Description | Remedy |
| :---: | :---: | :---: | :---: |
| 2 | 8235 | Internal error | Replace module |
| 2 | 8448 | Operation finished | Change Safety PLC switch address setting or remove memory card from non-safety PLC. Restart Safety PLC. If this error persists, replace Safety PLC. |
| 2 | 8449 | Wrong user data | Delete user data from Safety PLC. Restart Safety PLC and write user data again. |
| 2 | 8450 | Internal PROFIsafe initialization error | Restart Safety PLC. If this error persists, replace Safety PLC. Contact ABB technical support. |
| 2 | 8460 | Flash read error | Restart Safety PLC. If this error persists, replace Safety PLC. Contact ABB technical support. |
| 2 | 8466 | Internal error | Contact ABB technical support. Replace Safety PLC. |
| 2 | 8476 | Boot project download error | Reload boot project. If this error persists, replace Safety PLC. |
| 2 | 8488 | Wrong firmware version | Update Safety PLC firmware. Restart Safety PLC. If this error persists, replace Safety PLC. |
| 2 | 8491 | Internal error | Contact ABB technical support. Replace Safety PLC. |
| 2 | 8496 | Overvoltage or undervoltage detected | Restart Safety PLC. Check Safety PLC setting for power supply error. If this error persists, replace Safety PLC. |
| 2 | 8500 | Internal error | Contact ABB technical support. Replace Safety PLC. |
| 2 | 8704 | User program triggered safe stop | Check user program |
| 2 | 8705 | Internal error | Contact ABB technical support. Replace Safety PLC. |
| 2 | 8706 | Internal PROFIsafe error | Restart Safety PLC. If this error persists, replace Safety PLC. Contact ABB technical support. |
| 2 | 8707 | Internal error | Contact ABB technical support. Replace Safety PLC. |
| 2 | 8714 | Internal error | Contact ABB technical support. Replace Safety PLC. |
| 2 | 8717 | Flash write error | Restart Safety PLC. If this error persists, replace Safety PLC. Contact ABB technical support. |
| 2 | 8721 | Internal error | Contact ABB technical support. Replace Safety PLC. |
| 2 | 8722 | Internal error | Contact ABB technical support. Replace Safety PLC. |


| Severity | Error code | Description | Remedy |
| :--- | :--- | :--- | :--- |
| 2 | 8723 | Checksum error has <br> occured in Safety PLC | Restart Safety PLC. If this error <br> persists, replace Safety PLC. |
| 2 | 8729 | Internal error | Contact ABB technical support. <br> Replace Safety PLC. |
| 2 | 8741 | Cycle time error in Safety <br> PLC | Check Safety PLC watchdog <br> time. |
| 2 | 8742 | Internal error | Contact ABB technical support. <br> Replace Safety PLC. |
| 2 | 8746 | Internal error | Contact ABB technical support. <br> Replace Safety PLC. |
| 2 | 8756 | Internal error | Contact ABB technical support. <br> Replace Safety PLC. |
| 2 | 8990 | Internal error | Contact ABB technical support. <br> Replace Safety PLC. |
| 2 | 32561 | Internal error <br> PROFIsafe configuration <br> error | Contact ABB technical support. <br> Replace Safety PLC. |
| Check F-Parameter configura- |  |  |  |
| tion of I/O module and reload |  |  |  |
| boot project |  |  |  |\(\left|\begin{array}{l}Safety source addresses <br>

cannot be checked\end{array} \begin{array}{l}Check PROFIsafe F-Host <br>
library version (2.0.0 or above). <br>
If this error persists, contact <br>

ABB technical support.\end{array}\right|\)| 2758 |
| :--- |
| 3 |


| Severity | Error code | Description | Remedy |
| :---: | :---: | :---: | :---: |
| 3 | 32776 | Safety module has wrong type | Check configuration |
| 4 | 16640 | Reserved switch address setting. | Warning |
| 4 | 16644 | Boot project not loaded, maximum power dip reached | Restart Safety PLC |
| 4 | 16648 | Power dip data missed or corrupted. Default power dip data was flashed by Safety PLC | Warning |
| 4 | 16659 | CRC error boot project | Create new boot project and restart Safety PLC |
| 4 | 16909 | Flash write error (production data) | Warning |
| 4 | 16935 | More than one instance of SF_WDOG_TIME_SET or SF_MAX_POWER_DIP_S ET | Warning |
| 4 | 16922 | No or wrong configuration data from PM5x, run state not possible | Create correct boot project at PM5x |
| 4 | 17421 | Flash write error (boot project) | Warning |
| 4 | 17677 | Flash write error (boot code) | Warning |
| 4 | 17933 | Flash write error (firmware) | Warning |
| 4 | 18189 | Flash write error (password) | Warning |
| 4 | 18445 | Flash write error (user data) | Warning |
| 4 | 18701 | Flash write error (user data) | Warning |
| 4 | 18957 | Flash write error (internal) | Warning |
| 4 | 19213 | Flash write error (internal) | Warning |
| 4 | 19469 | Flash write error (internal) | Warning |
| 4 | 32777 | Program not started because of configuration error | Check configuration |
| 4 | 32778 | Program not started, no application running in safety module | Check configuration, download safety application to safety module |

Table 866: Error messages for safety I/O modules (channel or module reintegration is possible)

| Severity | Error code | Description | Remedy |
| :--- | :--- | :--- | :--- |
| 3 | 3 | Discrepancy time expired | Check discrepancy time value, <br> channel wiring and sensor. |
| 3 | 12 | Test pulse error | Check wiring and sensor. |


| Severity | Error code | Description | Remedy |
| :---: | :---: | :---: | :---: |
| 3 | 13 | Channel test pulse crosstalk error | Check wiring and sensor. If this error persists, replace I/O module. Contact ABB technical support. |
| 3 | 25 | Channel stuck-at error | Check I/O module wiring. Restart I/O module, if needed. If this error persists, replace I/O module. |
| 3 | 28 | Channel cross-talk error | Check I/O module wiring. Restart I/O module, if needed. If this error persists, replace I/O module. |
| 3 | 260 | Measurement overflow at the I/O module | Check channel wiring and sensor power supply. |
| 3 | 263 | Measurement underflow at the I/O module | Check channel wiring and sensor power supply. |
| 3 | 311 | Channel value difference too high | Adjust tolerance window for channels. Check channel wiring and sensor configuration. |
| 3 | 525 | Channel readback error | Check I/O module wiring. Restart I/O module, if needed. If this error persists, replace I/O module. |
| 3 | 530 | Channel cross-talk error | Check I/O module wiring. Restart I/O module, if needed. If this error persists, replace I/O module. |
| 3 | 16138 | Process voltage too high | Check process voltage |
| 3 | 16139 | Process voltage too low | Check process voltage |
| 3 | 16148 | PROFIsafe communication error | Restart I/O module. If this error persists, contact ABB technical support. |
| 3 | 16153 | PROFIsafe watchdog timed out. | Restart I/O module. If this error persists, increase PROFIsafe watchdog time. |
| 3 | 16171 | Internal error in the device | Replace I/O module |

Table 867: Error messages for safety I/O modules (channel or module reintegration ist not possible)

| Severity | Error code | Description | Remedy |
| :--- | :--- | :--- | :--- |
| 3 | 16146 | Plausibility check failed <br> (iParameter) | Check configuration |
| 3 | 16147 | Checksum error in the I/O <br> module | Check safety configuration and <br> CRCs for I-and F-Parameters. |
| 3 | 16154 | Parameter value | Check master or configuration |
| 3 | 16156 | F-Parameter configuration <br> and address switch value <br> do not match. | Check I/O module F-Param- <br> eter configuration and module <br> address switch value. |

### 6.9.3.6.7 CM579-PNIO - PROFINET I/O controller diagnosis

Diagnosis data for CM579-PNIO is not displayed in PLC display. In Automation Builder, we recommend to use methods with text output to get diagnosis messages in clear text format. E.g., DiagGetFirstValAndTxt.

Output string:
<timestamp>; <error severity>; <device name>; <error location>; error ID <id>: <error text>

Advanced
Direct access to the diagnosis data of CM579PNIO

If you need to access the diagnosis data directly, you have to interpret them manually
② Chapter 6.9.3.6.8 "Advanced: Manual interpretation of CM579-PNIO diagnosis" on page 4704

### 6.9.3.6.8 Advanced: Manual interpretation of CM579-PNIO diagnosis

General
Diagnosis messages are included in diagnosis text lists "Diag_PNIO_Controller" and "Diag_PNIO_Vendor ID_Device ID".
(4) We recommend that you refer to the example to learn how to perform the manual interpretation in the correct way.

| Erro <br> r <br> seve <br> rity | SubSysteminfo |  | Additional |  | Error code |  | Meaning | Rem edy |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | Word 2 (bit 16..31) | Word 1 (bit 0..15) | Word 2 <br> (bit 16..31) <br> ADD_SUB -TYPE | Word 1 (bit 0..15) ADD_TYP E | Word 2 (bit16...31) | Word 1 (bit0...15) |  |  |
|  | Sub1_ | Sub2 | Add_Word 1_Word 2 |  | Err_x or <br> Err_Word 1_Word 2 <br> (Word1/2 in hex format) |  |  |  |
| 3 | tbd | tbd | 0 | 1 (general) | Err_Gen_x |  | General error, <Err_Gen_x_text> |  |
| 3 | 0 | 0 | 0 | 2 (runtime) | Err_Rt_x |  | General error, <Err_Rt_x_text> |  |
| 3 | 0 | 0 | 0 | 2 (runtime) | 1 |  | Runtime error; communication module watchdog error |  |
| 3 | 0 | 0 | 0 | 2 (runtime) | 2 |  | Runtime error; PROFINET controller is not communicating |  |
| 3 | 0 | 0 | 0 | 2 (runtime) | 3 |  | Runtime error; PROFINET controller signals communication error |  |
| 4 | 0 | 0 | 0 | 2 (runtime) | 4 |  | Connection error; No connection to PROFINET I/O device |  |
| 11 | tbd | tbd | 0 | 3 (configuration) | Err_Cfg_x |  | Configuration error, <Err_Cfg_x_text> |  |


| Erro r seve rity | SubSysteminfo |  | Additional |  | Error code |  | Meaning | Rem edy |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | Word 2 (bit 16..31) | Word 1 (bit 0..15) | Word 2 <br> (bit 16..31) <br> ADD SUB -TYPE | Word 1 <br> (bit 0..15) <br> ADD_TYP <br> E | Word 2 (bit16...31) | Word 1 (bit0...15) |  |  |
|  | Sub1_ | Sub2 | Add_Word 1_Word 2 |  | Err_x or <br> Err_Word 1_Word 2 <br> (Word1/2 in hex format) |  |  |  |
| 3 | Subslot index (0 16\#9FFF) | Channel index (0 16\#7FFF) | 1 (USI: 16\#8000) channel diagnosis | 4 (diagnosis alarm) | 0 | Error type | Subslot <subslot idx>, channel <channel idx>, channel diagnosis; <error text> |  |
| 3 | Subslot index (0 16\#9FFF) | Channel index (16\#8000) | 1 (USI: 16\#8000) channel diagnosis | 4 (diagnosis alarm) | 0 | Error type | Subslot <subslot idx>, channel diagnosis; <error text> |  |
| 3 | Subslot index (0 16\#9FFF) | Channel index (0 16\#7FFF) | 2 (USI: 16\#8002) extended channel diagnosis | 4 (diagnosis alarm) | Extended error Type | Error type | Subslot <subslot idx>, channel <channel idx>, extended channel diagnosis; <error text> |  |
| 3 | Subslot index (0 16\#9FFF) | Channel index (16\#8000) | 2 (USI: 16\#8002) extended channel diagnosis | 4 (diagnosis alarm) | Extended error Type | Error type | Subslot <subslot idx>, extended channel diagnosis; <error text> |  |
| 3 | Subslot index (0 16\#9FFF) | Channel index (0 16\#7FFF) | 3 (USI: 16\#8003) qualified channel diagnosis | 4 (diagnosis alarm) | Extended error Type | Error type | Subslot <subslot idx>, channel <channel idx>, qualified channel diagnosis; <error text> |  |
| 3 | Subslot index (0 16\#9FFF) | Channel index (16\#8000) | 3 (USI: 16\#8003) qualified channel | 4 (diagnosis alarm) | Extended error Type | Error type | Subslot <subslot idx>, qualified channel diagnosis; <error text> |  |
| $\begin{aligned} & 2,3, \\ & 411 \end{aligned}$ | Subslot index (0 16\#9FFF) | Channel index (0 16\#7FFF) | 0 | 5 (S500 process alarm) | 32 bit error code |  | Subslot <subslot idx>, channel <channel idx>, S500 diagnosis; <error text> |  |
| $\begin{array}{ll} 2,3, \\ 411 \end{array}$ | Subslot index (0 16\#9FFF) | Channel index (16\#8000) | 0 | 5 (S500 process alarm) | 32 bit error code |  | Subslot <subslot idx>, S500 diagnosis; <error text> |  |
| 3 | Subslot index (0 16\#9FFF) | Channel index (0 16\#7FFF) | $\begin{aligned} & \text { Alarm type } \\ & =(14 . .30) \\ & \&(32 . .) \end{aligned}$ | 6 (alarm) | $\begin{aligned} & \text { Alarm type }=(14 . .30) \& \\ & (32 . .) \end{aligned}$ |  | Subslot <subslot idx>, channel <channel idx>, PNIO alarm; <error text> |  |
| 3 | Subslot index (0 16\#9FFF) | Channel index (16\#8000) | $\begin{aligned} & \text { Alarm type } \\ & =(14 . .30) \\ & \&(32 . .) \end{aligned}$ | 6 (alarm) | ${ }_{(32 \text {..) }}^{\text {Alarm }} \text { type }=(14 \text {.. 30) \& }$ |  | Subslot <subslot idx>, PNIO alarm; <error text> |  |
| 3 | Subslot index (0 16\#9FFF) | Channel index (0 16\#7FFF) | Alarm type = 1 | 6 (alarm) | Alarm type = 1 |  | Subslot <subslot idx>, channel <channel idx>, diagnosis alarm; <error text> |  |


| Erro <br> r <br> seve <br> rity | SubSysteminfo |  | Additional |  | Error code |  | Meaning | Rem edy |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | Word 2 <br> (bit 16..31) | Word 1 (bit 0..15) | Word 2 <br> (bit 16..31) <br> ADD_SUB _TYPE | Word 1 (bit 0..15) ADD_TYP E | Word 2 (bit16...31) | Word 1 (bit0...15) |  |  |
|  | Sub1_ | Sub2 | Add_Word 1_Word 2 |  | Err_x or <br> Err_Word 1_Word 2 <br> (Word1/2 in hex format) |  |  |  |
| 3 | Subslot index (0 16\#9FFF) | Channel index (16\#8000) | Alarm type = 1 | 6 (alarm) | Alarm type $=1$ |  | Subslot <subslot idx>, diagnosis alarm; <error text> |  |
| 3 | Subslot index (0 16\#9FFF) | Channel index (0 16\#7FFF) | Alarm type $=2$ | 6 (alarm) | Alarm type $=2$ |  | Subslot <subslot idx>, channel <channel idx>, process alarm; <error text> |  |
| 3 | Subslot index (0 16\#9FFF) | Channel index (16\#8000) | Alarm type = 2 | 6 (alarm) | Alarm type $=2$ |  | Subslot <subslot idx>, process alarm; <error text> |  |
| 3 | Subslot index (0 16\#9FFF) | Channel index (16\#8000) | Alarm type $=3$ | 6 (alarm) | Alarm type $=3$ |  | Subslot <subslot idx>, pull alarm |  |
| 3 | Subslot index (0 16\#9FFF) | Channel index (16\#8000) | Alarm type $=4$ | 6 (alarm) | Alarm type = 4 |  | Subslot <subslot idx>, plug alarm |  |
| 3 | Subslot index (0 16\#9FFF) | Channel index (16\#8000) | Alarm type $=5$ | 6 (alarm) | Alarm type $=5$ |  | Subslot <subslot idx>, status alarm |  |
| 3 | Subslot index (0 16\#9FFF) | Channel index (16\#8000) | Alarm type $=6$ | 6 (alarm) | Alarm type $=6$ |  | Subslot <subslot idx>, update alarm |  |
| 3 | Subslot index (0 16\#9FFF) | Channel index (16\#8000) | Alarm type $=7$ | 6 (alarm) | Alarm type $=7$ |  | Subslot <subslot idx>, rendundancy status changed alarm |  |
| 3 | Subslot index (0 16\#9FFF) | Channel index (16\#8000) | Alarm type = 8 | 6 (alarm) | Alarm type $=8$ |  | Subslot <subslot idx>, supervisor controlled alarm |  |
| 3 | Subslot index (0 16\#9FFF) | Channel index (16\#8000) | Alarm type = 9 | 6 (alarm) | Alarm type $=9$ |  | Subslot <subslot idx>, supervisor released alarm |  |
| 3 | Subslot index (0 16\#9FFF) | Channel index (16\#8000) | Alarm type $=10$ | 6 (alarm) | Alarm type $=10$ |  | Subslot <subslot idx>, wrong submodule plugged alarm |  |
| 3 | Subslot index (0 16\#9FFF) | Channel index (16\#8000) | Alarm type $=11$ | 6 (alarm) | Alarm type $=11$ |  | Subslot <subslot idx>, wrong submodule returned alarm |  |
| 3 | Subslot index (0 16\#9FFF) | Channel index (0 16\#7FFF) | Alarm type $=12$ | 6 (alarm) | Alarm type $=12$ |  | Subslot <subslot idx>, channel <channel idx>, diagnosis disappeared alarm |  |
| 3 | Subslot index (0 16\#9FFF) | Channel index (16\#8000) | Alarm type $=12$ | 6 (alarm) | Alarm type $=12$ |  | Subslot <subslot idx>, diagnosis disappeared alarm |  |


| Erro <br> r <br> seve <br> rity | SubSysteminfo |  | Additional |  | Error code |  | Meaning | Rem edy |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | Word 2 (bit 16..31) | Word 1 (bit 0..15) | Word 2 (bit 16..31) ADD SUB -TYPE | Word 1 <br> (bit 0..15) <br> ADD_TYP <br> E | Word 2 (bit16...31) | Word 1 (bit0...15) |  |  |
|  | Sub1_ | Sub2 | Add_Word 1_Word 2 |  | Err_x or <br> Err_Word 1_Word 2 <br> (Word1/2 in hex format) |  |  |  |
| 3 | Subslot <br> index (0 - <br> 16\#9FFF) | Channel index (16\#8000) | Alarm type $=13$ | 6 (alarm) | Alarm type $=13$ |  | Subslot <subslot idx>, port data changed alarm |  |
| 3 | Subslot index (0 16\#9FFF) | Channel index (16\#8000) | Alarm type $=31$ | 6 (alarm) | Alarm type $=31$ |  | Used module pulled alarm |  |

## Example

Type of diagnosis

For better understanding, we show the manual interpretation of CM579-PNIO diagnosis with an example.

System: AC500 CM579-PNIO + CI501-PNIO + optional S500 I/O inserted as PROFINET standard device
Error: discrepancy time expired (class 3, error ID 3) at channel 4 of first attached S500 I/O device on CI501-PNIO

For comparison: If a method with text output is used, e.g. DiagGetFirstValAndTxt the text output will be the string: Timestamp; E3; device name; subslot 1, channel 4, extended channel diagnosis; error ID 3: discrepancy time expired (class 3, error ID 3)

In Automation Builder the following error entry data is displayed:

| - Diag_Val[1] | AC500_DiagTypes.D... |  |  |
| :---: | :---: | :---: | :---: |
| - dtTimestamp | DATE_AND_TIME | DT $=2019-10-15-14: 48: 34$ | RTC time of event |
| - eClass | TECLASS | eDiagClass_3_Error | Severity of error event |
| - szDevice | STRING(80) | 'MO_PN03_M2' | Name of device |
| - eHwinterfaceld | TEHWID | eDiagHwId_Coupler2 | Identifier of hardware interface |
| - dwSubSystemInfo | DWORD | 65540 | Any number describin..ail/location within d... |
| - dwadditional | DWORD | 131076 | Additional number des...ng detai/location wi... |
| - dwErrorCode | DWORD | 196867 | Actual error code |
| - wSizeExtDiag | WORD | 44 | Number of bytes of extended diagnosis data |
| + pExtDiagData | POINTER TO BYTE | 16=0079EA40 | internal reference ne... for acknowledgement |
| + hSource | POINTER TO BYTE | $16=0077$ ABE8 | internal reference needed for text convertion |
| + pConn | POINTER TO IoStan... | 16=8573DDD8 | internal reference needed for text convertion |

Analyze the data in the following order: Element "dwAdditional" for the type of diagnosis, element "Error Code", element "SubSysteminfo".

1. Analyze element "dwAdditional", column "Additional" in error lists.

Convert the given value from decimal to hexadecimal format.
"dwAdditional" = $131076=16 \# 20004$
2. Interpretation:

Word 2 = 2
Word $1=4$
3. Generate error text Add_Word 1_Word 2 = Add_4_2
4. Look up which error type it is.

Add_4_2 = extended channel diagnosis

## Data analysis

1. Analyze element "Error Code":
"dwErrorCode" = $196867=16 \# 30103 \rightarrow$
Word 2 (extended error type) $=16 \# 0003$
Word 1 (error type) =16\#0103
$\rightarrow$ Error text: Err_Word 1 (hex)_Word 2 (hex)
$\rightarrow$ Err_0103_0003 $\rightarrow$
Error ID 3 - discrepancy time expired (class 3, error ID 3)
2. Analyze "SubSysteminfo"
"dwSubSysteminfo" $=65540=16 \# 10004 \rightarrow$
Word 2 (subslot index) = 1
Word $1($ channel index) $=4$

| Erro <br> r <br> seve <br> rity | SubSysteminfo |  | Additional |  | Error code |  | Meaning | $\begin{array}{\|l} \hline \text { Rem } \\ \text { edy } \end{array}$ |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | Word 2 <br> (bit 16..31) | Word 1 <br> (bit 0..15) | Word 2 <br> (bit 16..31) <br> ADD_SUB _TYPE | Word 1 <br> (bit 0..15) <br> ADD_TYP <br> E | Word 2 (bit16...31) | Word 1 (bit0...15) |  |  |
| 3 | Subslot index (0 16\#9FFF) | Channel index (016\#7FFF) | 2 (USI: 16\#8002) extended channel diagnosis | 4 (diagnosis alarm) | Extended error type | Error type | Subslot <subslot idx>, channel <channel idx>, extended channel diagnosis; <error text> |  |
| 3 | 1 | 4 | 2 | 4 | 16\#0003 | 16\#0103 | Subslot 1, channel 4, extended channel diagnosis; <error text> |  |
|  | dwSubSysteminfo |  | Add_Word 1_Word 2 |  | dwErrorCode |  | Entries in Diag values |  |
| 3 | $\begin{aligned} & \text { Sub2 = } \\ & \text { Subslot } \end{aligned}$ | Sub1_= Channel | Add_4_2 = extended channel diagnosis |  | Err_Word 1_Word 2 Err_0103_0003 = error ID $\overline{3}$ : Discrepancy time expired (class 3, error ID 3) |  | Text search criteria (ID) in text list Diag_PNIO_26_22 <br> Text from Automation Builder error list (default) in text list Diag_PNIO_26_22 |  |

## 7 Glossary

AC500
AC500-eCo
AC500-S
AC500-S Programming Tool
AC500-SXC

AC500 V2
AC500 V3
AC500-XC
Alarm

## Automation Builder

Cold start

## Download

Compact PLCs
Safety PLCs
IEC 61131-3 editor, integrated in engineering suite Automation Builder

Safety PLCs suitable for extreme environmental conditions

Range of AC500 and AC500-eCo CPUs
Range of AC500 and AC500-eCo CPUs
Standard PLCs suitable for extreme environmental conditions
Diagnosis message which must be acknowledged by the user. The alarm will continue to be displayed even if the issue has been resolved in the meantime until it is acknowledged.

Engineering suite for configuration and programming of all PLCs

Note: The AC500-eCo V3 does not use a battery for buffering the operand areas specified below, hence the "cold start" mode does not exist in this product.

- A cold start is performed by switching power OFF/ON if no battery is connected.
- All RAM memory modules are checked and erased $\Leftrightarrow$ Chapter 6.4.1.21.3.7.10 "Command 'Reset Cold'" on page 2633.
- If no user program is stored in the Flash EPROM, the default values (as set on delivery) are applied to the interfaces.
- If there is a user program stored in the Flash EPROM, it is loaded into RAM.
- The default operating modes set by the PLC configuration are applied.

Control panels for application visualizations

- Data buffering, i.e., maintaining data after power ON/OFF, is only possible, if a battery is connected for AC500 CPU and the buffering will take place in FLASH with AC500eCo V3 CPU. The following data can be buffered completely or in parts:
- Data in the addressable flag area (\%M area)
- RETAIN variable
- PERSISTENT variable (number is limited, no structured variables)
- PERSISTENT area (\%R area)
- In order to buffer particular data, the data must be excluded from the initialization process $\stackrel{y}{ } \stackrel{y}{c}$ Chapter 6.8.2.1 "Handling of remanent variables for AC500 V3 products" on page 4412.
- Download means loading the complete user program into the PLC's RAM. This process is started by selecting the menu item "Online/Download" in the programming system or after confirming a corresponding system message when switching to online mode (menu item "Online/Login").
- Execution of the user program is stopped.
- In order to store the user program to the Flash memory, the menu item "Online/Create boot project" must be called after downloading the program.
- Variables are set to their initialization values according to the initialization table.
- RETAIN variables can have wrong values as they can be allocated to other memory addresses in the new project!
- A download is forced by the following:
- changed PLC configuration
- changed task configuration
- changed library management
- changed compile-specific settings (segment sizes)
- execution of the commands "Project/Clean all" and "Project/Rebuild All".
Event $\quad$ Diagnosis message which does not require acknowledgement. The event describes the
HMI Human machine interface

Online change

Panel
Builder PB610

- After a project has changed, only these changes are compiled when pressing the key $<$ F11> or calling the menu item "Project/Build". The changed program parts are marked with a blue arrow in the block list.
- The term Online Change means loading the changes made in the user program into the PLC's RAM using the programming system (after confirming a corresponding system message when switching to online mode, menu item "Online/Login").
- Execution of the user program is not stopped. After downloading the program changes, the program is re-organized. During re-organization, no further online change command is allowed. The storage of the user program to the Flash memory using the command "Online/Create boot project" cannot be initiated until re-organization is completed.
- Online Change is not possible after:
- changes in the PLC configuration
- changes in the task configuration
- changes in the library management
- changed compile-specific settings (segment sizes)
- performing the commands "Project/Clean all" and "Project/Rebuild All".

Engineering tool for control panels, integrated in engineering suite Automation Builder

- Performs a START -> STOP process.
- Preparation for program restart, i.e., the variables (VAR) (exception: RETAIN variables) are set to their initialization values.
- Reset is performed using the menu item "Online/Reset" in the programming system or pressing the function key RUN for $\geq 5 \mathrm{~s}$ in STOP mode.
- Performs a START -> STOP process.
- Preparation for program restart, i.e., the variables (VAR) (also RETAIN variables) are set to their initialization values.
- Reset (cold) is performed using the menu item "Online/Reset (cold)" in the programming system.
- Resets the controller to its original state (deletion of Flash, SRAM (\%M, area, \%R area, RETAIN, RETAIN PERSISTENT), Communication Module configurations and user program!).
- Reset (original) is performed using the menu item "Online/Reset (original)" in the programming system.
- RUN -> STOP means pressing the RUN function key on the PLC while the PLC is in run mode (AC500 PLC display "run", AC500-eCo PLC "RUN LED" is ON).
- If a user program is loaded into RAM, execution is stopped.
- All outputs are set to FALSE or 0 .
- Variables keep their current values, i.e., they are not initialized.
- The AC500 PLC display changes from "run" to "StoP", AC500-eCo "RUN LED" changes from ON to OFF.

START -> STOP

STOP -> RUN

STOP START

Warm start

- START -> STOP means stopping the execution of the user program in the PLC's RAM using the menu item "Online/Stop" in the programming system.
- All outputs are set to FALSE or 0 .
- Variables keep their current values, i.e., they are not initialized.
- The AC500 PLC display changes from "run" to "StoP".
- STOP -> RUN means short pressing the RUN function key on the PLC while the PLC is in STOP mode (AC500 PLC display "StoP", AC500-eCo "RUN LED" is ON). "RUN LED" is OFF of the toggle switch of an AC500-eCo CPU.
- If a user program is loaded into RAM, execution is continued, i.e., variables will not be set to their initialization values.
- The AC500 PLC display changes from "StoP" to "run", AC500-eCo "RUN LED" changes from OFF to ON.
- STOP -> START means continuing the execution of the user program in the PLC's RAM using the menu item "Online/Start" in the programming system.
- If a user program is loaded into RAM, execution is continued, i.e., variables will not be set to their initialization values.
- The AC500 PLC display changes from "StoP" to "run", AC500-eCo PLC "RUN LED" changes from OFF to ON.
- A warm start is performed by switching power OFF/ON with a battery connected.
- All RAM memory modules are checked and erased except of the buffered operand areas and the RETAIN variables ${ }^{\circ}>$ Chapter 6.4.1.21.3.7.11 "Command 'Reset Warm"' on page 2634.
- If there is a user program stored in the Flash EPROM, it is loaded into RAM.
- The default operating modes set by the PLC configuration are applied.
8 Index
1, 2, 3
__AdrInst ..... 2335
BitOffset ..... 2335
_Cast ..... 2335
_CATCH ..... 2206
__COMPARE_AND_SWAP ..... 2212
__Copy ..... 2335
CRC ..... 2335
__CURRENTTASK ..... 2211
_DELETE ..... 2198
ENDTRY' ..... 2206
FCall ..... 2335
__FINALLY ..... 2206
__Init ..... 2335
__ISVALIDREF ..... 2201, 2245
__Lazy ..... 2335
_Localoffset ..... 2335
__MaxOffset ..... 2335
NEW ..... 2201
_POOL ..... 2217
__POSITION ..... 2214
_POUNAME ..... 2214
_Propertylnfo ..... 2335
__QUERYINTERFACE ..... 2204
__QUERYPOINTER ..... 2205
__RefAdr ..... 2335
__RELOC ..... 2335
__System.ExceptionCode, data type ..... 2206
__SystemScope ..... 2335
__TEST_AND_SET ..... 2215
__TRY ..... 2206
__TypeOf ..... 2335
_ UXINT ..... 2243
convert ..... 2159
__UXINT_TO__XINT ..... 2159
__UXINT_TO__XWORD ..... 2159
__UXINT_TO_BIT ..... 2159
_UXINT_TO_BOOL ..... 2159
__UXINT_TO_BYTE ..... 2159
_UXINT_TO_DATE ..... 2159
UXINT_TO_DINT ..... 2159
__UXINT_TO_DT ..... 2159
_UXINT_TO_DWORD ..... 2159
UXINT_TO_INT ..... 2159
UXINT_TO_LDATE ..... 2159
UXINT_TO_LDT ..... 2159
UXINT_TO_LINT ..... 2159
_UXINT_TO_LREAL ..... 2159
UXINT_TO_LTIME ..... 2159
UXINT_TO_LTOD ..... 2159
UXINT_TO_LWORD ..... 2159
UXINT_TO_REAL ..... 2159
UXINT_TO_SINT ..... 2159
UXINT_TO_STRING ..... 2159
UXINT_TO_TIME ..... 2159
UXINT_TO_TOD ..... 2159
UXINT_TO_UDINT ..... 2159
UXINT_TO_UINT ..... 2159
UXINT_TO_ULINT ..... 2159
UXINT_TO_USINT ..... 2159
UXINT_TO_WORD ..... 2159
UXINT_TO_WSTRING ..... 2159
Varlnfo ..... 2335
VCADD ..... 2255
vector operator ..... 2255
VCDIV ..... 2256
vector operator ..... 2256
VCDOT ..... 2257
vector operator ..... 2257
__VCLOAD_LREAL ..... 2259
vector operator ..... 2259
__VCLOAD_REAL ..... 2259
vector operator ..... 2259
__VCMAX ..... 2258
vector operator ..... 2258
__VCmin ..... 2258
vector operator ..... 2258
__VCMUL ..... 2256
vector operator ..... 2256
VCSET_LREAL ..... 2259
vector operator ..... 2259
_VCSET_REAL ..... 2258
vector operator ..... 2258
VCSQRT ..... 2257
vector operator ..... 2257
VCSTORE ..... 2260
vector operator ..... 2260
VCSUB ..... 2255
vector operator ..... 2255
VECTOR ..... 2253
Wait ..... 2335
XADD ..... 2213
XINT ..... 2243
convert ..... 2159
XINT_TO___UXINT ..... 2159
XINT_TO__XWORD ..... 2159
XINT_TO_BIT ..... 2159
XINT_TO_BOOL ..... 2159
XINT_TO_BYTE ..... 2159
XINT_TO_DATE ..... 2159
XINT_TO_DINT ..... 2159
XINT_TO_DT ..... 2159
XINT_TO_DWORD ..... 2159
XINT_TO_INT ..... 2159
XINT_TO_LDATE ..... 2159
XINT_TO_LDT ..... 2159
XINT_TO_LINT ..... 2159
XINT_TO_LREAL ..... 2159
XINT_TO_LTIME ..... 2159
XINT_TO_LWORD ..... 2159
XINT_TO_REAL ..... 2159
XINT_TO_SINT ..... 2159
XINT_TO_STRING ..... 2159
XINT_TO_TIME ..... 2159
XINT_TO_TOD ..... 2159
XINT_TO_UDINT ..... 2159
XINT_TO_UINT ..... 2159
XINT_TO_ULINT ..... 2159
XINT_TO_USINT ..... 2159
XINT_TO_WORD ..... 2159
XINT_TO_WSTRING ..... 2159
XWORD ..... 2243
convert ..... 2159
XWORD_TO__UXINT ..... 2159
XWORD_TO__XINT ..... 2159
XWORD_TO_BIT ..... 2159
XWORD_TO_BOOL ..... 2159
_ XWORD_TO_BYTE ..... 2159
XWORD TO DATE ..... 2159
XWORD_TO_DINT ..... 2159
XWORD_TO_DT ..... 2159
XWORD_TO_DWORD ..... 2159
XWORD_TO_INT ..... 2159
XWORD_TO_LDATE ..... 2159
XWORD_TO_LDT ..... 2159
XWORD_TO_LINT ..... 2159
XWORD_TO_LREAL ..... 2159
XWORD_TO_LTIME ..... 2159
XWORD_TO_LWORD ..... 2159
XWORD_TO_REAL ..... 2159
XWORD_TO_SINT ..... 2159
XWORD_TO_STRING ..... 2159
XWORD_TO_TIME ..... 2159
XWORD_TO_TOD ..... 2159
XWORD_TO_UDINT ..... 2159
XWORD_TO_UINT ..... 2159
XWORD_TO_ULINT ..... 2159
XWORD_TO_USINT ..... 2159
_XWORD_TO_WORD ..... 2159
__XWORD_TO_WSTRING ..... 2159
_cnt.library ..... 2035
_Itfs.library ..... 2035
//---timestampingserverurl, command line ..... 2034
//--compare, command line ..... 2029
//-culture, command line ..... 2028
//--enforcesignedcompiledlibraries, command line ..... 2033
//--profile, command line ..... 2028
//--project, command line ..... 2029
//--projectarchive, command line ..... 2029
//--runscript, command line ..... 2030
//--signaturethumbprint, command line ..... 2033
//--skipunlicensedplugins, command line ..... 2033
\% ..... 3329
format definition $\% \mathrm{c}$, \%s for text variables ..... 3330
format definition in output text, visualization ..... 3329
\%M ..... 4417
3rd party device diagnosis ..... 4606
21 CFR part 11 standard ..... 4398
A
AB (Automation Builder)
description95
ABS ..... 2194
AC500
connection and wiring ..... 137
control cabinet assembly ..... 104
description ..... 89
system structure ..... 86
AC500 V2/AC500 V3 ..... 6
AC500_Diag ..... 4621, 4631
AC500_DiagTypes ..... 4615, 4616
AC500-eCo
connection and wiring ..... 144
control cabinet assembly ..... 104
description ..... 93
mounting and demounting the option boards ..... 114, 127
system structure ..... 87
AC500-Safety ..... 94, 174
AC522 ..... 672
Analog I/O module ..... 672
Analog input/output module ..... 672
access
symbol set ..... 1941
access control properties ..... 2757
access protection
development system ..... 2041
general information ..... 2039
runtime system / PLC ..... 2041
user management ..... 2039
WebVisu ..... 2042
access right
device editor ..... 2453
object ..... 1822
visualization element ..... 3366
accessor ..... 2493
access method ..... 2493
access method, interface ..... 2490
accessories
AC500 ..... 126
AC500-eCo ..... 127
accessory ..... 1286
Ack ..... 4621, 4631
ACOS ..... 2198
ACS drives ..... 3781
action ..... 2075, 2497
SFC, do not display embedded objects ..... 2684
SFC, duplication mode ..... 2678
SFC, remove duplication ..... 2683
SFC, set duplication ..... 2683
ACTION ..... 2335
action association
insert, command ..... 2680
activate keyboard usage
command, visualization ..... 3343
ActiveX ..... 3258, 3683
visualization element ..... 3258, 3683
add
EN/ENO ..... 2686
file as text ..... 2566
input pin ..... 2695
language in a text list ..... 2728
output pin ..... 2695
ADD ..... 2133
add all instance paths ..... 2720
add file as text ..... 2566
add folder ..... 2599
add POU
dialog ..... 2477
add user
device user management ..... 2450
additional device-specific diagnosis ..... 4606
address ..... 2231
absolute ..... 1939
assign variable ..... 1902
broadcast ..... 1941
I/O ..... 1841
network ..... 1939
node ..... 1939
relative ..... 1939
specification ..... 2231
addressing ..... 1937
relative ..... 1937
ADR ..... 2150, 2243
AI523 ..... 698
Analog input module ..... 698
Al531 ..... 725
Analog input module ..... 725
Al561 ..... 606
AI562 ..... 621
Al563 ..... 632
Al581-S 94, 174
alarm ..... 4605
visualization ..... 2909
alarm acknowledgement ..... 3365
alarm banner ..... 3175, 3600
visualization element 3175, 3600
Alarm Class
Object ..... 2412
Alarm Configuration
Object ..... 2412
Alarm Group
Object ..... 2412
Alarm Storage
Object ..... 2412
alarm table ..... 3166, 3591
visualization element ..... 3166, 3591
alias ..... 2245, 2267
data type ..... 2267
Identifier ..... 2245
object DUT ..... 2461
alignment
command, visualization editor ..... 3344
alpha channel ..... 2914
alternative ..... 2679
analog I/O module ..... 606
analysis ..... 4228
attribute ..... 4228
pragma ..... 4228
analysis, attribute ..... 4229
analysis, pragma ..... 4228
analysis:report-multiple-instance-calls, attribute pragma ..... 4231
analyzation
library ..... 1884
analyzation library ..... 2072
SFC ..... 2072
AND ..... 2139
AND_THEN ..... 2140
AND, pragma ..... 2326
ANDN ..... 2087
animation
visualization element ..... 2912
ANY ..... 2239
data type ..... 2239
ANY BIT ..... 2239
ANY_DATE ..... 2239
ANY_INT ..... 2239
ANY_NUM ..... 2239
ANY_REAL ..... 2239
ANY_STRING ..... 2239
AO523 ..... 764
AO561 ..... 644
application ..... 2410
activate on toolbar ..... 2623
backup application files ..... 2435
build ..... 2619, 2627
build, options ..... 2759
clean ..... 2618
cold start ..... 2633
compare ..... 2626
compile ..... 2627
contents ..... 2626
delete ..... 2636
download ..... 2026, 2628
download with file ..... 2436
encrypt ..... 1820, 1915
encrypted transfer ..... 4207
encryption ..... 2754
encryption, instructions ..... 1830
information ..... 2626
memory mapping ..... 2411
online change ..... 2629
rebuild ..... 2619
remove from PLC ..... 2627
reset ..... 1990
reset (cold) ..... 2633
reset (origin) ..... 2635
reset (warm) ..... 2634
reset origin ..... 2635
set active ..... 2603
warm start ..... 2634
application code
code generation ..... 1976
messages for code generation ..... 1976
application composer
modules view ..... 2583
Application libraries ..... 3833
application update
control panel ..... 148
PLC ..... 148
applications, device editor ..... 2434
arccosine ..... 2198
archive
save ..... 2557
send ..... 2557
arcsine ..... 2197
arctangent ..... 2198
array ..... 2247
access ..... 2228
declaration ..... 2247
declare ..... 1853
initialization ..... 2247
monitor ..... 1998
monitoring ..... 2048
of arrays ..... 2251
of function blocks ..... 2247
of variable length ..... 2252
structure ..... 2247
visualizing ..... 2917
ARRAY ..... 2247
OF ..... 2247
ascii ..... 1590
ASIN ..... 2197
assemblies
EtherNet/IP adapter ..... 2828
assembly ..... 106
CM ..... 109
communication module ..... 109
CPU ..... 109, 114
FM ..... 107
function module terminal base ..... 107
I/O modules ..... 113, 123
option boards (AC500-eCo) ..... 114, 127
processor module ..... 109, 114
TA5301-CFA ..... 123
terminal base ..... 107
terminal block for onboard I/O (AC500-eCo) ..... 116
terminal unit ..... 111
assembly and disassembly ..... 106
assignment ..... 2054
FBD/LD/IL ..... 2092
output ST ..... 2052
ST operator ..... 2051
association
insert for SFC action, command ..... 2680, 2681
AT ..... 1902
declaration ..... 1902
ATAN ..... 2198
attribute
for pragma ..... 2271
authentication
SVN ..... 4352
automatic formatting ..... 2581
ST code ..... 2581
Automation Builder
description ..... 95
device state diagnosis ..... 4611
diagnosis description ..... 4611, 4615
getting started ..... 16
installation ..... 16
licensing ..... 16
AX521 ..... 780
AX522 ..... 810
AX561 ..... 655
B
B ..... 2231
size prefix ..... 2231
background
designing the visualization ..... 2884
placing, visualization ..... 3349
backup ..... 2024, 2435
create backup file ..... 2435
backup and restore ..... 2024, 2435
backup file ..... 2435
create, save ..... 2435
BACnet ..... 1734, 3833
BACnet Building Controller ..... 1736, 3835
BACnet configuration ..... 1738, 3838
BACnet libraries ..... 1748, 3847
BACnet network ..... 1736, 3835
BACnet server root object ..... 1739, 3838
BACnet-BC System ..... 1733, 3833
bar display ..... 3181, 3606
visualization element ..... 3181, 3606
Basic CPU ..... 176
battery ..... 4445
Bézier curve ..... 3012, 3437
visualization element ..... 3012, 3437
BIBBs and services ..... 1738, 3837
binary
display mode when monitoring ..... 2654
number ..... 2220
binary number
format definition \%b ..... 3329
BIT ..... 2243
convert ..... 2159
structure ..... 2262
bit access ..... 2229
in integer variable ..... 2229
in structure variable ..... 2229
BIT TO UXINT ..... 2159
BIT_TO___XINT ..... 2159
BIT_TO__XWORD ..... 2159
BIT_TO_BOOL ..... 2159
BIT_TO_BYTE ..... 2159
BIT_TO_DATE ..... 2159
BIT_TO_DINT ..... 2159
BIT_TO_DT ..... 2159
BIT_TO_DWORD ..... 2159
BIT_TO_INT ..... 2159
BIT_TO_LDATE ..... 2159
BIT_TO_LDT ..... 2159
BIT_TO_LINT ..... 2159
BIT_TO_LREAL ..... 2159
BIT_TO_LTIME ..... 2159
BIT_TO_LTOD ..... 2159
BIT_TO_LWORD ..... 2159
BIT_TO_REAL ..... 2159
BIT_TO_SINT ..... 2159
BIT_TO_STRING ..... 2159
BIT_TO_TIME ..... 2159
BIT_TO_TOD ..... 2159
BIT_TO_UDINT ..... 2159
BIT_TO_UINT ..... 2159
BIT_TO_ULINT ..... 2159
BIT_TO_USINT ..... 2159
BIT_TO_WORD ..... 2159
BIT_TO_WSTRING ..... 2159
BITADR ..... 2151
bitmap, properties ..... 2758
block diagramsgrounding concept131
bookmarks
clear all ..... 2571
clear all in active editor ..... 2571
next ..... 2570
next in active editor ..... 2570
previous ..... 2570
previous in active editor ..... 2570
set ..... 1908
toggle ..... 2569
view ..... 2585
BOOL ..... 2235
constant ..... 2220
convert ..... 2154
data type ..... 2235
BOOL_TO___UXINT ..... 2154
BOOL_TO___XINT ..... 2154
BOOL_TO__XWORD ..... 2154
BOOL_TO_BIT ..... 2154
BOOL_TO_BYTE ..... 2154
BOOL_TO_DATE ..... 2154
BOOL_TO_DINT ..... 2154
BOOL_TO_DT ..... 2154
BOOL_TO_DWORD ..... 2154
BOOL_TO_INT ..... 2154
BOOL_TO_LDATE ..... 2154
BOOL_TO_LDT ..... 2154
BOOL TO LINT ..... 2154
BOOL_TO_LREA ..... 2154
BOOL_TO_LTIME ..... 2154
BOOL_TO_LTOD ..... 2154
BOOL_TO_LWORD ..... 2154
BOOL_TO_REAL ..... 2154
BOOL_TO_SINT ..... 2154
BOOL_TO_STRING ..... 2154
BOOL_TO_TIME ..... 2154
BOOL_TO_TOD ..... 2154
BOOL_TO_UDINT ..... 2154
BOOL_TO_UINT ..... 2154
BOOL_TO_ULINT ..... 2154
BOOL_TO_USINT ..... 2154
BOOL_TO_WORD ..... 2154
BOOL_TO_WSTRING ..... 2154
boot application browse ..... 1908delete2636
encrypt ..... 1820, 4202
encrypted transfer ..... 4207
encryption ..... 1915
generate ..... 1978, 2627
options ..... 2759
properties ..... 2754
signing 1915, 1917
Boot Support Package ..... 4397
box
CFC ..... 2110
clean, FBD/LD ..... 2710
FBD/LD/IL ..... 2092
FBD/LD/IL, empty box with en/eno ..... 2702
insert parallel LD ..... 2702
insert, FBD/LD/IL ..... 2701
repair ..... 2710
with EN/ENO, FBD/LD/IL ..... 2702
box input
insert, FBD/LD/IL ..... 2703
bracket, code ..... 2568
branch ..... 2078
add ..... 2679
closed ..... 2096
insert right ..... 2679
set end point ..... 2710
start/end ..... 2095
branch/tag
create ..... 4347
breakpoint ..... 1981
concept ..... 1981
condition ..... 2750
define condition ..... 1983
define execution point ..... 1984
disable ..... 2646
edit ..... 2645
enable ..... 2646
execution point ..... 2751
new ..... 2645
position ..... 2752
set ..... 1983
toggle ..... 2646
view ..... 2586
broadcast address ..... 1941
BSP ..... 4397
build
application ..... 2619
exclude ..... 2755
messages for code generation ..... 1976
properties ..... 2755
build information ..... 2618
bulk data manager ..... 3862
bus cycle
EtherCAT ..... 1627
J1939 ..... 1603
PROFINET IO ..... 1642
bus cycle task
device editor ..... 2447
busy symbol ..... 3266, 3691
cube, visualization element ..... 3266, 3691
flower, visualization element ..... 3270, 3695
button ..... 3089, 3514
visualization element ..... 3089, 3514
BY ..... 2056
byte
addressing mode ..... 2231
BYTE ..... 2235
convert ..... 2159
byte order ..... 2321
BYTE_TO___UXINT ..... 2159
BYTE_TO___XINT ..... 2159
BYTE TO XWORD ..... 2159
BYTE_TO_BIT ..... 2159
BYTE_TO_BOOL ..... 2159
BYTE_TO_DATE ..... 2159
BYTE_TO_DINT ..... 2159
BYTE_TO_DT ..... 2159
BYTE_TO_DWORD ..... 2159
BYTE_TO_INT ..... 2159
BYTE_TO_LDATE ..... 2159
BYTE_TO_LDT ..... 2159
BYTE_TO_LINT ..... 2159
BYTE_TO_LREAL ..... 2159
BYTE_TO_LTIME ..... 2159
BYTE_TO_LTOD ..... 2159
BYTE_TO_LWORD ..... 2159
BYTE_TO_REAL ..... 2159
BYTE_TO_SINT ..... 2159
BYTE TO STRING ..... 2159
C0042 ..... 2355
BYTE_TO_TOD ..... 2159
BYTE_TO_UDINT ..... 2159
C0044 ..... 2356C00432355
BYTE TO UINT ..... 2159
C0045 ..... 2356
BYTE_TO_ULINT ..... 2159
C0046 ..... 2356
BYTE_TO_USINT ..... 2159
C0047 ..... 2357
BYTE_TO_WORD ..... 2159C00482357
BYTE_TO_WSTRING ..... 2159
BYTEBIT_TO_TIME ..... 2159
CC integration
IDE, path ..... 2756
properties ..... 2756
C stub file ..... 2619
C stub file for external library ..... 2619
C0001 ..... 2344
C0002 ..... 2344
C0004 ..... 2345
C0005 ..... 2345
C0006 ..... 2346
C0007 ..... 2346
C0008 ..... 2346
C0009 ..... 2347
C0010 ..... 2347
C0011 ..... 2347
C0013 ..... 2348
C0015 ..... 2348
C0018 ..... 2348
C0020 ..... 2349
C0022 ..... 2349
C0023 ..... 2349
C0026 ..... 2350
C0027 ..... 2350
C0030 ..... 2350
C0031 ..... 2351
C0032 ..... 2351
C0033 ..... 2351
C0035 ..... 2352
C0036 ..... 2352
C0037 ..... 2352
C0038 ..... 2353
C0039 ..... 2353
C0040 ..... 2354
C0041 ..... 2354
C0049 ..... 2358
C0050 ..... 2358
C0051 ..... 2358
C0053 ..... 2359
C0061 ..... 2359
C0062 ..... 2359
C0064 ..... 2360
C0065 ..... 2360
C0066 ..... 2360
C0068 ..... 2361
C0069 ..... 2361
C0070 ..... 2362
C0072 ..... 2362
C0074 ..... 2362
C0075 ..... 2363
C0076 ..... 2363
C0077 ..... 2363
C0078 ..... 2364
C0080 ..... 2364
C0081 ..... 2365
C0082 ..... 2365
C0084 ..... 2365
C0085 ..... 2366
C0086 ..... 2366
C0087 ..... 2367
C0089 ..... 2367
C0090 ..... 2368
C0091 ..... 2368
C0094 ..... 2368
C0096 ..... 2369
C0097 ..... 2369
C0098 ..... 2370
C0099 ..... 2370
C0101 ..... 2371
C0102 ..... 2371
C0104 ..... 2371
C0114 ..... 2371
C0115 ..... 2372
C0116 ..... 2372
C0197 ..... 2389
C0117 2372 C0198 ..... 2389
C0118 2372 C0199 ..... 2389
C0119 2373 C0201 ..... 2390
C0120 ..... 2373
C0203 ..... 2390
C0122 ..... 2374
C0204 ..... 2391
C0124 ..... 2374
C0205 ..... 2391
C0125 ..... 2374
C0206 ..... 2391
C0126 ..... 2375
C0207 ..... 2391
C0130 ..... 2375
C0208 ..... 2392
C0131 ..... 2376
C0209 ..... 2392
C0132 ..... 2376
C0211 ..... 2392
C0136 ..... 2376
C0212 ..... 2393
C0138 ..... 2377
C0215 ..... 2393
C0139 ..... 2377
C0216 ..... 2393
C0140 2377 C0217 ..... 2393
C0141 ..... 2378
C0218 ..... 2394
C0142 2378 C0219 ..... 2394
C0143 ..... 2378
C0221 ..... 2395
C0144 ..... 2379
C0222 ..... 2395
C0145 2379 C0224 ..... 2395
C0149 ..... 2380
C0225 ..... 2396
C0161 2380 C0227 ..... 2396
C0162 ..... 2380
C0228 ..... 2397
C0164 ..... 2381
C0230 ..... 2397
C0165 ..... 2381
C0232 ..... 2397
C0168 2382 C0233 ..... 2398
C0169 2382 C0234 ..... 2398
C0173 ..... 2383
C0235 ..... 2399
C0174 2383 C0236 ..... 2399
C0175 ..... 2383
C0237 ..... 2399
C0177 ..... 2384
C0238 ..... 2400
C0178 ..... 2384
C0239 ..... 2400
C0179 385 C0240 ..... 2401
C0180 ..... 2385C02412401, 2404
C0182 ..... 2385
C0242 ..... 2401
C0183 ..... 2386
C0243 ..... 2402
C0185 ..... 2386
C0380 ..... 2402
C0186 2386 C0509 ..... 2403
C0188 ..... 2387
C0542 ..... 2404
C0189 2387 C0543 ..... 2405
C0190 ..... 2388
C0549 ..... 2405
C0191 ..... 2388
C0550 ..... 2406
C0195 2388 C0554 ..... 2406
C01962388
C0555 ..... 2407
CA-signed certificate . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2044
CA-signed certificate ..... 1830
CAA library ..... 3849
CAA SerialCom ..... 1590
CAL ..... 2152
CALC ..... 2087
CALCN ..... 2087
call
function ..... 2482
function block, ST ..... 2060
program ..... 2478
call stack ..... 1994
view ..... 2590
call tree
show ..... 2572
view ..... 2590
call_after_global_init_slot ..... 2273
pragma attribute ..... 2273
call after init ..... 2273
pragma attribute ..... 2273
call_after_online_change_slot ..... 2274
pragma attribute ..... 2274
call_before_global_exit_slot
pragma attribute ..... 2275
call_on_type_change ..... 2275
pragma attribute ..... 2275
calls
function block ..... 2479
cam
object properties ..... 2763
CANbus
J1939 ..... 1603
parameters ..... 2433
CANopen ..... 336, 953
local device ..... 1602
modular device ..... 1597
non-modular device ..... 1597
remote device ..... 1597
Cartesian XY chart ..... 3296
visualization element ..... 3296
CASE ..... 2057
CATCH ..... 2206
category
configure in visualization toolbox ..... 3368
create for visualization elements ..... 2873
CD522 ..... 907
central I/O extension ..... 86, 88
centralized I/O expansion ..... 86, 88
centralized I/O extension ..... 86
centralized IO extension ..... 88
certificate ..... 1820
application ..... 1915
boot application, download, online change ..... 4202
CA-signed ..... 1820, 1830, 2044
controller ..... 4201
delete ..... 1830
encrypted communication ..... 1915, 1967
encryption ..... 1820
encryption, instructions ..... 1830
expiration date ..... 1967
expired ..... 2043
general information ..... 2040
issue more ..... 2043
project settings ..... 2772
request from PLC ..... 1830
Security Agent ..... 4201
sign boot application ..... 1917
signing ..... 1820
time stamp by command line ..... 2034
via PLC shell ..... 2044
Windows Certificate Store ..... 1820
CFC ..... 1866
add input ..... 2695
add output ..... 2695
connect structure ..... 2698
connection mark ..... 2696
create control point ..... 2696
edit page size ..... 2686
edit parameters ..... 2692
edit worksheet ..... 2685
editor ..... 2098
end with selected elements ..... 2689
force FB ..... 2697
group ..... 2696
keyboard shortcuts ..... 2102
move down ..... 2690
move up ..... 2689
order by data flow ..... 2691
order by topology ..... 2691
page-oriented ..... 1867, 2098
CheckDivLReal' ..... 2507
parameter values ..... 2693
programming in CFC editor ..... 1871
properties ..... 2761
reference ..... 2687
remove control point ..... 2695
remove unused pins ..... 2694
Reset ..... 2687
reset connecting line ..... 2695
reset pins ..... 2694
route connections ..... 2695
select connected pins ..... 2694
send to front ..... 2688
Set ..... 2687
set element number within execution order ..... 2690
show next collision ..... 2694
starting point in feedback network ..... 1869
ungroup ..... 2697
unlock connection ..... 2693
CFC editor ..... 2098
breakpoint ..... 2103
debugging ..... 2103
monitoring ..... 2103
online mode ..... 2103
option ..... 2784
page-oriented ..... 2101
setting ..... 2784
toolbar ..... 2048
Change over to another module type ..... 1414, 1475
change the language
font settings per language ..... 2909
character string
with placeholder, visualization ..... 3329
character string literal ..... 2221
chart ..... 2804, 3296
in trace editor ..... 2804
visualization element ..... 3296
check
avoid implicit checks ..... 2298
check box ..... 3156, 3577
visualization element ..... 3156, 3577
CheckBounds ..... 2502
array ..... 2247
CheckDivInt ..... 2505
CheckDivLInt' ..... 2505
CheckDivReal ..... 2506
CheckLRangeSigned ..... 2510
CheckLRangeUnsigned ..... 2512
CheckPointer ..... 2513
CheckRangeSigned ..... 2268, 2508
CheckRangeUnsigned ..... 2268, 2511
checks_in_libs ..... 2500
chromium browser ..... 4397
Cl ..... 92
CI501 ..... 1203
CI501-PNIO ..... 1203
CI502 ..... 1244
CI502-PNIO ..... 1244
CI511 ..... 1018
CI512 ..... 1051
CI512-ETHCAT ..... 1051
CI521 ..... 1070
CI521-MODTCP ..... 1070
CI522 ..... 1111
CI522-MODTCP ..... 1111
CI541 ..... 1136
CI541-DP ..... 1136
CI542 ..... 1176
CI542-DP ..... 1176
CI581 ..... 953
CI581-CN ..... 953
CI582 ..... 993
clean all ..... 2618
clean gaps ..... 2719
clock
visualization element ..... 3317, 3737
cloned code ..... 4216
closed branch, LD ..... 2096
CloseDialog ..... 3337
CloseDialog2 ..... 3337
CM ..... 89, 327
CM579 ..... 344, 365
CM579- EtherCAT master ..... 344
CM579-ETHCAT
diagnosis ..... 4665
CM579-PNIO ..... 365
CM582 ..... 354
CM589 ..... 371
CM589-PNIO ..... 371
CM589-PNIO-4 ..... 371
CM589-PNIO(-4) ..... 371
CM592 ..... 358
CM592- PROFIBUS DP Master ..... 358
CM598 ..... 336
CM5610-2RS ..... 331
CM5610-2RS-XC ..... 331
CM5640-2ETH ..... 348
CM5640-2ETH-XC ..... 348
cmp libraries
app library ..... 3773
iec task library ..... 3773
log library ..... 3773
CmpTraceMgr
runtime system component ..... 2008
CmpTraceMgr.library ..... 2007
code
analysis, pragmas ..... 4228
analyze ..... 1904
checks ..... 4218
collapse all ..... 2568
duplicate ..... 4216
encrypt ..... 1820
encryption ..... 1977
exclude from the static analysis ..... 1905
expand all ..... 2568
format code ..... 2581
generate ..... 2618
generate, active application ..... 2621
go to matching bracket ..... 2568
code analysis
getting started ..... 4209
metrics ..... 4226
naming conventions ..... 4219
prohibited symbols ..... 4227
rules ..... 4218
code check
metrics ..... 4226
prohibited symbols ..... 4227
rules ..... 4218
settings ..... 4217
code clone ..... 4216
code duplicate ..... 4216
code generation ..... 2618
application code ..... 1976
application in library project ..... 2621
messages ..... 1976
coding guidelines ..... 1904
CoE online ..... 1622
coil ..... 2095
insert ..... 2704
reset ..... 2704
set ..... 2704
cold start ..... 2633
collapse all ..... 2568
color ..... 2914
code as hexadecimal number ..... 2914
visualization ..... 2876
color animation
configure for visualization element ..... 2878, 2916
color definition ..... 2914
byte order ..... 2914
color gradient
specify for visualization element ..... 2877
color space ..... 2914
combo box, array ..... 3079, 3497
visualization element ..... 3079, 3497
combo box, integer ..... 3072, 3503
visualization element ..... 3072, 3503
command 'Go to instance' ..... 2577
command icon ..... 2801
customize ..... 1805
command line ..... 2028
---timestampingserverurl ..... 2034
--compare ..... 2029
--culture ..... 2028
--enforcesignedcompiledlibraries ..... 2033
--ignorecomments ..... 2032
--ignoreproperties ..... 2033
--ignorewhitespace ..... 2032
--profile ..... 2028
--project ..... 2029
--projectarchive ..... 2029
--runscript ..... 2030
--signaturethumbprint ..... 2033
--skipunlicensedplugins ..... 2033
comment
CFC ..... 2111
library documentation ..... 2061
ST ..... 2061
comment out ..... 2569
commit
ignore object, SVN ..... 4335
commit accepted changes ..... 2611
communication
controller, encrypted ..... 4201
device editor ..... 2427
edit ..... 1957
enable unencrypted ..... 2046
encrypt ..... 1820
encrypted ..... 1967
encryption, certificate ..... 4201
setting, classic display ..... 2786
Communication
Modbus TCP/IP ..... 4523
communication gateway ..... 42, 2817
communication interface module ..... 951
CANopen ..... 953
EtherCAT ..... 1018
Modbus ..... 1070
PROFIBUS ..... 1136
PROFINET ..... 1201
communication module ..... 327
CANopen ..... 336
EtherCAT ..... 343
Ethernet ..... 348
mounting and demounting ..... 109
PROFIBUS ..... 354
PROFINET ..... 365
serial ..... 331
communication parameters in Windows 19, 42, 1382, 2817
communication policy ..... 2427
compare ..... 4331
projects 1405, 1817, 2607
with HEAD revision, SVN ..... 4331
with revision, SVN ..... 4331
working copy and base revision, SVN ..... 4331
working copy and project in SVN repository ..... 4332
compare objects command ..... 2607
compare view ..... 1405, 1817
detail ..... 1405, 1817
open detailed ..... 1407, 1818
project ..... 1405, 1817
COMPARE AND SWAP ..... 2212
Comparison
AC500 V3 terminal bases ..... 303
comparison view ..... 1406, 1818
compatibility
library2622
compiled libraries signature ..... 2033
compiled library
save ..... 2556
compiled library, see compiled library ..... 2517
compiled-library ..... 2035
compiled-library-v3 ..... 2035
compiler
options ..... 2769
version ..... 2778
warnings ..... 2770
compiler version
project environment ..... 2778
Compiling a project ..... $42,63,67,80$
components
pumping library ..... 4059
composer
CFC ..... 2111
compression
project ..... 2791
condition
breakpoint ..... 2750
conditional compilation ..... 2318
conditionalshow, pragma ..... 2276
conditionalshowsymbols, command-line command2277, 2278
condtionalshow_all_locals pragma pragma ..... 2277
configuration ..... 145
device ..... 1836
programming system ..... 2745
task ..... 2538
trace ..... 2804
configuration tool for MODBUS ..... 1565
configuration variable ..... 2121
connect
to device ..... 2640
connection
CFC, connect pins ..... 2693
CFC, route ..... 2695
connection mark ..... 2696
Connection
AC522 ..... 674
connection and wiring ..... 137
AC500-eCo ..... 144
connection mark
CFC ..... 2112
connections
EtherNet/IP adapter ..... 2826
Connections
AC522 ..... 674
const_non_replaced, pragma ..... 2278
const replaced, pragma ..... 2278
constant ..... 2219
BOOL ..... 2220
date ..... 2225
input/output variable ..... 2117
LTIME ..... 2224
REAL and LREAL ..... 2221
string ..... 2221
time ..... 2223
TIME ..... 2223
time of day ..... 2225
variable ..... 2121
CONSTANT ..... 2219
variable ..... 2121
constants
numeric ..... 2220
contact ..... 2094
insert ..... 2707
Id ..... 2704
negated ..... 2706
negated, parallel ..... 2706
paste right after ..... 2707
right ..... 2705
Content of the memory card for application update1432
Content of the memory card for firmware update 1432
Content of the memory card for firmware/applica- tion update ..... 1432
Content of the SD card for firmware/application update ..... 1432
content operator ..... 2151
CONTINUE ..... 2060
Continuous Function Chart ..... 1866
page-oriented ..... 1867, 2098
Continuous Function Chart (CFC) ..... 1866
Continuous Function Chart (CFC) - page-oriented ..... 1867, 2098
control cabinet ..... 134
control cabinet assembly ..... 104
control elements ..... 148, 1338
control panel ..... 3282, 3707, 4397
visualization element ..... 3282, 3707
control panels ..... 95
control panels CP600 ..... 130, 145
control point
CFC ..... 2109
create ..... 2696
remove ..... 2695
control variable
IEC 61850 server ..... 1679
controller
communication, certificate-encrypted ..... 1820
security ..... 2041
symbol access ..... 2458
unencrypted communication ..... 2046
wink ..... 2640
conversion ..... 1394, 1799
convert
device ..... 2747
integer ..... 2159
library reference ..... 2746
strings ..... 2174
to FBD ..... 2711
to IL ..... 2711
to LD ..... 2711
TO ..... 2153
TRUNC ..... 2193
TRUNC_INT ..... 2193
convert V2 project to V3 project ..... 1394, 1799
copied code ..... 4216
core dump
close ..... 2654
create ..... 2653
load ..... 2653
load device log ..... 2654
open memory view ..... 2592
COS ..... 2196
coupler ..... 91
mounting and demounting ..... 109
cp 2nd generation ..... 157, 4398
CP-C. 1 ..... 1327
CP6xx ..... 4397
CP6xx-eCo ..... 4397
CP6xx-Pro ..... 4397
CP64xx ..... 4397
CP66xx ..... 4397
CP600 ..... 95, 130, 145
converting a project ..... 157
runtime ..... 157
CPU
mounting and demounting ..... 109, 114
CPU display
diagnosis description ..... 4607
CPU load ..... 2014, 2537
CPU selection ..... 97
cpuload trace ..... 2008, 2740
CpuLoad
DeviceTrace ..... 2015
create device list CSV
command ..... 2665
create localization template ..... 2605
cross reference browse ..... 2571
global ..... 2571
cross-reference ..... 2590
auto-update ..... 2796
IEC address ..... 2786
cross-reference list ..... 2587
classic view ..... 2590
collapse all ..... 2568
limit ..... 2744
occurrence location ..... 1906
view ..... 2587
CurrentVisu
Variable for visualization name ..... 3398
custom data type ..... 2461
customize ..... 2800
cyber security ..... 1340
cyber security requirements ..... 1349
cycle consistency ..... 1855


## D


#### Abstract

D 2231


keyword ..... 2225
size prefix ..... 2231
DA501 ..... 840
DA502 ..... 874
Digital/Analog input/output module ..... 874
dar file ..... 2453
data
exchange ..... 1936
record and trace ..... 2007
data breakpoint ..... 1981
condition ..... 2750
execution point ..... 2751
data logger ..... 3849
data logger library ..... 3849
data logging ..... 3849
data persistence ..... 1920
data record (see sample) ..... 2007
data security ..... 1340, 1971
data set
IEC 61850 server ..... 1688
data source ..... 1947, 2414
add ..... 2413
add initially ..... 1949
choose variables ..... 2416
communication via address monitoring ..... 2422
communication via symbols ..... 2417
object ..... 2414
OPC UA Client ..... 1960, 2424
symbolic access ..... 2418
type ..... 1947
type mapping ..... 2416
update rate ..... 2426
variables ..... 2415
data source manager general ..... 1947
Data Source Manager
object ..... 2412
data source type
ApplicationV3 ..... 1948
symbolic ..... 1947
data type
__System.ExceptionCode ..... 2207
alias ..... 2267
ANY ..... 2239
BIT ..... 2243
date and time ..... 2238
enumeration ..... 2263
integer ..... 2235
LTIME ..... 2237
message ..... 2304
overflow underflow ..... 2129
reference ..... 2245
standard data types ..... 2234
structure ..... 2261
UNION ..... 2268
user defined ..... 2263
WSTRING ..... 2242
data unit type ..... 2461
dataflow, pragma attribute ..... 2279
dataset
IEC 61850 server ..... 1688
date ..... 2238
constant ..... 2225
data type ..... 2238
DATE ..... 2238
convert ..... 2187
data type ..... 2238
keyword ..... 2225
date picker ..... 3311, 3730
visualization element ..... 3311, 3730
date range picker ..... 3301, 3721
visualization element ..... 3301, 3721
DATE_AND_TIME ..... 2238
data type ..... 2238
keyword ..... 2225
DATE_TO___UXINT ..... 2187
DATE_TO__ XINT ..... 2187
DATE_TO__XWORD ..... 2187
DATE_TO_BOOL ..... 2187
DATE_TO_BYTE ..... 2187
DATE_TO_DINT ..... 2187
DATE_TO_DT ..... 2187
DATE_TO_DWORD ..... 2187
DATE_TO_INT ..... 2187
DATE_TO_LDATE ..... 2187
DATE_TO_LDT ..... 2187
DATE_TO_LINT ..... 2187
DATE TO LREAL ..... 2187
DATE_TO_LTOD ..... 2187
DATE_TO_LWORD ..... 2187
DATE_TO_REAL ..... 2187
DATE_TO_SINT ..... 2187
DATE_TO_STRING ..... 2187
DATE_TO_TIME ..... 2187
DATE_TO_TOD ..... 2187
DATE_TO_UDINT ..... 2187
DATE_TO_UINT ..... 2187
DATE_TO_ULINT ..... 2187
DATE_TO_USINT ..... 2187
DATE_TO_WORD ..... 2187
DATE_TO_WSTRING ..... 2187
date/time formats ..... 3331
format definition \%t ..... 3331
date/time picker ..... 3324, 3744
visualization element ..... 3324, 3744
DC522 ..... 517
DC523 ..... 529
Digital input/output module ..... 529
DC532 ..... 541
Digital input/output module ..... 541
DC562 ..... 390
DCF file ..... 2663
DCS drives ..... 3782
debug ..... 1985
CFC editor ..... 2103
flow control ..... 1992
operating mode ..... 2642
run to cursor ..... 2648
set next statement ..... 2648
show next statement ..... 2648
step out ..... 2647
using step into ..... 2647
using step over ..... 2646
debug mode ..... 1985
Debugging ..... 4209
decentral I/O extension ..... 87, 88
decentralized I/O expansion ..... 87, 88
decentralized I/O extension ..... 87
decentralized IO extension ..... 87, 88
decimal
display mode when monitoring ..... 2654
number ..... 2220
decimal number
format definition \%d, \%i . . . . . . . . . . . . . . . . . 3329
declaration ..... 4228
attribute, static analysis ..... 4228
Auto Declare ..... 1886
change order ..... 1912
global variable ..... 1854
go to ..... 1908
refactoring ..... 1912
declaration editor ..... 1851, 2047
edit declaration header ..... 2717
option ..... 2785
show/hide ..... 2672
tabular/textual ..... 1851
declare ..... 1847
array ..... 1853
automatic ..... 2796
short form feature ..... 1887
task-local global variable list ..... 1855
variable, command ..... 2572
Decommissioning ..... 157
default keyboard shortcuts
visualization manager ..... 3402
define, pragma ..... 2318
Definitions: PLC system start-up ..... 4420
DELETE ..... 2198
delete IL line ..... 2707
demounting ..... 106
CM ..... 109
communication module ..... 109
FM ..... 107
function module terminal base ..... 107
I/O modules ..... 113, 123
option boards (AC500-eCo) ..... 114, 127
processor module ..... 109, 114
terminal base ..... 107
terminal block for onboard I/O (AC500-eCo) ..... 116
terminal unit ..... 111
demounting a processor module on DIN rail ..... 120
on metal plate ..... 122
dereferencing ..... 2243
description
AC500 ..... 89
S500 ..... 89
design attribute ..... 2303
devdesc.xml ..... 2663
development system
appearance and behavior ..... 1802
Development System
Features ..... 1800
device ..... 2427
add ..... 2599
application ..... 2434
configuration ..... 2427
configuration mode ..... 2616
configure ..... 1836
connect ..... 2427
connection ..... 2640
conversion ..... 2747
database ..... 2663
devices view ..... 2582
disable ..... 2614
encrypted communication ..... 1967, 2427, 4201
favorite ..... 2427
files ..... 2437
function block instance ..... 2449
I/O mapping ..... 2444
IEC objects ..... 2449
insert ..... 2614
install ..... 2038, 2663
interactive login ..... 2765
$\log$ ..... 2437
map I/Os ..... 1837
online config mode ..... 2616
options ..... 2765
PLC settings ..... 2439
PLC shell ..... 2022
plug ..... 2600
properties ..... 2765
scan ..... 2427
scan hardware 1607, 1628, 2600 ..... 2832
security ..... 1967
send echo service ..... 2427
simulation mode ..... 1980, 2640
symbol access ..... 2458
uninstall ..... 2663
update ..... 2602
user management ..... 1971, 2450, 2453
wink ..... 2427
device description download ..... 2663
download, option ..... 2785
install ..... 2663
device diagnosis 2812, 4613, 4628, 4648
device ECAD data
command ..... 2665
device editor ..... 2447
access rights ..... 2453
add user ..... 2451
applications ..... 2434
backup and restore ..... 2435
communication ..... 2427
communication settings ..... 2427
encrypted communication ..... 2427
EtherCAT ..... 1610
files ..... 2027, 2437
generic ..... 2427
I/O mapping ..... 2444
IEC objects ..... 2449
information ..... 2460
KNX ..... 1730
$\log$ ..... 2437
options ..... 2786
parameters ..... 2433
PLC settings ..... 2439
PLC shell ..... 2441
status ..... 2460
symbol rights ..... 2458
synchronized file ..... 2436
task deployment ..... 2459
users and groups ..... 2450
device permission management file ..... 2451, 2454
dm ..... 2451, 2454
device reader ..... 2668
device repository ..... 2663
renew ..... 2663
device state 4606, 4611, 4619, 4629
device useradd2637
change password ..... 2639
logout current user ..... 2637
remove ..... 2638
device user management ..... 1971
device user management file 2451, 2454
dum2, dum ..... 2451, 2454
device version ..... 2779
DEVICE_STATE ..... 4629
devices view ..... 2582
DeviceTrace ..... 2007
CPU load ..... 2014
download ..... 2740
object ..... 2426
DI524 ..... 553
Digital input module ..... 553
DI561 ..... 402
Digital input module ..... 402
DI562 ..... 409
DI571 ..... 418
Digital input module ..... 418
DI572 ..... 428
Digital input module ..... 428
DI581-S ..... 94, 174
Diag (function block) ..... 4621, 4631
DIAG_TXT_TYPE ..... 4617
DIAG_VAL_TYPE ..... 4616
Diag.NumClass ..... 4620
Diag.NumTotal ..... 4620
DiagAck ..... 4621, 4631
DiagGet... (method) ..... 4621, 4631
diagnosis ..... 4605
acknowledge ..... 2602
device diagnosis ..... 4613, 4648
fieldbus ..... 2812
PLC shell ..... 2022
subtree ..... 2602
system diagnosis ..... 4613, 4619
Diagnosis
AC522 ..... 689
VisuDrvModbusRTUBroadcast ..... 3822
diagnosis description 4606, 4607, 4611, 4615
diagnosis messageV34605, 4606
diagnosis messages list ..... 4655
CM579-ETHCAT ..... 4665
CM579-PNIO ..... 4704
CM582-DP ..... 4699
CM589-PNIO ..... 4688
CM598-CN ..... 4689
diagnosis system ..... 4605
diagnostic messages list CM579-ETHCAT ..... 4665
CM579-PNIO ..... 4704
CM589-PNIO ..... 4688
CM598-CN ..... 4689
diagnostic system ..... 4605
diagnostics history ..... 1621
diagram
autofit Y -trace axis ..... 2733
mouse zooming ..... 2737
DiagValToTxt ..... 4627, 4637
dialog ..... 2957, 2962
calling in a visualization ..... 2957
close, input action ..... 3371
implement with interface ..... 2962
open, input action ..... 3371
opening globally in a visualization ..... 2959
user management for visualization ..... 3400
dialog manager ..... 3335
methods ..... 3335
Differentiation AC500 V2/AC500 V3 ..... 6
digital I/O module ..... 390
digital/analog I/O module ..... 840
DINT ..... 2235
convert ..... 2159
DINT_TO___UXINT ..... 2159
DINT_TO__ XINT ..... 2159
DINT_TO__XWORD ..... 2159
DINT_TO_BIT ..... 2159
DINT_TO_BOOL ..... 2159
DINT_TO_BYTE ..... 2159
DINT_TO_DATE ..... 2159
DINT_TO_DT ..... 2159
DINT_TO_DWORD ..... 2159
DINT_TO_INT ..... 2159
DINT_TO_LDATE ..... 2159
DINT_TO_LDT ..... 2159
DINT_TO_LINT ..... 2159
DINT_TO_LREAL ..... 2159
DINT_TO_LTIME ..... 2159
DINT_TO_LTOD ..... 2159
DINT_TO_LWORD ..... 2159
DINT_TO_REAL ..... 2159
DINT_TO_SINT ..... 2159
DINT TO STRING ..... 2159
DINT_TO_TIME ..... 2159
DINT_TO_TOD ..... 2159
DINT_TO_UDINT ..... 2159
DINT_TO_UINT ..... 2159
DINT_TO_ULINT ..... 2159
DINT_TO_USINT ..... 2159
DINT_TO_WORD ..... 2159
DINT_TO_WSTRING ..... 2159
dip switch ..... 3231, 3656
visualization element ..... 3231, 3656
disable breakpoint ..... 2646
disassembly ..... 106
CM ..... 109
communication module ..... 109
CPU ..... 109, 114
FM ..... 107
function module terminal base ..... 107
I/O modules ..... 113, 123
option boards (AC500-eCo) ..... 114, 127
processor module ..... 109, 114
terminal base ..... 107
terminal block for onboard I/O (AC500-eCo) ..... 116
terminal unit ..... 111
disassembly file ..... 2622
disconnect
device ..... 2640
display ..... 148, 1338
display variant ..... 2973
configure ..... 2973
executing as integrated ..... 2976
executing webvisu ..... 2974
maximum number ..... 3402
size of the paintbuffer ..... 3401
TargetVisu ..... 3408
WebVisu ..... 3409
displaymode, pragma ..... 2280
DIV ..... 2136
division by zero ..... 2505, 2506, 2507
dm file
device permission management file ..... 2451, 2454
DO ..... 2056
DO524 ..... 562
Digital output module ..... 562
DO526 ..... 571
DO561 ..... 437
Digital output module ..... 437
DO562 ..... 447
Digital output module ..... 447
DO571 ..... 457
Digital output module ..... 457
DO572 ..... 468
Digital output module ..... 468
DO573 ..... 478
Digital output module ..... 478
document ..... 2606
dongle ..... 1825, 2659
encryption ..... 1915
project settings ..... 2772
dot product ..... 2257
downloadcommand2637
device description from the server ..... 2785
encrypt ..... 4202
encrypt code ..... 1915
library from server ..... 2790
multiple ..... 2631
source code, project setting ..... 2770
trace ..... 2734
user data, visualization ..... 3406
download manager
command ..... 2631
download to controller
multiple ..... 2631
Drive parameter settings ..... 3811
DT ..... 2238
convert ..... 2187
data type ..... 2238
keyword ..... 2225
DT_TO__XWORD ..... 2187
DT_TO__UXINT ..... 2187
DT_TO__XINT ..... 2187
DT_TO_BOOL ..... 2187
DT_TO_BYTE ..... 2187
DT_TO_DATE ..... 2187
DT_TO_DINT ..... 2187
DT_TO_DWORD ..... 2187
DT_TO_INT ..... 2187
DT_TO_LDATE ..... 2187
DT_TO_LDT ..... 2187
DT_TO_LINT ..... 2187
DT_TO_LREAL ..... 2187
DT_TO_LTOD ..... 2187
DT_TO_LWORD ..... 2187
DT_TO_REAL ..... 2187
DT_TO_SINT ..... 2187
DT_TO_STRING ..... 2187
DT_TO_TIME ..... 2187
DT_TO_TOD ..... 2187
DT_TO_UDINT ..... 2187
DT_TO_UINT ..... 2187
DT_TO_ULINT ..... 2187
DT_TO_USINT ..... 2187
DT_TO_WORD ..... 2187
DT_TO_WSTRING ..... 2187
dum file ..... 2450
dum2 file, dum file
device user management file ..... 2451, 2454
duplicated code ..... 4216
duplication ..... 2678
remove ..... 2683
SFC ..... 2678
SFC, set ..... 2683
duplication mode ..... 2678
DUT ..... 2461
add ..... 2462
DWORD ..... 2235
convert ..... 2159
DWORD_TO__UXINT ..... 2159
DWORD_TO___XINT ..... 2159
DWORD_TO__XWORD ..... 2159
DWORD_TO_BIT ..... 2159
DWORD_TO_BOOL ..... 2159
DWORD_TO_BYTE ..... 2159
DWORD_TO_DATE ..... 2159
DWORD_TO_DINT ..... 2159
DWORD_TO_DT ..... 2159
DWORD_TO_INT ..... 2159
DWORD_TO_LDATE ..... 2159
DWORD_TO_LDT ..... 2159
DWORD_TO_LINT ..... 2159
DWORD_TO_LREAL ..... 2159
DWORD_TO_LTIME ..... 2159
DWORD_TO_LTOD ..... 2159
DWORD_TO_LWORD ..... 2159
DWORD_TO_REAL ..... 2159
DWORD_TO_SINT ..... 2159
DWORD_TO_STRING ..... 2159
DWORD_TO_TIME ..... 2159
DWORD_TO_TOD ..... 2159
DWORD_TO_UDINT ..... 2159
DWORD_TO_UINT ..... 2159
DWORD_TO_ULINT ..... 2159
DWORD_TO_USINT ..... 2159
DWORD_TO_WORD ..... 2159
DWORD_TO_WSTRING ..... 2159
DX522 ..... 581
Digital input/output module ..... 581
DX531 ..... 594
Digital input/output module ..... 594
DX561 ..... 491
DX571 ..... 503
DX581-S ..... 94, 174
dynamic memory allocation ..... 2201
Eearthing conceptblock diagrams131
echo service ..... 2427
Edge browser ..... 53, 57, 2855
edit code
format document ..... 2581
editing mode graphical editor ..... 2048
editor
close all ..... 2669
close all (inactive applications) ..... 2670
close all other ..... 2673
next ..... 2669
previous ..... 2669
visualization ..... 3393
Electrical Connection
AC522 ..... 674
element
selection, tab order ..... 3342
element list
command, visualization ..... 3342
tab, visualization ..... 3342
element properties ..... 2584
SFC ..... 2080
element property
visualization ..... 3396
elements for alarms acknowledgement ..... 3365
elimination of security risks ..... 1345
ellipse ..... 2988, 3413
visualization element ..... 2988, 3413
ELSE ..... 2055
ELSIF ..... 2055
EMC-conforming mounting and construction ..... 132
empty box
insert ..... 2702
empty box with eno ..... 2702
EN/ENO
add ..... 2686
FBD/LD/IL ..... 2092
enable
breakpoint ..... 2646
enable_dynamic_creation, pragma ..... 2281
encoder, counter and PWM module ..... 907
encrypted communication
data source OPC UA Client ..... 1960
device editor ..... 2427
encryption ..... 2039
application ..... 1915
boot application ..... 4207
boot application, download, online change, cer- tificate ..... 4202
certificate ..... 1820
certificate, controller ..... 4201
communication with PLC ..... 1967
dialog, security screen ..... 2592
dongle ..... 1915
download code ..... 1915
method ..... 2039
project ..... 1825
properties, application ..... 2754
Security Agent ..... 4201
signature ..... 2039
with certificate, instructions ..... 1830
wizard, boot application ..... 4207
encryption wizard ..... 4207
boot application, download, online change ..... 4202
END_ACTION ..... 2335
END_CASE ..... 2057
END_FOR ..... 2056
END_FUNCTION ..... 2335
END_FUNCTION_BLOCK ..... 2335
END_IF ..... 2055
END_PROGRAM ..... 2335
END_REPEAT ..... 2058
END_STRUCT ..... 2261
END_TYPE ..... 2263
END_UNION ..... 2268
END_VAR ..... 2113
END_WHILE ..... 2058
endianess ..... 2321
ENDTRY' ..... 2206
engineering software ..... 95
enlarging/reducing a pin group ..... 2302
entry action ..... 2076
SFC, add ..... 2678
enumeration ..... 2263
conversion TO_STRING ..... 2314
data type ..... 2263
default value ..... 2263
initialization ..... 2263
namespace ..... 2217
object DUT ..... 2461
pragma strict ..... 2265
EQ ..... 2149
ERROR ..... 4630
error list ..... 4655
CM579-PNIO ..... 4704
CM589-PNIO ..... 4688
CM598-CN ..... 4689
Error list
CM579-ETHCAT ..... 4665
error message
V3 ..... 4605, 4606
ERROR_ID ..... 4617
estimated-stack-usage, pragma ..... 2281
EtherCAT 343, 1018, 1610, ..... 436
bus cycle ..... 1627
generate xml file ..... 2614
I/O mapping ..... 1610
IEC objects ..... 1610
information ..... 1610
parameters ..... 1610
status ..... 1610
EtherCAT Master
general ..... 1610
parameters ..... 1613
EtherCAT Master/Slave ..... 1622
file transfer ..... 1621
EtherCAT module
startup parameters ..... 1624
EtherCAT Slave
CoE online ..... 1622
diagnostics history ..... 1621
Ethernet connection ..... 1623
expert mode process data ..... 1617
FMMU/sync ..... 1616
general ..... 1613
IDN ..... 1619
online ..... 1621
parameters ..... 1623
process data, inputs/outputs ..... 1619
SDO ..... 1619
startup parameters ..... 1619
Ethernet ..... 348
EtherCAT connection ..... 1623
Ethernet communication interface modules ..... 1270
Ethernet over EtherCAT ..... 1623
Ethernet protocols and ports for AC500 V3 prod- ucts ..... 307, 4480
EtherNet/IP
Net $X$ configuration ..... 2824
EtherNet/IP adapter
assemblies ..... 2828
connections ..... 2826
general ..... 2825, 2831
new connection ..... 2827
select parameters ..... 2830
user parameters ..... 2829
EtherNet/IP Scanner general ..... 2824
EtherNet/IP scanner NetX general ..... 2825
ETS5 parameters ..... 1732
event ..... 4605
example project with Automation Builder and AC500 AC500 V3 products ..... 24, 71
exception
stop execution ..... 2639
Exception
catch in IEC code ..... 2206
exception error, see exception ..... 2206
exception handling ..... 2206
display variant ..... 2974
exchange localization files project ..... 1833
execute ..... 2094
ST code in FBD/LD/IL ..... 2094
execute command ..... 3373
input action ..... 3373
execution order ..... 1867
by data flow ..... 1867
CFC ..... 1867
end with selected elements ..... 2689
move down ..... 2690
move up ..... 2689
order by data flow ..... 2691
selected elements to front ..... 2688
set number of element ..... 2690
set start of feedback in CFC POU ..... 2688
show tags ..... 2688
execution point ..... 2751
EXIT ..... 2060
exit action ..... 2076
add, SFC ..... 2678
EXP ..... 2195
expand all ..... 2568
expandfully, pragma ..... 2284
expert mode process data
EtherCAT Slave ..... 1617
export
library ..... 2037
library from Library Manager ..... 2716
library from repository ..... 2657
export server
IEC 61850 ..... 1693, 1713
export/import
I/O mapping ..... 2616
PLCopen XML ..... 1815
PLCopenXML ..... 2612
text list ..... 2729
XML ..... 1815
Exportieren ..... 2611
expression, ST ..... 2050
EXPT ..... 2195
operator ..... 2195
ExST ..... 1879
reset ..... 2053
Set ..... 2052
EXT ..... 1885, 2755
extend memory profile ..... 2555
extend profile ..... 2555
extended diagnosis ..... 4606, 4607
Extended Structured Text ..... 1879
EXTENDS ..... 1930
external file ..... 2464
properties ..... 2757
external implementation ..... 1885
configuration ..... 2755
external variable ..... 2120
extract archive ..... 2558
F
FALSE ..... 2220
FAQ ..... 1341
fast counters 210, 539, 910, 1135, 4538
FB_Exit ..... 2336
method ..... 2336
FB_Init ..... 2336
method ..... 2336
FB_Reinit ..... 2336
method ..... 2336
FBD ..... 1860
option ..... 2787
programming in ..... 1862
FBD/LD/IL
insert assignment, ..... 2701
insert box ..... 2701
insert empty box with en/eno ..... 2702
insert input ..... 2703
insert jump ..... 2703
insert jump label ..... 2703
insert network ..... 2700
insert network below ..... 2701
insert return ..... 2703
online operation ..... 2086
view as function block diagram ..... 2711
view as instruction list ..... 2711
view as ladder logic ..... 2711
FBD/LD/IL editor ..... 2082
line branch ..... 2710
toolbar ..... 2048
Features
Development System ..... 1800
fieldbus devices ..... 2812
fieldbus diagnosis ..... 2812
fileadd2464
download during application download ..... 2436
link to object ..... 2763
save ..... 1831
save as ..... 1831
to and from PLC ..... 2027
transfer, input action ..... 3379
file transfer
configure mode ..... 2978
EtherCAT Master/Slave ..... 1621
visualization - PLC ..... 3379
File transfer
on controller and visualization ..... 3401
files
device editor ..... 2437
filing
project ..... 1822
FINAL ..... 2477
FINALLY ..... 2206
find ..... 2563
find next (selection) ..... 2565
find previous (selection) ..... 2566
in help ..... 2674
next ..... 2565
previous ..... 2565
firewall protection ..... 1341
firmware update
control panel ..... 148
PLC ..... 148
Firmware updatewith IP configuration tool1463, 1513
flag memory ..... 2231
flash ..... 2411
external memory ..... 2411
flexconf ..... 1443
flexible configuration ..... 1443
floating-point number ..... 2236
constant ..... 2221
format definition \%f, \%e ..... 3330
flow control ..... 1992, 2652
flow control mode ..... 4076
FM ..... 92
FMMU/sync
EtherCAT Slave ..... 1616
font
visualization manager ..... 3407
visualization, language ..... 2909
FOR ..... 2056
force ..... 1987
force values ..... 2649
handling in watch list ..... 2584
in CFC ..... 2697
watch all forces ..... 1989
forcing
add all forces to watchlist ..... 2585
prepare value ..... 2749
format definition
in output text, visualization ..... 3329
format document ..... 2581
format code ..... 2581
frame ..... 3053, 3478
selection, visualization command ..... 3348
switch visualization, input action ..... 3377
switch visualizations in a frame via follow-up actions ..... 2945
switch visualizations in a frame with a variable2941
update parameters, visualization ..... 3367
visualization element ..... 3053, 3478
Frame
select visualizations ..... 3348
frame visualization ..... 2941
FTP server ..... 1721
full screen mode ..... 2597
function ..... 2482
call ..... 2482
call via event ..... 2533
call with external implementation ..... 1885
monitor ..... 2001
reaction to type change ..... 2275
FUNCTION ..... 2482
function as operand ..... 2233
function block ..... 2479, 3773
add input, CFC ..... 2112
add output, CFC ..... 2112
assignment, info ..... 2293
call ..... 2479
call with external implementation ..... 1885
call, ST ..... 2060
extend ..... 1930
I/O channel ..... 1840
I/O channel, mapping ..... 2449
I/O mapping ..... 1840, 2293, 2444
implement interface ..... 1932
initialization on call ..... 2290
map I/O channel ..... 2444
monitor ..... 1998
monitor with properties ..... 2000
property ..... 2493
select for I/O mapping ..... 2746
test, reflection ..... 2313
function block diagram ..... 1860
function extraction ..... 4215
function module ..... 907
function module terminal base
mounting and demounting ..... 107, 111
FUNCTION_BLOCK ..... 2479
functionality
AC522 ..... 673
G
gateway
add ..... 2427
block driver ..... 2721
configuration ..... 2721
configuration file ..... 2721
manage ..... 2427
gateway.cfg ..... 2721
GE ..... 2149
general
EtherCAT Master ..... 1610
EtherCAT Slave ..... 1613
EtherNet/IP adapter ..... 2825, 2831
EtherNet/IP Scanner ..... 2824
EtherNet/IP scanner NetX ..... 2825
KNX ..... 1730, 1731, 1732
generate codeIEC 61850 server1712
gesture
for operating a visualization ..... 2887

get
access method, interface ..... 2490
Get
accessor method ..... 2493
Get... (method) ..... 4621, 4631
GetClientInterface ..... 3336
GetDialog ..... 3336
getting started
display histogram ..... 2862
trend visualization ..... 2928
global namespace operator ..... 2216
global network variable list ..... 1946
global text list
add language and translate text ..... 1891
check ..... 1894
compare and export differences ..... 1893
create ..... 2728
create again with current IDs ..... 1895
enter text in visualization element ..... 1894
export ..... 1891
for static application ..... 1894
import file ..... 1892
object ..... 2465
remove text list entries ..... 1895
update ID ..... 1895
update with replacement file ..... 1895
global variable ..... 2118
declare ..... 1854
global variable list ..... 2465
declare task-local ..... 1855
task-local ..... 2466
global variables list namespace ..... 2216
global_init_slot, pragma ..... 2285
GloballmagePool ..... 1899
go to
definition, how to ..... 1908
line ..... 2567
matching bracket ..... 2568
network ..... 2712
go to definition command ..... 2576
go to source position ..... 2583
GOOSE publisher
IEC 61850 server ..... 1691
GOOSE subscriber
IEC 61850 server ..... 1694
gradient editor
visualization ..... 3369
graphical editor toolbar ..... 2048
grid
visualization ..... 3385
grounding concept
block diagrams ..... 131
group ..... 2450
CFC, create ..... 2696
CFC, remove ..... 2697
in the visualization editor ..... 3347
user management ..... 1821
user management, visualization ..... 3403
group box ..... 3101, 3526
visualization element ..... 3101, 3526
grouping
ungrouping, visualization editor ..... 3348
GSD file ..... 2663
GT ..... 2148
GVL ..... 2465
declare task-local ..... 1855
namespace ..... 2216
property ..... 2493
task-local ..... 2466
H
HA-Modbus TCP ..... 4535
System Technology ..... 3859
hardware protection ..... 1345
hasattribute, pragma ..... 2318
hasconstantvalue, pragma ..... 2318
hastype, pragma ..... 2318
help
language ..... 2790
offline help ..... 2789
online help ..... 2789
hexadecimal
display mode when monitoring ..... 2654
number ..... 2220
hexadecimal number
format definition \%x, \%llx ..... 3329
hide windows ..... 1807
hide_all_locals, pragma ..... 2289
hide, pragma ..... 2286
High Availability system ..... 3859
High performance range ..... 1327
histogram ..... 3216, 3641
configure ..... 2862
visualization element ..... 3216, 3641
HMI ..... 95
hot swap behaviour ..... 157
HTTP library ..... 1757, 1798, 3833
human machine interface ..... 95
I

1. ..... 2231
memory range prefix ..... 2231
I/O bus ..... 97
I/O channel
function block ..... 1840
IEC objects ..... 2449
map to function block ..... 2444
map to variable ..... 2444
select function block ..... 2746
I/O configuration
AC522 ..... 684
i/o faq ..... 303
I/O mapping ..... 1405, 1836, 2812
all devices ..... 1843
change address ..... 1841
device editor ..... 2444
edit ..... 2615
EtherCAT ..... 1610
export to CSV ..... 2616
force ..... 1843
function block ..... 1840
import from CSV ..... 2615
KNX ..... 1730
monitoring ..... 1842
procedure ..... 1837
select function block ..... 2746
task deployment ..... 1994
update ..... 1842
I/O module ..... 387
analog ..... 606
digital ..... 390
digital/analog ..... 840
function module ..... 907
general ..... 1644
mounting and demounting ..... 113, 123
I/O selection ..... 97
Identifier
alias ..... 2245
identifiers
rules ..... 2327
search order ..... 2333
IEC 61850
export server ..... 1693, 1713
IEC 61850 server ..... 1671
dataset ..... 1688
generate code ..... 1712
GOOSE publisher ..... 1691
import server ..... 1713
options ..... 1713
properties ..... 1679
report ..... 1689
variable ..... 1679
IEC action
SFC ..... 2075
IEC application
device diagnosis ..... 4628
device state diagnosis ..... 4619, 4629
IEC objectsdevice editor2449
EtherCAT ..... 1610
KNX ..... 1730
IEC task ..... 4425
IF ..... 2055
statement ..... 2055
if, pragma ..... 2318
IL ..... 1861
online operation ..... 2086
option ..... 2787
image ..... 3038, 3463
visualization element ..... 3038, 3463
image file
insert ..... 2717
names and directories for visualization ..... 3385
image pool ..... 1899, 2468
create ..... 1899
download ..... 2764
global ..... 1899
object ..... 2468
properties ..... 2764
image selection ..... 2469
image switcher ..... 3221, 3646
visualization element ..... 3221, 3646
implement interfaces ..... 2744
IMPLEMENTS ..... 1932
import
I/O mapping ..... 2615
PLCopenXML ..... 2612
project in SVN ..... 4326
Import ..... 2612
import assistant
configuration programming system ..... 2745
import server
IEC 61850 ..... 1713
import users
device user management ..... 2450
index ..... 2674
index access ..... 2244
INDEXOF ..... 2137
information ..... 2675
device editor ..... 2460
EtherCAT ..... 1610
KNX ..... 1730
information model object ..... 2473
information model OPC UA ..... 2473
INI ..... 2218
init step, SFC ..... 2675
init_namespace, pragma attribute ..... 2291
init_on_onlchange, pragma ..... 2291
initialization ..... 1851
array ..... 2247
avoid, pragma ..... 2299
FB on call ..... 2290
input variable, pragma ..... 2290
namespace, pragma ..... 2291
online change, download ..... 2291
order, pragma ..... 2285
vector ..... 2253
with FB_Init, FB_Reinit ..... 2336
initialize on call, pragma ..... 2290
inline monitoring ..... 1996
enable ..... 2569
example ..... 1996
input
CFC ..... 2109
device ..... 1837
input action ..... 3370
visualization ..... 3370
input assistance ..... 1885
Auto Declare ..... 1886
behavior in visualization ..... 3385
input assistant ..... 1886
List components ..... 1886
short form feature ..... 1887
input assistant ..... 2574
categories ..... 2575
dialog ..... 2575
text search ..... 2575
Input Assistant options ..... 2796
input configuration ..... 3370
visualization ..... 3370
input event
visualization element ..... 2886
input memory ..... 2231
input pin order ..... 2303
Input simulator ..... 1305
input variable ..... 2113
refactoring ..... 1911
input/output module
mounting and demounting ..... 113, 123
input/output variable ..... 2114
constant ..... 2117
VAR_IN_OUT ..... 2114
insert ..... 2717
box parallel LD ..... 2702
branch above ..... 2709
branch below ..... 2709
branch, Id ..... 2709
coil ..... 2704
contact ..... 2707
contact parallel above ..... 2705, 2706
contact parallel below ..... 2705
contact right ..... 2705
contact, Id ..... 2704
contact, negated parallel ..... 2706
empty box ..... 2702
image file ..... 2717
instruction line ..... 2707
line branch prallel ..... 2709
negated contact ..... 2706
set coil ..... 2704
text in a text list ..... 2729
variable declaration in the tabular editor ..... 2717
insert action association insert, command ..... 2681
insert assignment, FBD/LD/IL ..... 2701
insert contact
parallel above ..... 2705, 2706
parallel below ..... 2705
insert input
box, FBD/LD/IL ..... 2703
insert jump ..... 2703
insert jump label ..... 2703
insert network ..... 2700
insert network below ..... 2701
install
device ..... 2038, 2663
library ..... 2657
install additional license command ..... 2655
instance path ..... 2720
add in global persistent variable list ..... 2720
add in variables configuration ..... 2720
instance variables ..... 2120
instance-path, pragma ..... 2292
instruction list ..... 1861
INT ..... 2235
convert ..... 2159
INT_TO___UXINT ..... 2159
INT_TO__XINT ..... 2159
INT_TO__XWORD ..... 2159
INT_TO_BIT ..... 2159
INT_TO_BOOL ..... 2159
INT_TO_BYTE ..... 2159
INT_TO_DATE ..... 2159
INT_TO_DINT ..... 2159
INT_TO_DT ..... 2159
INT_TO_DWORD ..... 2159
INT_TO_LDATE ..... 2159
INT_TO_LDT ..... 2159
INT_TO_LINT ..... 2159
INT_TO_LREAL ..... 2159
INT_TO_LTIME ..... 2159
INT_TO_LTOD ..... 2159
INT_TO_LWORD ..... 2159
INT_TO_REAL ..... 2159
INT_TO_SINT ..... 2159
INT_TO_STRING ..... 2159
INT_TO_TIME ..... 2159
INT_TO_TOD ..... 2159
INT_TO_UDINT ..... 2159
INT_TO_UINT ..... 2159
INT_TO_ULINT ..... 2159
INT_TO_USINT ..... 2159
INT_TO_WORD ..... 2159
INT_TO_WSTRING ..... 2159
integer convert ..... 2159
integer data type ..... 2235
Intended purpose
AC522 ..... 673
interface ..... 2484
call visualization with interface ..... 2952
command, visualization ..... 3340
editor, visualization ..... 3340
extend ..... 1933
implement ..... 1932
property ..... 2490
update data source programmatically ..... 1958
update, visualization ..... 3367
INTERFACE ..... 2484
interface method ..... 2490
INTERNAL ..... 2485
method ..... 2485
property ..... 2493
Internal data exchange
AC522 ..... 684
interpretation of CM579-PNIO diagnosis ..... 4704
invisible input ..... 3147, 3572
visualization element ..... 3147, 3572
IO mapping ..... 1405
io modules ..... 94
IO modules ..... 91
io_function_block pragma attribute ..... 2746
io_function_block_mapping ..... 2293
pragma attribute ..... 2293, 2746
IP address
change ..... 1461, 1511
IP configuration tool ..... 1457, 1506
IP-configuration
command ..... 2655
is_connected ..... 2293
pragma attribute ..... 2293
IT security ..... 1341
J
J1939 ..... 1603
bus cycle ..... 1603
jitter ..... 1915
task configuration ..... 1915
JMP ..... 2059, 2087
JMPC ..... 2087
JMPCN ..... 2087
jump ..... 2079
add ..... 2681
CFC ..... 2110
insert after ..... 2681
insert, FBD/LD/IL ..... 2703
jump label
FBD/LD/IL ..... 2093
insert, FBD/LD/IL ..... 2703
ST ..... 2059
K
key
certificate ..... 2040
key combination ..... 1805
key file ..... 1814
keyboard
call for virtual input, visualization ..... 2889
keyboard configuration command, visualization ..... 3341
tab, visualization ..... 3341
keyboard shortcut ..... 2802
customize ..... 1805
keyboard shortcuts ..... 1805
configure for elements ..... 2892
configure on visualization ..... 2893
for default keyboard action, visualization ..... 3338
keypad
visualization ..... 3399
keyword ..... 2335
uppercase ..... 2796
KNX ..... 1730
device editor ..... 1730
ETS5 ..... 1732
general ..... 1730, 1731, 1732
I/O mapping ..... 1730
IEC objects ..... 1730
information ..... 1730
parameters ..... 1730
status ..... 1730
L
label ..... 2111, 3068, 3493
CFC ..... 2111
visualization element ..... 3068, 3493
ladder diagram ..... 1860
lamp ..... 3226, 3651
visualization element ..... 3226, 3651
language
current in the visualization ..... 3399
help ..... 2790
project localization ..... 1833
switch input action ..... 3372
user interface, command line ..... 2028
user interface, options ..... 2790
visualization ..... 2906
language switching
configure, instructions ..... 2906
latency ..... 1915
task configuration ..... 1915
LD 1860 ..... , 2087
closed branch ..... 2096
keyword ..... 2225
online operation ..... 2086
option ..... 2787
programming in ..... 1864
LDATE ..... 2238
convert ..... 2187
data type ..... 2238
keyword ..... 2225
LDATE AND TIME ..... 2238
data type ..... 2238
keyword ..... 2225
LDATE_TO___UXINT ..... 2187
LDATE_TO___XINT ..... 2187
LDATE_TO___XWORD ..... 2187
LDATE_TO_BOOL ..... 2187
LDATE_TO_BYTE ..... 2187
LDATE_TO_DATE ..... 2187
LDATE_TO_DINT ..... 2187
LDATE_TO_DT ..... 2187
LDATE_TO_DWORD ..... 2187
LDATE_TO_INT ..... 2187
LDATE_TO_LDT ..... 2187
LDATE_TO_LINT ..... 2187
LDATE_TO_LREAL ..... 2187
LDATE_TO_LTOD ..... 2187
LDATE_TO_LWORD ..... 2187
LDATE_TO_REAL ..... 2187
LDATE_TO_SINT ..... 2187
LDATE_TO_STRING ..... 2187
LDATE_TO_TIME ..... 2187
LDATE_TO_TOD ..... 2187
LDATE_TO_UDINT ..... 2187
LDATE_TO_UINT ..... 2187
LDATE_TO_ULINT ..... 2187
LDATE_TO_USINT ..... 2187
LDATE_TO_WORD ..... 2187
LDATE_TO_WSTRING ..... 2187
LDN ..... 2087
LDT
convert ..... 2187
keyword ..... 2225
LDT_TO__ XWORD ..... 2187
LDT_TO__UXINT ..... 2187
LDT_TO XINT ..... 2187
LDT_TO_BOOL ..... 2187
LDT_TO_BYTE ..... 2187
LDT_TO_DATE ..... 2187
LDT_TO_DINT ..... 2187
LDT_TO_DT ..... 2187
LDT_TO_DWORD ..... 2187
LDT_TO_INT ..... 2187
LDT_TO_LDATE ..... 2187
LDT_TO_LINT ..... 2187
LDT TO LREAL ..... 2187
LDT_TO_LTOD ..... 2187
LDT_TO_LWORD ..... 2187
LDT_TO_REAL ..... 2187
LDT_TO_SINT ..... 2187
LDT_TO_STRING ..... 2187
LDT_TO_TIME ..... 2187
LDT_TO_TOD ..... 2187
LDT_TO_UDINT ..... 2187
LDT_TO_UINT ..... 2187
LDT_TO_ULINT ..... 2187
LDT_TO_USINT ..... 2187
LDT_TO_WORD ..... 2187
LDT_TO_WSTRING ..... 2187
LE ..... 2148
LEDs ..... 148, 1338
legend ..... 3254, 3679
visualization element 3254, 3679
level control mode ..... 4080
level-control mode ..... 4080
library ..... 3773
add ..... 2712
add to project ..... 2036
check compatibility ..... 2622
checks_in_libs ..... 2500
convert library reference ..... 2746
create ..... 2852
download ..... 2469
download, option ..... 2790
export ..... 2037, 2657, 2716
install ..... 2657
integrate ..... 2469
integrate in project ..... 2471
Library Manager ..... 2469
library types ..... 2035
location ..... 2657
mapping definition ..... 2790
namespace ..... 2217, 2469, 2470
options ..... 2790
outdated versions ..... 2778
placeholder ..... 2716
profile ..... 2657
properties ..... 2469, 2714
protected, signed ..... 2035
referenced libraries ..... 2469
reload ..... 2713
save as compiled library ..... 2556
signing ..... 2556
summary ..... 2469
uninstall ..... 2657
use POUs ..... 1890
library development
information ..... 2035
Library Development Summary ..... 2852
library documentation
comment ..... 2061
library manager ..... 3773
Library Manager ..... 2469
general ..... 2034
library project
category ..... 2515
compiled library ..... 2517
license ..... 1814
licensing ..... 2517
sign ..... 2517
library reference
conversion ..... 2746
library repository ..... 2657
adding a library ..... 2037
general ..... 2034
license
activate ..... 2659
information ..... 2675
manage ..... 2659
plug-in ..... 2675
request ..... 2659
restore ..... 2659
return ..... 2659
start development system without license prompt ..... 2033
license Information
controller ..... 2668
License Manager ..... 2659
license repository ..... 2662
LIMIT ..... 2147
line ..... 3000, 3425
visualization element ..... 3000, 3425
line branch ..... 2710
IL ..... 2093
open ..... 2093
start point ..... 2709
start/end ..... 2095
linkalways ..... 2294
pragma ..... 2294
Lint ..... 1904
programming tool for code analysis ..... 1904
LINT ..... 2235
convert ..... 2159
data type ..... 2235
LINT_tO___UXINT ..... 2159
LINT_tO__XINT ..... 2159
LINT_tO__XWORD ..... 2159
LINT_tO_BIT ..... 2159
LINT_tO_BOOL ..... 2159
LINT_tO_BYTE ..... 2159
LINT_TO_DATE ..... 2159
LINT_TO_DINT ..... 2159
LINT_TO_DT ..... 2159
LINT_TO_DWORD ..... 2159
LINT_TO_INT ..... 2159
LINT_TO_LDATE ..... 2159
LINT_TO_LDT ..... 2159
LINT_TO_LREAL ..... 2159
LINT_TO_LTIME ..... 2159
LINT_TO_LTOD ..... 2159
LINT_TO_LWORD ..... 2159
LINT_TO_REAL ..... 2159
LINT_TO_SINT ..... 2159
LINT_TO_STRING ..... 2159
LINT_TO_TIME ..... 2159
LINT_TO_TOD ..... 2159
LINT_TO_UDINT ..... 2159
LINT_TO_UINT ..... 2159
LINT_TO_ULINT ..... 2159
LINT_TO_USINT ..... 2159
LINT_TO_WORD ..... 2159
LINT_TO_WSTRING ..... 2159
List components ..... 1886
literal ..... 2219
character ..... 2221
date ..... 2225
time of day ..... 2225
typed ..... 2228
LN ..... 2194
load
project, option ..... 2791
loading to the controller ..... 2026
local variable ..... 2113
localization
project ..... 1833
localization template
project localization ..... 1833
lock
get, SVN ..... 4336
steal, SVN ..... 4337
locked, operating mode ..... 2642
log
device ..... 2654
open ..... 2437
PLC ..... 2021
SVN ..... 4337
VendorException ..... 2437
Log ..... 4647
LOG ..... 2195
log of the PLC
device editor ..... 2437
log-in to a CPU ..... 50
login ..... 2624
as user ..... 1828
via user account ..... 1827
with certificate only ..... 1820
wrong password ..... 2045
logout ..... 2627, 2637
LOWER_BOUND ..... 2252
array ..... 2252
lowercase ..... 2567
LREAL ..... 2236
constant ..... 2221
convert ..... 2171
literal ..... 2221
LREAL_TO__XWORD ..... 2171
LREAL_TO__UXINT ..... 2171
LREAL_TO_XINT ..... 2171
LREAL_TO_BIT ..... 2171
LREAL_TO_BOOL ..... 2171
LREAL_TO_BYTE ..... 2171
LREAL_TO_DATE ..... 2171
LREAL_TO_DINT ..... 2171
LREAL_TO_DT ..... 2171
LREAL_TO_DWORD ..... 2171
LTIME_TO_WSTRING ..... 2182
LREAL_TO_INT ..... 2171
LREAL_TO_LINT ..... 2171
LREAL_TO_LREAL ..... 2171
LREAL_TO_LWORD ..... 2171
LREAL_TO_SINT ..... 2171
LREAL_TO_STRING ..... 2171
LREAL_TO_UDINT ..... 2171
LREAL_TO_UINT ..... 2171
LREAL_TO_ULINT ..... 2171
LREAL_TO_USINT ..... 2171
LT ..... 2148
LTIME ..... 2237
constant ..... 2224
convert ..... 2182
literal ..... 2224
LTIME_OF_DAY ..... 2238
data type ..... 2238
keyword ..... 2225
LTIME_TO___UXINT ..... 2182
LTIME_TO___XINT ..... 2182
LTIME_TO__XWORD ..... 2182
LTIME_TO_BOOL ..... 2182
LTIME_TO_BYTE ..... 2182
LTIME_TO_DATE ..... 2182
LTIME_TO_DINT ..... 2182
LTIME_TO_DT ..... 2182
LTIME_TO_DWORD ..... 2182
LTIME_TO_INT ..... 2182
LTIME_TO_LDATE ..... 2182
LTIME_TO_LDT ..... 2182
LTIME_TO_LINT ..... 2182
LTIME_TO_LREAL ..... 2182
LTIME_TO_LTOD ..... 2182
LTIME_TO_LWORD ..... 2182
LTIME_TO_REAL ..... 2182
LTIME_TO_SINT ..... 2182
LTIME_TO_STRING ..... 2182
LTIME_TO_TIME ..... 2182
LTIME_TO_TOD ..... 2182
LTIME_TO_UDINT ..... 2182
LTIME_TO_UINT ..... 2182
LTIME_TO_ULINT ..... 2182
LTIME_TO_USINT ..... 2182
LTIME_TO_WORD ..... 2182
LTOD ..... 2238
convert ..... 2187
data type ..... 2238
keyword ..... 2225
LTOD_TO__XWORD ..... 2187
LTOD_TO__UXINT ..... 2187
LTOD_TO__XINT ..... 2187
LTOD_TO_BOOL ..... 2187
LTOD_TO_BYTE ..... 2187
LTOD_TO_DATE ..... 2187
LTOD_TO_DINT ..... 2187
LTOD_TO_DT ..... 2187
LTOD_TO_DWORD ..... 2187
LTOD_TO_INT ..... 2187
LTOD_TO_LDATE ..... 2187
LTOD_TO_LDT ..... 2187
LTOD_TO_LINT ..... 2187
LTOD_TO_LREAL ..... 2187
LTOD_TO_LWORD ..... 2187
LTOD_TO_REAL ..... 2187
LTOD_TO_SINT ..... 2187
LTOD_TO_STRING ..... 2187
LTOD_TO_TIME ..... 2187
LTOD_TO_TOD ..... 2187
LTOD_TO_UDINT ..... 2187
LTOD_TO_UINT ..... 2187
LTOD_TO_ULINT ..... 2187
LTOD_TO_USINT ..... 2187
LTOD_TO_WORD ..... 2187
LTOD_TO_WSTRING ..... 2187
LWORD ..... 2235
convert ..... 2159
LWORD_TO___UXINT ..... 2159
LWORD_TO__XINT ..... 2159
LWORD_TO__XXORD ..... 2159
LWORD_TO_BIT ..... 2159
LWORD_TO_BOOL ..... 2159
LWORD_TO_BYTE ..... 2159
LWORD_TO_DATE ..... 2159
LWORD_TO_DINT ..... 2159
LWORD_TO_DT ..... 2159
LWORD_TO_DWORD ..... 2159
LWORD_TO_INT ..... 2159
LWORD_TO_LDATE ..... 2159
LWORD_TO_LDT ..... 2159
LWORD_TO_LINT ..... 2159
LWORD_TO_LREAL ..... 2159
LWORD_TO_LTIME ..... 2159
LWORD_TO_LTOD ..... 2159
LWORD_TO_REAL ..... 2159
LWORD_TO_SINT ..... 2159
LWORD_TO_STRING ..... 2159
LWORD_TO_TIME ..... 2159
LWORD_TO_TOD ..... 2159
LWORD_TO_UDINT ..... 2159
LWORD_TO_UINT ..... 2159
LWORD_TO_ULINT ..... 2159
LWORD_TO_USINT ..... 2159
LWORD_TO_WORD ..... 2159
LWORD_TO_WSTRING ..... 2159
M
M ..... 2231
memory range prefix ..... 2231
M4 interface file for external library ..... 2619
macro ..... 2079
add ..... 2682
insert after ..... 2682
SFC ..... 2079
zoom into ..... 2682
zoom out of ..... 2682
magnification tool ..... 2048
mail service ..... 4361
main action ..... 2076
manage localizations ..... 2605
map pool devices
command ..... 2611
mapping ..... 1836
I/O mapping ..... 1836
OPC UA to IEC ..... 2848
mapping (see I/O mapping) ..... 1837
MAX ..... 2146
MC ..... 126, 129, 1426
MC5102 ..... 1286, 1309
MC5141 ..... 1315
Measuring ranges
AC522 ..... 692
mechanical installation ..... 104
mechanical planning ..... 104
mechanical planning and installation ..... 104
memory
display memory snapshot ..... 2592
dynamic allocation ..... 2201
memory card ..... 126, 129, 1426
MC5102 (micro) ..... 1286, 1309
MC5141 ..... 1315
memory range ..... 2231
memory reserve
function block ..... 2595
online change ..... 2595
memory view ..... 2592
menu ..... 2801
customize ..... 1802
merge changes ..... 4344
message
go to source position ..... 2575
next ..... 2576
previous ..... 2576
message pragma ..... 2269
message view ..... 2583
meta-information ..... 1813
add to project ..... 1813
meter ..... 3201, 3626
$90^{\circ}$, visualization element ..... 3187, 3612
$180^{\circ}$, visualization element ..... 3194, 3619
visualization element ..... 3201, 3626
method ..... 2485
call ..... 1933, 2486
call recursively ..... 1935
call with external implementation ..... 1885
example of recursive call ..... 2282
factorial calculation ..... 2282
FB_Init, FB_Reinit, FB_Exit ..... 2336
interface ..... 2490
monitor ..... 2000
object-oriented programming ..... 2485
reaction to type change ..... 2275
recursive call ..... 2487
virtual call ..... 1933
METHOD ..... 2485
Methode ..... 2120
metrics
code analysis ..... 4226
static analysis ..... 4213, 4226
micro browser ..... 4397
micro memory card MC5102 1286, 1309
micro memory card adapter TA5350-AD ..... 1286, 1309
migrate third party device command ..... 2655
migration ..... 1394, 1799
MIN ..... 2146
MOD ..... 2137
Modbus ..... 1070
parameters ..... 2433
RTU protocol ..... 1716
TCP/IP protocol ..... 1714, 4523
Modbus configurator ..... 1565
Modbus sockets ..... 4568
Modbus TCP ..... 3780
modifier. IL ..... 2087
ModRtuMast ..... 4506
ModRtuToken ..... 4506
ModTcpMast2 ..... 4568
module
call tree ..... 2572
monitoring ..... 1996
area for arrays 2048, 2752
CFC editor ..... 2103
display mode ..... 2654
enable inline monitoring ..... 2569
encoding ..... 2297
function call ..... 2295
inline ..... 1996
interval ..... 2765
options ..... 2792
pragma ..... 2295, 2297
properties ..... 2766
property ..... 2295, 2297
SFC ..... 2063
using pous for implicit checks ..... 1928
monitoring area ..... 2048
dialog ..... 2048
monitoring function implicit ..... 2500
monitoring variable
IEC 61850 server ..... 1679
Motion control library ..... 3958
mounting ..... 106
CM ..... 109
communication module ..... 109
FM ..... 107
function module terminal base ..... 107
I/O modules ..... 113, 123
option boards (AC500-eCo) ..... 114, 127
processor module ..... 109, 114
TA5301-CFA ..... 123
TB ..... 107
terminal base ..... 107
terminal block for onboard I/O (AC500-eCo) ..... 116
terminal unit ..... 111
TU ..... 111
mounting a processor module on DIN rail ..... 119
on metal plate ..... 121
MOVE ..... 2137
move down ..... 2718
move up ..... 2718
MQTT ..... 1722
MQTT client library ..... 4048
MQTT_CONNECTION ..... 4050
MQTT_ERROR_ID ..... 4048
MQTT_MESSAGE ..... 4050
MQTT_QOS ..... 4050
MSSQL ..... 1757, 3833
MUL ..... 2134
multi online change command ..... 2665
multicore ..... 2537
show CPU load ..... 2015
trace ..... 2014
use task-local global variable list ..... 1855
multicore operator
__COMPARE_AND SWAP ..... 2212
XADD ..... 2213
TEST_AND_SET ..... 2215
multitouch
for operating a visualization ..... 2887
implement event handling ..... 2888
visualization ..... 3401
MUX ..... 2147
MySQL ..... 1757, 3833

## N

name
convention, static analysis ..... 4228
nameprefix ..... 4228
attribute ..... 4228
nameprefix, attribute ..... 4230
namespace ..... 2327
automatic ..... 2796
enumeration ..... 2217
GVL ..... 2216
library ..... 2217
of variables ..... 2327
naming ..... 4228
attribute ..... 4228
naming convention ..... 2327, 4219
code analysis ..... 4219
disable ..... 4229
naming ..... 4229
static analysis ..... 4219
suppress ..... 4229
naming, attribute ..... 4229
NE ..... 2149
Negate ..... 2686
negated coil ..... 2095
negated contact ..... 2094
negation, fbd/ld/il ..... 2708
network ..... 1936
address ..... 1939
addressing ..... 1937
comment out ..... 2701
FBD/LD/IL ..... 2091
FBD/LD/IL, insert ..... 2700
FBD/LD/IL, insert below ..... 2701
jump ..... 2712
scan ..... 2427
settings ..... 2761
topology ..... 1937
Network scan ..... 1459, 1509
network variable list (receiver) ..... 2475
add ..... 2475
network variable list (sender) ..... 2475
add ..... 2475
network variables ..... 1946
properties ..... 2760
NetX configuration
EtherNet/IP ..... 2824
NEW ..... 2201
new connection
EtherNet/IP adapter ..... 2827
next message ..... 2583
no AC31 header ..... 1750
no_assign_warning, pragma ..... 2298
no_assign, pragma ..... 2298
no_check, pragma ..... 2298
no_copy, pragma attribute ..... 2299
no_fast_online_change ..... 2291
no_init ..... 4417
no_instance_in_retain, pragma ..... 2300
no_virtual_actions, pragma ..... 2300
no-exit, pragma ..... 2299
node address ..... 1939
noinit, pragma ..... 2299
NOT ..... 2139
NOT, operator in pragma ..... 2326
notification center ..... 2408
notifications ..... 2408
NTP ..... 1716
NumClass ..... 4620
numeric constants ..... 2220
numeric keypad
call, visualization ..... 2890
numpad
visualization ..... 3399
NumTotal ..... 4620
NVL (receiver) ..... 2475
NVL (sender) ..... 2475
0
object
access right ..... 1822
add, visualization ..... 3367
edit ..... 2603
edit (offline) ..... 2603
edit with ..... 2603
find ..... 2582
link to file ..... 2763
open detailed compare view ..... 1407, 1818
select in device tree ..... 2673
select parent object in device tree ..... 2673
Object
add, command . . . . . . . . . . . . . . . . . . . . . . 2598
Properties ..... 2753
Object-oriented programming ..... 1929
obsolete, pragma ..... 2304
occurrence location variable ..... 1906
octal
number ..... 2220
octal number
format definition \%o ..... 3329
OF ..... 2247
array ..... 2247
offline help option ..... 2789
older versions of this document ..... 10
onboard ..... 4436
Onboard I/Os
PM50x2 ..... 196
Onboard I/Os in processor module PM50x2 ..... 196
online
log in to application ..... 2624
logout from application ..... 2627
multiple download ..... 2631
pointer reference ..... 2576
online change ..... 2629
active application ..... 2629
attribute ..... 2291
compiler definition ..... 2291
encrypt ..... 4202
selected application ..... 2629
online config mode ..... 2616
online help
option ..... 2789
online mode
task monitoring ..... 2536
OOP ..... 1929
OPC DA
Communication and protocols$307,308,310,312,4481,4483,4484$
OPC DA server ..... 1750, 1787, 2834
OPC UA
information model ..... 2473
information model repository ..... 2665
OPC UA Client
data source ..... 2424
OPC UA companion information model ..... 2846
OPC UA information model use ..... 2846
OPC UA server ..... 1787, 2834
OPC UA types
IEC types ..... 2848
map to IEC ..... 2848
open ..... 1808
library projects ..... 1808
project ..... 1808
project archives ..... 1808
write-protected project ..... 1808
open ports ..... 1349
open ports and service ..... 1349
OpenDialog ..... 3336
Operand ..... 2235
operating mode
debug ..... 2642
locked ..... 2642
operational ..... 2642
operation
description of LEDs ..... 1431, 4456
operational, operating mode ..... 2642
operator ..... 2129
binding strength ..... 2050
IL ..... 2087
precedence ..... 2050
option board ..... 218
analog input, TA5120-2AI-UI ..... 240
analog input, TA5120-2AI-UIW ..... 240
analog input, TA5123-2AI-RTD ..... 255
analog input, TA5123-2AI-RTW ..... 255
analog output, TA5126-2AO-UI ..... 269
analog output, TA5126-2AO-UIW ..... 269
COMx serial communication, TA5141-RS232I ..... 285
COMx serial communication, TA5141- RS232IW ..... 285
COMx serial communication, TA5142-RS485 ..... 296
COMx serial communication, TA5142-RS485I ..... 289
COMx serial communication, TA5142- RS485IW ..... 289
COMx serial communication, TA5142-RS485W296
digital I/O, TA5110-2DI2DOT ..... 231
digital I/O, TA5110-2DI2DOW ..... 231
digital input, TA5101-4DI ..... 218
digital input, TA5101-4DIW ..... 218
digital output, TA5105-4DOT ..... 224
digital output, TA5105-4DOTW ..... 224
KNX address push button, TA5130-KNXPB ..... 280
KNX adress push button, TA5130-KNXPBW ..... 280
real-time clock, TA5131-RTC ..... 283
option board for processor modules PM50xx ..... 1504
options ..... 2667, 2668
development status ..... 1802
device editor ..... 2786
IEC 61850 server ..... 1713
import assistant ..... 2745
monitoring ..... 2792
SVN ..... 4349, 4350
visualization ..... 3384
OR ..... 2139
OR_ELSE ..... 2140
OR, pragma ..... 2326
order
command, visualization editor ..... 3344
pin ..... 2303
Ordering data
AC522 ..... 698
ORN ..... 2087
output
assign ST ..... 2052
CFC ..... 2109
device ..... 1837
reset ..... 2687
output memory ..... 2231
output pin order ..... 2303
output text
with placeholder and format definition ..... 3330
output variable ..... 2114
output/input module
mounting and demounting ..... 113, 123
overflow data type ..... 2129
overlay icon ..... 4319
SVN ..... 4319
overloading ..... 2153
overview of product family ..... 85
P
pack_mode, pragma ..... 2305
package
manage ..... 2655
uninstall ..... 2655
Package Manager ..... 2655
packages
import assistant ..... 2745
page
CFC ..... 2109
page oriented
CFC object ..... 2101
page size
edit ..... 2686
page-oriented ..... 2101
pane
next ..... 2671
previous ..... 2671
Panel Builder ..... 4397, 4398
Panel Builder 600 ..... 148
panning tool ..... 2048
parallel ..... 2678
parameter
update, fbd/ld/il cfc ..... 2710
parameter mode ..... 2616
Parameterization
AC522 ..... 685
I/O bus ..... 1569
IO bus ..... 1569
parameters
CANbus ..... 2433
device editor ..... 2433
edit ..... 2692
EtherCAT ..... 1610
EtherCAT Master ..... 1613
EtherCAT Slave ..... 1623
Modbus ..... 2433
PROFIBUS DP ..... 2433
PROFINET IO ..... 2433
parameterstringof
pragma, visualization ..... 3338
parametrization
IEC 61850 server ..... 1679
Parametrization I/O bus ..... 1569
PARAMS ..... 2335
pass parameters
pass pointer, visualization ..... 2970
pass-by-reference parameter ..... 2114
password
indicate at login ..... 1827
project ..... 1824
project settings ..... 2772
wrong ..... 2045
password manager ..... 1821
paste after ..... 2683
path3d
camera control ..... 3281, 3706
Path3D ..... 3279, 3704
PB610 ..... 148
PB610 Panel Builder 600 ..... 4397
PDOs ..... 1599
persistence ..... 1920
clean gaps ..... 2719
rearrange list ..... 2719
save values ..... 2719
persistence editor ..... 2476
Persistence Manager remanent variable ..... 1926
PERSISTENT ..... 2122
variable ..... 1923, 2122
persistent variable ..... 1920
declare ..... 1927
saving in a recipe ..... 1928
persistent variable list ..... 2476
PID controller ..... 4424
PID function blocks ..... 4424
pie ..... 3025, 3450
visualization element ..... 3025, 3450
pin
reset ..... 2694
select ..... 2694pin_presentation_order_inputs, pragma attribute2303
pin_presentation_order_outputs, pragma attribute2303
pingroup, pragma ..... 2302
pins
remove ..... 2694
placeholder ..... 2716
with format definition in character string, visual- ization ..... 3329
planning ..... 97
Plastic labels ..... 1330
Plastic markers ..... 1330
PLC
read parameter file to configuration ..... 2616
security ..... 2041
PLC behavior after voltage dip ..... 1480
PLC behaviour after voltage dip ..... 1480
PLC firmware ..... 1446
PLC load ..... 2014
PLC log ..... 2021, 4647
PLC runtime licensing ..... 1446
PLC settings
device editor ..... 2439
PLC shell ..... 2022
device editor ..... 2441
PLC system start-up ..... 4420
PLC_PRG ..... 34
plcload
2008, 2740
PlcLoad
DeviceTrace ..... 2015
PLCopenXML
export/import ..... 2612
import ..... 2612
option ..... 2793
PlcOperationControl ..... 2023
PLCs ..... 85
pluggable label mounting ..... 1328
pluggable marker holder ..... 1328
PM ..... 89, 93, 175
PM50xx ..... 175
PM5012-R-ETH ..... 175
PM5012-T-ETH ..... 175
PM5032-R-ETH ..... 175
PM5032-T-ETH ..... 175
PM5052-R-ETH ..... 175
PM5052-T-ETH ..... 175
PM5072-T-2ETH ..... 175, 4447
PM5072-T-2ETHW ..... 175
PM5082-T-2ETH ..... 175
PM5630 ..... 313
PM5650 ..... 313
PM5670 ..... 313
PM5675 ..... 313
po file ..... 1833
pointer ..... 2243
index access ..... 2244
SUPER ..... 2125
THIS ..... 2126
Pointer
check function CheckPointer ..... 2513
POINTER TO ..... 2243
pointers
Go To Reference ..... 2576
polygon ..... 3012, 3437
polygon, visualization element ..... 3012, 3437
polyline ..... 3012, 3437
visualization element ..... 3012, 3437
POOL ..... 2217
position
breakpoint ..... 2752
POSITION ..... 2214
pot file ..... 1833
potentiometer ..... 3208, 3633
visualization element ..... 3208, 3633
POU ..... 2477
add ..... 2477
change type ..... 2717
cross references ..... 2571
global cross references ..... 2571
implicit checks ..... 2500
monitor function call ..... 2001
POUs view ..... 2583
POU locations ..... 2411
POU view
reference an object ..... 2217
syntax check ..... 2621
POUNAME ..... 2214
POUs for implicit checks ..... 1928
POUs view ..... 2583
power function ..... 2195
operator ..... 2195
power switch ..... 3231, 3656
visualization element ..... 3231, 3656
pragma ..... 2318
analysis:report-multiple-instance-calls ..... 4231
attribute ..... 2271
conditional ..... 2318
dataflow ..... 2279
define ..... 2318
effect on symbol ..... 2315
enable_dynamic_creation ..... 2281
hasattribute ..... 2318
hasconstantvalue ..... 2318
hastype ..... 2318
if ..... 2318
message ..... 2269
no_copy ..... 2299
parameterstringof ..... 3338
ProcessValue ..... 2312, 2698
region ..... 2327
static analysis ..... 4228
undefine ..... 2318
use ..... 1888
VAR_IN_OUT_AS_POINTER ..... 3337
pragmas ..... 2269
precedence, ST ..... 2050
preconditions
drives library ..... 3778
prefix
convention, static analysis ..... 4228
prepare value ..... 2749
pressure control mode ..... 4070
pressure-control mode ..... 4070
previous message ..... 2583
print
input action ..... 3373
page setup ..... 2772
PRIVATE ..... 2485
method ..... 2485
property ..... 2493
private key ..... 2040
Pro CPU ..... 177
Pro Static Analysis tool ..... 4209
process data
EtherCAT Slave ..... 1619
processing order in SFC ..... 2064
processing order, ST ..... 2050
processor load ..... 2740
processor module ..... 175
mounting and demounting 109, 114, 116
processor module selection ..... 97
processor modules ..... 89
ProcessValue ..... 2312
PROFIBUS ..... 354, 1136
PROFIBUS DP parameters ..... 2433
PROFIBUS DP slave ..... 354
PROFINET 365, 1201, 3787
PROFINET Device general ..... 1642
PROFINET Field Device general ..... 1645, 1646
PROFINET IO bus cycle ..... 1642
parameters ..... 2433
PROFINET IO controller general ..... 1640
PROFINET IO device general ..... 1644
PROFINET IO module general ..... 1644
profinet redundancy ..... 4535
program ..... 2478
execute on client, input action ..... 3373
execute on controller, input action ..... 3373
property ..... 2493
PROGRAM ..... 2478
programmable logic controllers ..... 85
programming ..... 145
reference, visualization ..... 2986
progress bar ..... 3152, 3582
visualization element ..... 3152, 3582
prohibited symbols
code check ..... 4227
static analysis ..... 4227
project 1390, 1398, 1808
access protection ..... 1819
add folder ..... 2599
commit accepted changes ..... 2611
compare ..... 1405, 1817, 2607
comparison 1405, 1406, 1817, 1818
create property with key ..... 1813
document ..... 2606
dongle ..... 1825
encryption ..... 1819, 1825
encryption, instructions ..... 1830
export ..... 1815
export/import ..... 1815
file information ..... 2515
filing ..... 1822
functions for accessing properties ..... 1813
include with source code management ..... 1833
information ..... 2515
install in the library repository ..... 2556
key for meta-information ..... 2516
last used ..... 2561
localization ..... 2605
login data ..... 1827
manage localizations ..... 2605
meta-information ..... 2515
migrate V2 project to V3 project ..... 1394, 1799
new ..... 2552
object statistics ..... 2517
open ..... 2553
open by command line ..... 2029
open V2.3 ..... 1809
open, option ..... 2791
password ..... 1824
password protection ..... 1819
POUs for keys ..... 2516
project settings ..... 1814
protection ..... 1819
query information ..... 1813
released ..... 1823
restore ..... 2791
rights management ..... 1822
save ..... 1831
save as ..... 1831
save as compiled library ..... 2556
save as, command ..... 2555
saving in project archive ..... 1832
security ..... 1819
source code ..... 1980
template ..... 2552
toggle localization ..... 2606
transfer ..... 1815, 1816
update ..... 1394, 1799
user management ..... 1825
VisuSymbolLibrary key ..... 2516
write protection ..... 1819, 1823
Project
close ..... 2554
project archive ..... 1832
extract by command line ..... 2029
project compare
configuration ..... 2607
detail ..... 2609
differences ..... 2607
project compression ..... 2791
project documentation print ..... 2606
project environment
symbol library in visualization 2780, 3386
visualization profile ..... 2779, 3385
visualization style ..... 2780, 3386
project localization ..... 1833
create template ..... 2605
localization template ..... 1833
manage ..... 2605
toggle ..... 2606
project restore information ..... 2791
project setting
command ..... 2603
project settings ..... 2514
command ..... 2604
make ..... 1814
object ..... 2514
rules, Static Analysis ..... 4218
SFC ..... 2767
SVN ..... 4350
user management ..... 1825
users and groups ..... 2768
visualization ..... 2776, 3387
properties
access control ..... 2757
bitmap ..... 2758
boot application ..... 2754
build ..... 2755
build, C-integration ..... 2756
cam ..... 2763
common ..... 2753
device ..... 2765
external file ..... 2757
IEC 61850 server ..... 1679
image pool ..... 2764
link to file ..... 2763
monitoring ..... 2765, 2766
network settings ..... 2761
network variables ..... 2760
security ..... 2754
SFC ..... 2762
task configuration ..... 2533
text list ..... 2765
Properties
of an object ..... 2753
property
CFC execution order ..... 2761
monitor ..... 1998
object ..... 2493
object-oriented programming ..... 2493
SFC, do not display embedded objects ..... 2684
PROPERTY ..... 2493
object ..... 2493
PROTECTED ..... 2485
method ..... 2485
property ..... 2493
protected environment ..... 1342
protection
data security ..... 1971
project ..... 1819
protocols
IEC 60870-5-104 (Telecontrol) ..... 1647
Protocols
BACnet ..... 1733, 3833
Modbus RTU ..... 1716
Modbus TCP/IP ..... 1714, 4523
MQTT ..... 1722
NTP ..... 1716
OPC UA ..... 1787, 2834
Secure ..... 1724
SNTP ..... 1716
UDP ..... 1750
proxy
access data ..... 2793
server option ..... 2793
server, setting ..... 2793
PUBLIC ..... 2485
method ..... 2485
property ..... 2493
public key ..... 2040
push switch ..... 3231, 3656
visualization element ..... 3231, 3656
push switch LED ..... 3231, 3656
visualization element ..... 3231, 3656
Python ..... 4362
Q
Q ..... 2231
memory range prefix ..... 2231
qualified_only, pragma ..... 2312
qualifiers for SFC actions ..... 2065
QUERYINTERFACE ..... 2204
QUERYPOINTER ..... 2205
R
$\mathrm{R}=$ ..... 2053
reset assignment ..... 2053
radio buttons ..... 3161, 3586
visualization element ..... 3161, 3586
READ_ONLY ..... 2335
READ_WRITE ..... 2335
REAL ..... 2236
constant ..... 2221
convert ..... 2171
literal ..... 2221
REAL_TO___UXINT ..... 2171
REAL_TO_XINT ..... 2171
REAL_TO__XWORD ..... 2171
REAL_TO_BIT ..... 2171
REAL_TO_BOOL ..... 2171
REAL_TO_BYTE ..... 2171
REAL_TO_DATE ..... 2171
REAL_TO_DINT ..... 2171
REAL_TO_DT ..... 2171
REAL_TO_DWORD ..... 2171
REAL_TO_INT ..... 2171
REAL_TO_LINT ..... 2171
REAL_TO_LREAL ..... 2171
REAL_TO_LWORD ..... 2171
REAL_TO_SINT ..... 2171
REAL_TO_STRING ..... 2171
REAL_TO_UDINT ..... 2171
REAL_TO_UINT ..... 2171
REAL_TO_ULINT ..... 2171
REAL TO WORD ..... 2171
REAL_TO_WSTRING ..... 2171
real-time clock ..... 4444
Realization with centralized PLC based motion control ..... 3982
realtime clock ..... 4444
rearrange list ..... 2719
recent projects ..... 2561
recipe ..... 2003, 2522
add ..... 2723
create, input action ..... 3373
delete, input action ..... 3373
insert variable ..... 2723
load ..... 2724
load and write ..... 2725
load from controller ..... 2006
load from device ..... 2727
load from file ..... 2005
load, input action ..... 3373
read ..... 2725
read and save ..... 2726
read, input action ..... 3373
remanent variable ..... 1926
remove ..... 2724
remove variables ..... 2726
save ..... 2724
save, input action ..... 3373
visualization ..... 2939
write ..... 2725
write, input action ..... 3373
recipe definition ..... 2522
recipe file
load ..... 2005
recipe management ..... 2003
controller ..... 2006
memory usage ..... 2006
Recipe Manager ..... 2519
recording, see data recording ..... 2805
rectangle ..... 2988, 3413
visualization element ..... 2988, 3413
Recycling ..... 158
redundancy ..... 4535
REF= ..... 2054, 2245
refactoring ..... 1910
add variable ..... 2578
add/remove variable ..... 1911
code clone ..... 4216
declaration order of variables ..... 1912
duplicated code ..... 4216
function extraction ..... 4215
option ..... 2794
remove variable ..... 2580
rename ..... 2577
rename variable ..... 1910
reorder variables ..... 2581
update referenced pins ..... 2578
reference ..... 2245
_ISVALIDREF ..... 2246
data type ..... 2245
test operator ..... 2246
valid ..... 2246
REFERENCE TO ..... 2245
reflection ..... 2293, 2313
pragma attribute ..... 2313
refresh
structured variables ..... 2727
released ..... 1823
remanent ..... 2122
recipe ..... 1926
remanent variable of the Persistence Manager ..... 1926
variable ..... 2122
remanent variables
AC500 V3 products ..... 4412
Remote Alarms
Object ..... 2412
remote data
visualize ..... 1959
remove
IL line ..... 2707
Reset ..... 2687
Set ..... 2687
unused parameters, FBD/LD ..... 2710
remove force list ..... 2650
remove unused FB call parameters ..... 2710
rename
refactoring ..... 2577
repair
box ..... 2710
REPEAT ..... 2058
replace ..... 1910
command ..... 2564
replace a module ..... 152
replacement of a module ..... 152
replacement of an I/O module with hot swap ..... 153
report
IEC 61850 server ..... 1689
repository ..... 2657, 4316
browse SVN repository ..... 4322
information model OPC UA ..... 2665
library ..... 2657
OPC UA information model ..... 2665
SVN ..... 4316
visualization element ..... 3361
reserve memory
online change ..... 2595
reset
application ..... 1990
application (reset cold) ..... 2633
application (reset origin) ..... 2635
application (reset warm) ..... 2634
assignment ST ..... 2053
cold ..... 2633
device to origin ..... 2636
origin ..... 2635
origin device ..... 2636
SVN ..... 4339
warm ..... 2634
Reset
output ..... 2687
remove ..... 2687
reset coil ..... 2095
insert ..... 2704
reset origin ..... 2635
application ..... 2635
reset warm ..... 149
restore ..... 2024
restore values from recipe ..... 2719
RET ..... 2087
RETAIN ..... 2124
remanent variable ..... 1925
variable ..... 2124
RETC ..... 2087
RETCN ..... 2087
return
CFC . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2111
FBD/LD/IL . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2093
insert, FBD/LD/IL . . . . . . . . . . . . . . . . . . . . . . . . 2703
RETURN . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2059
revision
copy to branch/tag . . . . . . . . . . . . . . . . . . . . . 4347
select in SVN . . . . . . . . . . . . . . . . . . . . . . . . . 4351
RGB . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2914
color space . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2914
RGBA . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2914
color space extended with alpha channel . . . 2914
rights management . . . . . . . . . . . . . . . . . . . . . . 1821
project . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1822
rising edge detection . . . . . . . . . . . . . . . . . . . . . 2708
rocker switch . . . . . . . . . . . . . . . . . . . . . . 3231, 3656
visualization element . . . . . . . . . . . . . . 3231, 3656
ROL . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2143
ROR . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2144
rotary switch
visualization element . . . . . . . . . . . . . . 3235, 3660
rounded rectangle . . . . . . . . . . . . . . . . . . 2988, 3413
2988, 3413
routing . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1937
RTC . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4444
run
stepping . . . . . . . . . . . . . . . . . . . . . . . . . . 1985
to cursor . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2648
using step out . . . . . . . . . . . . . . . . . . . . . . . . . 2647
run static analysis . . . . . . . . . . . . . . . . . . . . . . . . 4212
runtime
security . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2041
runtime licensing
command . . . . . . . . . . . . . . . . . . . . . . . . . . 2617
runtime system files
generate . . . . . . . . . . . . . . . . . . . . . . . . . . . 2619
runtime system service
disable.................................. . . 2022
S
S= . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2052
set assignment . . . . . . . . . . . . . . . . . . . . . . . 2052
S500
description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 89
system structure . . . . . . . . . . . . . . . . . . . . . . . . 86
S500-eCodescription93
system structure ..... 87
SA0001 ..... 4234
SA0002 ..... 4234
SA0003 ..... 4235
SA0004 ..... 4235
SA0005 ..... 4263
SA0006 ..... 4236
SA0007 ..... 4237
SA0008 ..... 4237
SA0009 ..... 4238
SA0010 ..... 4239
SA0011 ..... 4239
SA0012 ..... 4240
SA0013 ..... 4240
SA0014 ..... 4241
SA0015 ..... 4242
SA0016 ..... 4242
SA0017 ..... 4243
SA0018 ..... 4243
SA0019 ..... 4258
SA0020 ..... 4244
SA0021 ..... 4245
SA0022 ..... 4245
SA0023 ..... 4246
SA0024 ..... 4246
SA0025 ..... 4247
SA0026 ..... 4247
SA0027 ..... 4248
SA0028 ..... 4248
SA0029 ..... 4249
SA0031 ..... 4249
SA0032 ..... 4250
SA0033 ..... 4250
SA0034 ..... 4252
SA0035 ..... 4251
SA0036 ..... 4251
SA0037 ..... 4252
SA0038 ..... 4253
SA0040 ..... 4254, 4296
SA0041 ..... 4255
SA0042 ..... 4256
SA0043 ..... 4256
SA0044 ..... 4257
SA0046 ..... 4297
SA0121 ..... 4292
SA0047 ..... 4264
SA0048 ..... 4264
SA0051 ..... 4265426642664267SA0054
4268
SA0055
4268
SA0056
SA0057 ..... 4269
SA0058 ..... 4269
SA0059 ..... 4271
SA0060 ..... 4271
SA0061 ..... 4271
SA0062 ..... 4272
SA0063 ..... 4272
SA0064 ..... 4273
SA0065 ..... 4273
SA0066 ..... 4274
SA0072 ..... 4276
SA0073 ..... 4276
SA0075 ..... 4278
SA0076 ..... 4279
SA0077 ..... 4280
SA0078 ..... 4280
SA0080 ..... 4276
SA0081 ..... 4277
SA0090 ..... 4281
SA0095 ..... 4281
SA0100 ..... 4282
SA0101 ..... 4283
SA0102 ..... 4283
SA0103 ..... 4284
SA0105 ..... 4285
SA0106 ..... 4286
SA0107 ..... 4287
SA0111 ..... 4289
SA0112 ..... 4289
SA0113 ..... 4289
SA0114 ..... 4290
SA0115 ..... 4290
SA0117 ..... 4290
SA0118 ..... 4295
SA0119 ..... 4291
SA0120 ..... 4291
SA0122 ..... 4293
SA0123 ..... 4293
SA0124 ..... 4295
SA0125 ..... 4295
SA0130 ..... 4259
SA0131 ..... 4260
SA0132 ..... 4261
SA0133 ..... 4261
SA0134 ..... 4262
SA0140 ..... 4296
SA0145 ..... 4298
SA0147 ..... 4293
SA0148 ..... 4294
SA0150 ..... 4299
SA0160 ..... 4299
SA0161 ..... 4300
SA0162 ..... 4301
SA0163 ..... 4302
SA0164 ..... 4303
SA0165 ..... 4303
SA0166 ..... 4304
SA0167 ..... 4304
sa0168 ..... 4305
sa0169 ..... 4306
SAE J1939 ..... 1522
Safety devices ..... 94, 174
Safety instructions
drives library ..... 3778
safety notice ..... 10, 12
sample
show in the trace editor ..... 2007
samples
save in trace file ..... 2009
save
project archive ..... 1832
project, option ..... 2791
Save
project ..... 1831
save current values to recipe ..... 2719
save the project ..... 2554
save values to recipe ..... 2719
saving
project ..... 1822
scalar product ..... 2257
scan devices ..... 1607, 1628, 2600, 2832
SCE in LD ..... 2096
scope ..... 2113
script
execute ..... 2667
Script
enable tracing ..... 2667
script file
run by command line ..... 2030
scripting
execute ..... 2666
execute script file ..... 2666
Scripting
enable script tracing ..... 2667
Scripts
Python ..... 4362
scroll bar ..... 3125, 3550
visualization element ..... 3125, 3550
SD card ..... 126, 129
SD memory card ..... 126, 129, 1426
sdcard ..... 1426
sdcard.ini ..... 1426
SDOs ..... 1601
search 1910, 2565 ..... 2566
object ..... 2582
search order
identifiers ..... 2333
variable name ..... 2333
Secure communication ..... 1724
secure operation ..... 1342
Secure protocols ..... 1724
security ..... 1340, 2039
add device user ..... 2637
certificate ..... 2040
certificate via PLC shell ..... 2044
certificates ..... 1819, 2592
client ..... 2043
communication with controller ..... 4201
data security ..... 1971
development system ..... 2041
device ..... 1967
disable user management ..... 2045
encrypt the boot application, download, and online change ..... 4202
encrypted communication ..... 2427
encryption, signing, certificates ..... 1820
general information ..... 2039
password device user ..... 2639
project encryption ..... 1819
project settings ..... 2772
remove device user ..... 2638
runtime system / PLC ..... 2041
Security Agent ..... 4201
unencrypted communication ..... 2046
WebVisu ..... 2042
Security Agent ..... 4201
certificate ..... 4201
security functions certificate ..... 2040
development system ..... 2041
general information ..... 2039
runtime system / PLC ..... 2041
WebVisu ..... 2042
security notice ..... 10, 12
security screen ..... 2592
SEL ..... 2145
select matching bracket ..... 2568
select none
of the seclected visualization elements ..... 3365
selection
alarm class ..... 3389
alarm group ..... 3390
selector
CFC ..... 2111
semi-transparency
visualization ..... 3401
Sequence generator ..... 4063
sercos
generate xml ..... 2614
Server
FTP ..... 1721
services ..... 1349
set
access method, interface ..... 2490
Set
accessor method ..... 2493
assignment ST ..... 2052
output ..... 2687
remove ..... 2687
set coil ..... 2095
insert ..... 2704
set output connection, FBD/LD ..... 2708
set/reset, FBD/LD/IL ..... 2708
settings
code check ..... 4217
static analysis ..... 4217
SFC ..... 1880
action ..... 2075
action qualifiers ..... 2065
analyzation library ..... 2072
analyzation, library ..... 1884
branch ..... 2078
build ..... 2763
code generation ..... 2763, 2767
copy implementation ..... 2678
copy reference ..... 2678
do not display embedded objects ..... 2684
duplication mode ..... 2678
element properties ..... 2080
implicit variables ..... 2066
init step ..... 2675
jogging mode ..... 2067
jump ..... 2079
library ..... 2767
macro ..... 2079
online mode ..... 2063
processing order ..... 2064
programming ..... 1880
project settings ..... 2767
properties ..... 2762
step ..... 2073
step time ..... 2066
transition ..... 2073
SFC editor ..... 2062
character set ..... 2795
layout ..... 2795
online, step time ..... 2796
options ..... 2795
properties, visibility ..... 2796
settings ..... 2795
step actions, options ..... 2795
toolbar ..... 2048
SFC flag ..... 2067
shadowing ..... 2333
shadowing rules ..... 2333
SHL ..... 2141
short form feature ..... 1887
short-circuit evaluation ..... 2096
show source position: ..... 2589
show windows ..... 1807
show/hide implementation view ..... 2672
SHR ..... 2142
signature
compiled library ..... 2033
encryption ..... 2039
enforce signing of compiled libraries ..... 2033
signing
boot application ..... 1915
certificate ..... 1820
library project ..... 2517
with certificate, instructions ..... 1830
Simple motion ..... 4543
simulation ..... 1980
command ..... 2640
for testing ..... 1980
SIN ..... 2196
single cycle ..... 2645
SINT ..... 2235
convert ..... 2159
SINT_TO___UXINT ..... 2159
SINT TO XINT ..... 2159
SINT_TO__XWORD ..... 2159
SINT_TO_BIT ..... 2159
SINT_TO_BOOL ..... 2159
SINT_TO_BYTE ..... 2159
SINT_TO_DATE ..... 2159
SINT_TO_DINT ..... 2159
SINT TO DT ..... 2159
SINT_TO_DWORD ..... 2159
SINT_TO_INT ..... 2159
SINT_TO_LDATE ..... 2159
SINT_TO_LDT ..... 2159
SINT_TO_LINT ..... 2159
SINT_TO_LREAL ..... 2159
SINT_TO_LTIME ..... 2159
SINT_TO_LTOD ..... 2159
SINT_TO_LWORD ..... 2159
SINT_TO_REAL ..... 2159
SINT_TO_STRING ..... 2159
SINT_TO_TIME ..... 2159
SINT_TO_TOD ..... 2159
SINT_TO_UDINT ..... 2159
SINT_TO_UINT ..... 2159
SINT_TO_ULINT ..... 2159
SINT_TO_USINT ..... 2159
SINT_TO_WORD ..... 2159
SINT_TO_WSTRING ..... 2159
SIZEOF ..... 2138
slider ..... 3134, 3559
visualization element ..... 3134, 3559
SM560-S ..... 94, 174
SM560-S-FD-1 ..... 94, 174
SM560-S-FD-4 ..... 94, 174
smart tag ..... 1888
SmartCoding ..... 1885
SmartCoding, options ..... 2796
SMTP/SMTPS ..... 4361
SNMP ..... 4362
SNTP ..... 1716
IEC 61850 server ..... 1707
software ..... 145
source code ..... 1980
download from controller ..... 2559
download to controller ..... 2560
download, project setting ..... 2770
management ..... 1833
write, to connected device ..... 2631
space character
in the text editor ..... 2799
spin box ..... 3140, 3565
visualization element 3140, 3565
SQRT ..... 2194
ST ..... 1878, 2087
assignment ..... 2051
expression ..... 2050
extended ..... 1879
format code ..... 2581
programming in ..... 1879
$\mathrm{R}=$ ..... 2053
reset assignment ..... 2053
S= ..... 2052
set assignment ..... 2052
ST code
execute, input action ..... 3379
extract ..... 4215
ST code in FBD, LD ..... 2094
ST editor ..... 2049
automatic formatting ..... 2049
browse ..... 2049
format code ..... 2581
online operation ..... 2050
option ..... 2798
syntax error ..... 2049
stack checking of recursive methods ..... 2281
standard commands ..... 2562
Standard CPU ..... 176
standard data types ..... 2234
standard keyboard handling
activate, visualization ..... 2895
standard metrics ..... 4213
start page ..... 2596
startup parameters
EtherCAT module ..... 1624
EtherCAT Slave ..... 1619
state
device state ..... 4606
state LEDs ..... 148, 1338
State LEDs
AC522 ..... 691
statement ..... 2055
IF ..... 2055
set next ..... 2648
show next ..... 2648
static
code analysis ..... 1904
static analysis
getting started ..... 4209
pragmas ..... 4228
run ..... 4212
Static Analysis Light ..... 1904
project settings ..... 2773
static variable ..... 2119
Statistic analysis ..... 4209
status
device editor ..... 2460
device state ..... 4606
EtherCAT ..... 1610
KNX ..... 1730
SFC actions ..... 2066
SFC steps ..... 2066
status bar
IEC 61850 server ..... 1686
status LEDs ..... 148, 1338
status LEDs, display and control elements ..... 148, 1338
step ..... 2073
add exit action, SFC ..... 2678
insert ..... 2676
insert after ..... 2676
SFC, add entry action ..... 2678
SFC, duplication mode ..... 2678
step action
SFC ..... 2075
step into ..... 2647
run ..... 2647
step over ..... 2646
run ..... 2646
step status ..... 2066
step-transition
add ..... 2677
insert after ..... 2677
STN ..... 2087
strict
pragma for enumeration ..... 2265
STRING ..... 2237
convert ..... 2175
data type ..... 2237
index access ..... 2244
string constants ..... 2221
STRING_TO___UXINT ..... 2175
STRING_TO__UXWORD ..... 2175
STRING_TO___XINT ..... 2175
STRING_TO_BIT ..... 2175
STRING_TO_BOOL ..... 2175
STRING_TO_BYTE ..... 2175
STRING_TO_DATE ..... 2175
STRING_TO_DINT ..... 2175
STRING_TO_DT ..... 2175
STRING_TO_DWORD ..... 2175
STRING_TO_INT ..... 2175
STRING_TO_LDATE ..... 2175
STRING_TO_LDT ..... 2175
STRING_TO_LINT ..... 2175
STRING_TO_LREAL ..... 2175
STRING_TO_LTIME ..... 2175
STRING TO LTOD ..... 2175
STRING_TO_LWORD ..... 2175
STRING_TO_REAL ..... 2175
STRING_TO_SINT ..... 2175
STRING_TO_TIME ..... 2175
STRING_TO_TOD ..... 2175
STRING_TO_UDINT ..... 2175
STRING_TO_UINT ..... 2175
STRING_TO_ULINT ..... 2175
STRING_TO_USINT ..... 2175
STRING_TO_WORD ..... 2175
STRING_TO_WSTRING ..... 2175
strings
convert ..... 2174
StringUtils library ..... 3774
STRUCT ..... 2261
structure ..... 2261
access ..... 2228
BIT ..... 2262
data type ..... 2261
extend ..... 2261
EXTENDS ..... 2261
object DUT ..... 2461
symbolic bit access ..... 2262
Structure: DrvPdPrmDpv1DataType ..... 3832
Structures and enumerations ..... 4048
style color
select, visualization element ..... 2877
visualization ..... 2876
style property
assign to a visualization element ..... 2979
localize ..... 2985
SUB ..... 2135
subrange types ..... 2268
subsequent, pragma ..... 2313
Subversion
source management ..... 1833
SUPER ..... 2125
suppress warning, pragma ..... 2315
SVN ..... 1833, 4315
_VERSION_INFO ..... 4355
checkout ..... 4327
commands ..... 4320
info ..... 4335
overlay icon ..... 4319
project settings ..... 4350
repository ..... 4316
repository browser ..... 4322
version control ..... 4316
version Info ..... 4355
switch ..... 3231, 3656
visualization element 3231, 3235, 3656, 3660
symbol
access rights ..... 1941
access to controller ..... 2458
overlay ..... 4319
symbol configuration ..... 1941, 2523
access rights ..... 2523
add ..... 2523
comments and attributes ..... 2526
data layout ..... 2523
editor ..... 2523
OPC UA ..... 2523
symbol set ..... 1941
task synchronization ..... 2527
symbol file ..... 2523
symbol library
project environment of visualization . . . 2780, 3386
update ..... 2780, 3386
symbol rights
device editor ..... 2458
symbol access ..... 1941
symbol set
symbol configuration ..... 1941
symbol, pragma ..... 2314
synchronize
cycle-consistent variables ..... 1855
file by application download ..... 2436
syntax check ..... 2621
system diagnosis ..... 4613, 4619
system event ..... 2533
function call ..... 2533
system libraries
cmp libraries ..... 3773
system protection ..... 1341
system structure
AC500 ..... 86
AC500-eCo ..... 87
S500 ..... 86
S500-eCo ..... 87
system time
output in visualization ..... 3331
system variable ..... 2022
operation control ..... 2022
SYSTEM.VAR_INFO ..... 2208
data structure ..... 2208
T
T
constant ..... 2223
literal ..... 2223
TA521 ..... 1319
TA523 ..... 1328
TA524 ..... 1323
TA525 ..... 1330
TA526 ..... 1324, 1331
TA535 ..... 1332
TA5101-4DI
digital input option board ..... 218
TA5101-4DIW
digital input option board ..... 218
TA5105-4DOT
digital output option board ..... 224
TA5105-4DOTW
digital output option board ..... 224
TA5110-2DI2DOT
digital I/O option board ..... 231
TA5110-2DI2DOW
digital I/O option board ..... 231
TA5120-2AI-UI
analog input option board ..... 240
TA5120-2AI-UIW
analog input option board ..... 240
TA5123-2AI-RTD
analog input option board ..... 255
TA5123-2AI-RTW
analog input option board ..... 255
TA5126-2AO-UI analog output option board ..... 269
TA5126-2AO-UIW
analog output option board ..... 269
TA5130-KNXPB
option board KNX address push button ..... 280
TA5130-KNXPBW
option board KNX adress push button ..... 280
TA5131-RTC
option board for real-time clock ..... 283
TA5141-RS232
option board, COMx serial communication ..... 285
TA5141-RS232IW
option board, COMx serial communication ..... 285
TA5142-RS485
option board, COMx serial communication ..... 296
TA5142-RS485I
option board, COMx serial communication ..... 289
TA5142-RS485IW
option board, COMx serial communication ..... 289
TA5142-RS485W
option board, COMx serial communication ..... 296
TA5211-TSCL-B ..... 116, 1292
TA5211-TSPF-B ..... 116, 1292
TA5212-TSCL ..... 116, 1292
TA5212-TSPF ..... 116, 1292
TA5220-SPF5 ..... 116, 1292
TA5220-SPF6 ..... 116, 1292
TA5220-SPF7 ..... 116, 1292
TA5220-SPF8 ..... 116, 1292
TA5300-CVR ..... 1302
TA5350-AD ..... 1286, 1309
TA5400-SIM ..... 1305
TA5450-CASE ..... 21
tab
selection, visualization elements ..... 3342
tab group
new horizontal ..... 2670
new vertical ..... 2670
tab order
element list ..... 3342
table ..... 3106, 3531
display structured variables ..... 2864, 2918
visualization element ..... 3106, 3531
visualizing data arrays ..... 2917
table of contents ..... 2674
tabs ..... 3084, 3509
reference visualizations ..... 2948
visualization element ..... 3084, 3509
TAN ..... 2197
Target change ..... 1414, 1475
TargetVisu
object ..... 3408
task ..... 2538
check ..... 1994
configuration ..... 2538
cycle consistency ..... 1855
cycle times ..... 2021
jitter, latency ..... 1915
monitor ..... 2021
monitoring ..... 2536
processing order ..... 1914
statistics ..... 2021
task cycle time ..... 2538
task-local variables ..... 1855
type ..... 2538
watchdog ..... 2538
task configuration ..... 1914
basic settings ..... 2533
basics ..... 1914
create ..... 1914
jitter latency ..... 1915
object ..... 2533
properties ..... 2533
task deployment
check ..... 1994
device editor ..... 2459
task groups ..... 2537
task monitoring, online ..... 2536
task-local
declare GVL ..... 1855
GVL ..... 2466
TB ..... 90, 377
TB56xx ..... 377
TB511
Technical data ..... 383
TB521
Technical data ..... 383
TB523
Technical data ..... 383
TB541
Technical data ..... 383
TB5600 ..... 377
Technical data ..... 383
TB5610 ..... 377
Technical data ..... 383
TB5620 ..... 377
Technical data ..... 383
TB5640 ..... 377
Technical data ..... 383
TB5660 ..... 377
Technical data ..... 383
Technical data
AC522 ..... 694
TB511 ..... 383
TB521 ..... 383
TB523 ..... 383
TB541 ..... 383
TB5600 ..... 383
TB5610 ..... 383
TB5620 ..... 383
TB5640 ..... 383
TB5660 ..... 383
teClass ..... 4618
teEvent ..... 4618
teHwld ..... 4618
Telecontrol ..... 1647
template element ..... 3350
visualization ..... 3350
temporary variable ..... 2119
terminal base ..... 377
mounting and demounting ..... 107, 111
terminal block for onboard I/O (AC500-eCo) mounting and demounting ..... 116
terminal unit
for communication modules ..... 1270
for S500 I/O modules ..... 938
Terminal unit for PROFINET communication inter- face modules ..... 1282
terminal units for communication interface modules ..... 1274, 1278
terminal units for l/O modules ..... 938, 945
test functions ..... 2500
also in libraries ..... 2500
TEST_AND_SET ..... 2215
testing a program ..... 51, 82
text
display, visualization ..... 2878
output configuration ..... 2881
translate and manage ..... 1891
visualization, multi-language capability ..... 2906
text display
animate in visualization ..... 2883
animating with a visualization element ..... 2914
text editor ..... 2569, 3274, 3699
option ..... 2798
show whitespace ..... 2566
visualization element ..... 3274, 3699
Text Editor
configuring, visualization ..... 2934
text field ..... 3113, 3538
configure dynamic text output ..... 2880
configure input ..... 2882
visualization element ..... 3113, 3538
text file
configuring the display, visualization ..... 2934
configuring the processing, visualization ..... 2934
text input
define for all visualizations throughout the application ..... 2891
text list
add language ..... 2728
add language and translate text ..... 1891
availability ..... 2765
check ID ..... 2731
compare and export differences ..... 1893
display text dynamically ..... 1898
download ..... 2765
DUT ..... 2732
export ..... 1891
export as unicode text ..... 2729
export everything as text ..... 2728
export/import ..... 2729
for dynamic application ..... 1898
for input assistance ..... 1892
import file ..... 1892
insert text ..... 2729
names and directories for visualization ..... 3385
object ..... 2532
properties ..... 2765
remove language ..... 2730
remove unused entries ..... 2731
rename language ..... 2730
update ID ..... 2731
visualization ..... 2906
text list support
add ..... 2732
DUT ..... 2732
TIME TO UDINT ..... 2182
TIME_TO_UINT ..... 2182
TIME_TO_ULINT ..... 2182
TIME_TO_USINT ..... 2182
TIME_TO_WORD ..... 2182
TIME_TO_WSTRING ..... 2182
TIME() ..... 2233
TO__UXINT ..... 2153
TO__XINT ..... 2153
TO__XWORD ..... 2153
TO_BIT ..... 2153
TO_BOOL ..... 2153
TO_BYTE ..... 2153
TO_DATE ..... 2153
TO_DINT ..... 2153
TO_DT ..... 2153
TO_DWORD ..... 2153
TO_INT ..... 2153
TO_LDATE ..... 2153
TO_LDT ..... 2153
TO_LINT ..... 2153
TO_LREAL ..... 2153
TO_LTIME ..... 2153
TO LTOD ..... 2153
TO_LWORD ..... 2153
TO_REAL ..... 2153
TO_SINT ..... 2153
to_string ..... 2314
pragma attribute ..... 2314
TO_STRING ..... 2153
TO_TIME ..... 2153
TO_TOD ..... 2153
TO_UDINT ..... 2153
TO_UINT ..... 2153
TO_ULINT ..... 2153
TO_USINT ..... 2153
TO_WORD ..... 2153
TO_WSTRING ..... 2153
TOD ..... 2238
convert ..... 2187
data type ..... 2238
keyword ..... 2225
TOD_TO___XWORD ..... 2187
TOD_TO__UXINT ..... 2187
TOD_TO_XINT ..... 2187
TOD TO BOOL ..... 2187
TOD_TO_BYTE ..... 2187
TOD_TO_DATE ..... 2187
TOD_TO_DINT ..... 2187
TOD_TO_DT ..... 2187
TOD_TO_DWORD ..... 2187
TOD_TO_INT ..... 2187
TOD_TO_LDATE ..... 2187
TOD_TO_LDT ..... 2187
TOD_TO_LINT ..... 2187
TOD_TO_LREAL ..... 2187
TOD_TO_LTOD ..... 2187
TOD_TO_LWORD ..... 2187
TOD_TO_REAL ..... 2187
TOD_TO_SINT ..... 2187
TOD_TO_STRING ..... 2187
TOD_TO_TIME ..... 2187
TOD_TO_UDINT ..... 2187
TOD_TO_UINT ..... 2187
TOD_TO_ULINT ..... 2187
TOD_TO_USINT ..... 2187
TOD_TO_WORD ..... 2187
TOD_TO_WSTRING ..... 2187
toggle localization ..... 2606
toggle subview ..... 2672
toggle/tap variable couple with Button state variable . . . . . 3098, 3523
toolbar ..... 2802
customize ..... 1804
toolbox ..... 2584
tooltip
visualization, multi-language capability ..... 2906
total current consumption ..... 101
trace ..... 2007
access all traces on controller ..... 2014
add variable ..... 2732
advanced settings ..... 2803
advanced settings, visualization element ..... 3391
assign task ..... 2011
CmpTraceMgr.library ..... 2007
configuration ..... 2804
configure ..... 2733
configure data recording ..... 2805
configure display ..... 2012
configure recording, visualization ..... 3355
configure variable ..... 2011
configure variables, visualization ..... 3356
configure, visualization ..... 3355
control data recording ..... 2014
convert to multi-channel ..... 2737
convert to single-channel ..... 2738
cpuload, plcload ..... 2740
create configuration ..... 2010
data of IEC variable ..... 2010
delete from runtime ..... 2740
delete variable ..... 2011
DeviceTrace ..... 2007
DeviceTrace object ..... 2426
display setting of visualization ..... 3391
download configuration to controller ..... 2734
editor ..... 2542
file formats ..... 2009
getting started ..... 2009
list ..... 2739
load file ..... 2737
load to trace editor ..... 2742
manage as file ..... 2016
navigate in data in diagrams ..... 2015
navigate in diagram ..... 2544
object ..... 2541
online list ..... 2739
open statistics ..... 2742
processor load ..... 2740
read y-value ..... 2733
runtime buffer ..... 2803
runtime system component CmpTraceMgr ..... 2008
save samples to file ..... 2009
save to file ..... 2741
show statistics ..... 2016
start ..... 2741
stop ..... 2741
trace cursor ..... 2733
trigger ..... 2008
upload ..... 2742
visualization element ..... 3240, 3665
trace configuration
export ..... 3357
export symbolic ..... 2735
trace element
getting started ..... 2926
insert elements for control, visualization ..... 3358
record data of a variable ..... 2927
wizard of visualization element ..... 3358
trace graph
compress ..... 2733
reset to default view ..... 2740
stretch ..... 2742
trace manager (see CmpTraceMgr) ..... 2008
training case ..... 21
transfer parameters
update ..... 2952
transition ..... 2073, 2499
insert ..... 2677
insert after ..... 2676
SFC, do not display embedded objects ..... 2684
trend
basis, visualization ..... 2928
configure display settings, command . . 3353, 3359
configure recording, visualization ..... 2546
configure variables, visualization ..... 2547
configure, visualization ..... 3360
visualization element ..... 3246, 3671
Trend
insert elements for control, command ..... 3360
trend configuration
add parameters ..... 2020
add variable ..... 2019
assign tasks ..... 2018
configure additional buffering ..... 2020
configure data buffering on RTS ..... 2020
delete variable ..... 2019
start conditional ..... 2019
trend recording ..... 2016, 2545
configure ..... 2018
getting started ..... 2017
set additional buffer ..... 2810
storage configuration ..... 2809
trend recording task ..... 2548
trend recording manager ..... 2545
trend visualization
edit ..... 2931
programming ..... 2931
sine-shaped curve of IEC variable, example ..... 2929
Trend visualization getting started ..... 2928
trigger
activate in trace configuration ..... 2012
reset trace configuration ..... 2740
trigger option
IEC 61850 terver ..... 1686
Trouble-shooting ..... 4209
TRUE ..... 2220
TRUNC ..... 2193
TRUNC_INT ..... 2193
TRY ..... 2206
TU 91, 938, 1270
TU507 ..... 1270
TU508 ..... 1270
TU509 ..... 1274
TU510 ..... 1274
TU515 ..... 938
TU516 ..... 938
TU517 ..... 1278
TU518 ..... 1278
TU520 ..... 1282
TU531 ..... 945
TU532 ..... 945
TU541 ..... 938
TU542 ..... 938
TU582-S ..... 94, 174
type ..... 2461
object DUT ..... 2461
TYPE ..... 2263
typed literals ..... 2228
U
UDINT ..... 2235
convert ..... 2159
UDINT_TO___UXINT ..... 2159
UDINT_TO___XINT ..... 2159
UDINT_TO___XWORD ..... 2159
UDINT_TO_BIT ..... 2159
UDINT_TO_BOOL ..... 2159
UDINT_TO_BYTE ..... 2159
UDINT_TO_DATE ..... 2159
UDINT_TO_DINT ..... 2159
UDINT_TO_DT ..... 2159
UDINT_TO_DWORD ..... 2159
UDINT_TO_INT ..... 2159
UDINT_TO_LDATE ..... 2159
UDINT_TO_LDT ..... 2159
UDINT_TO_LINT ..... 2159
UDINT_TO_LREAL ..... 2159
UDINT_TO_LTIME ..... 2159
UDINT_TO_LTOD ..... 2159
UDINT_TO_LWORD ..... 2159
UDINT_TO_REAL ..... 2159
UDINT_TO_SINT ..... 2159
UDINT_TO_STRING ..... 2159
UDINT_TO_TIME ..... 2159
UDINT_TO_TOD ..... 2159
UDINT_TO_UINT ..... 2159
UDINT_TO_ULINT ..... 2159
UDINT_TO_USINT ..... 2159
UDINT_TO_WORD ..... 2159
UDINT_TO_WSTRING ..... 2159
UDP ..... 1750
UINT ..... 2235
convert ..... 2159
UINT_TO___UXINT ..... 2159
UINT_TO__ XINT ..... 2159
UINT_TO___XWORD ..... 2159
UINT_TO_BIT ..... 2159
UINT_TO_BOOL ..... 2159
UINT_TO_BYTE ..... 2159
UINT_TO_DATE ..... 2159
UINT_TO_DINT ..... 2159
UINT_TO_DT ..... 2159
UINT_TO_DWORD ..... 2159
UINT_TO_INT ..... 2159
UINT_TO_LDATE ..... 2159
UINT_TO_LDT ..... 2159
UINT_TO_LINT ..... 2159
UINT_TO_LREAL ..... 2159
UINT_TO_LTIME ..... 2159
UINT_TO_LTOD ..... 2159
UINT_TO_LWORD ..... 2159
UINT_TO_REAL ..... 2159
UINT_TO_SINT ..... 2159
UINT_TO_STRING ..... 2159
UINT_TO_TIME ..... 2159
UINT_TO_TOD ..... 2159
UINT_TO_UDINT ..... 2159
UINT_TO_ULINT ..... 2159
UINT_TO_USINT ..... 2159
UINT TO WORD ..... 2159
UINT_TO_WSTRING ..... 2159
ULINT ..... 2235
convert ..... 2159
ULINT_TO___UXINT ..... 2159
ULINT_TO__ XINT ..... 2159
ULINT_TO__XWORD ..... 2159
ULINT_TO_BIT ..... 2159
ULINT_TO_BOOL ..... 2159
ULINT_TO_BYTE ..... 2159
ULINT_TO_DATE ..... 2159
ULINT_TO_DINT ..... 2159
ULINT_TO_DT ..... 2159
ULINT_TO_DWORD ..... 2159
ULINT_TO_INT ..... 2159
ULINT_TO_LDATE ..... 2159
ULINT_TO_LDT ..... 2159
ULINT_TO_LINT ..... 2159
ULINT_TO_LREAL ..... 2159
ULINT_TO_LTIME ..... 2159
ULINT_TO_LTOD ..... 2159
ULINT_TO_LWORD ..... 2159
ULINT_TO_REAL ..... 2159
ULINT_TO_SINT ..... 2159
ULINT_TO_STRING ..... 2159
ULINT_TO_TIME ..... 2159
ULINT_TO_TOD ..... 2159
ULINT_TO_UDINT ..... 2159
ULINT_TO_UINT ..... 2159
ULINT_TO_USINT ..... 2159
ULINT_TO_WORD ..... 2159
ULINT_TO_WSTRING ..... 2159
uncomment ..... 2569
undefine, pragma ..... 2318
underflow data type ..... 2129
unforce ..... 2650
unforce values ..... 2650
unicode
text list ..... 2729
Unicode
character string in visualization ..... 3398
uninstall
device ..... 2663
library ..... 2657
union
object DUT ..... 2461
UNION ..... 2268
unit conversion
link with variable ..... 2939
object ..... 2549
unlock connection ..... 2693
UNTIL ..... 2058
update
boot project ..... 1430
firmware ..... 1430
IDs in text list ..... 2731
parameters, fbd/ld/il cfc ..... 2710
SVN project ..... 4340
update referenced pins
refactoring ..... 2578
update structured variables ..... 2727
upgrade ..... 1394, 1799
UPPER_BOUND ..... 2252
array ..... 2252
uppercase ..... 2567
keyword ..... 2796
URL
open web page, input action ..... 3373
USB ..... 148
user
login as this ..... 1828
user group ..... 1821, 3404
activate group hierarchy, visualization ..... 3406
hierarchy for permissions, visualization ..... 3404
import/export user groups, visualization ..... 3404
user input
visualization element ..... 2886
user input event
capture in application ..... 2895
user interface
language ..... 2790
user management ..... 1821
controller, device ..... 1971
create for visualization ..... 3400
device, enforce ..... 1967
disable ..... 2045
general information ..... 2039
input action, visualization ..... 3370
options, visualization ..... 3383
project ..... 1825
project settings ..... 2768
visualization ..... 3403
user-defined attributes ..... 2272
user-defined data type ..... 2461
user-defined parameters EtherNet/IP adapter ..... 2829
users and groupsdevice editor2450
project settings ..... 2768
visualization ..... 3403
USINT ..... 2235
convert ..... 2159
USINT_TO___UXINT ..... 2159
USINT TO XINT ..... 2159
USINT_TO___XWORD ..... 2159
USINT_TO_BIT ..... 2159
USINT_TO_BOOL ..... 2159
USINT_TO_BYTE ..... 2159
USINT_TO_DATE ..... 2159
USINT_TO_DINT ..... 2159
USINT_TO_DT ..... 2159
USINT_TO_DWORD ..... 2159
USINT_TO_INT ..... 2159
USINT_TO_LDATE ..... 2159
USINT_TO_LDT ..... 2159
USINT_TO_LINT ..... 2159
USINT_TO_LREAL ..... 2159
USINT_TO_LTIME ..... 2159
USINT_TO_LTOD ..... 2159
USINT_TO_LWORD ..... 2159
USINT_TO_REAL ..... 2159
USINT_TO_SINT ..... 2159
USINT_TO_STRING ..... 2159
USINT_TO_TIME ..... 2159
USINT_TO_TOD ..... 2159
USINT_TO_UDINT ..... 2159
USINT_TO_UINT ..... 2159
USINT_TO_ULINT ..... 2159
USINT_TO_WORD ..... 2159
USINT_TO_WSTRING ..... 2159
V
V2.3 project ..... 1809
V2/V3 ..... 6
VAR ..... 2113
VAR_ACCESS ..... 2335
VAR_CONFIG ..... 2121
VAR EXTERNAL ..... 2120
VAR_GLOBAL ..... 2118
VAR_IN_OUT ..... 2114
CONSTANT ..... 2117
input/output variable ..... 2114
VAR_IN_OUT_AS_POINTER
pragma, visualization ..... 3337
VAR_INFO ..... 2208
data structure, SYSTEM ..... 2208
VAR_INPUT ..... 2113
VAR_INST ..... 2120
VAR OUTPUT ..... 2114
VAR_STAT ..... 2119
VAR_TEMP ..... 2119
variable ..... 2219, 2220
access ..... 2228
add by refactoring ..... 2578
allocate memory ..... 2313
assign address ..... 1902
bit access ..... 2229
constant ..... 2121
declare ..... 1847, 1852
declare, command ..... 2572
declare, tabular ..... 1852
declare, textual ..... 1852
display format, pragma ..... 2280
external ..... 2120
global ..... 2118
hide, pragma ..... 2286
IEC 61850 server ..... 1679
initialize ..... 1851
input ..... 2113
insert, tabular ..... 2717
local ..... 2113
monitor ..... 1996
occurrence location ..... 1906
operator for information ..... 2208
output ..... 2114
Persistence Manager ..... 1926
persistent ..... 1923
PERSISTENT ..... 2122
remanent ..... 2124
remove by refactoring ..... 2580
rename ..... 1910
RETAIN ..... 1925, ..... 2124
rules for names ..... 2327
short form feature ..... 1887
static ..... 2119
switch, input action ..... 3379
temporary ..... 2119
value in online mode ..... 1996
write, input action ..... 3378
variable declaration ..... 1847
move down ..... 2718
move up ..... 2718
variable list ..... 2476
global, persistent ..... 2476
variable usage ..... 2537
variables configuration ..... 1901
VARINFO ..... 2208
operator ..... 2208
vector ..... 2253
declaration ..... 2253
initialization ..... 2253
VendorException ..... 2437
log ..... 2437
version
development System ..... 2675
Info, SVN ..... 4355
information ..... 2675
operating system ..... 2675
view
bookmarks ..... 2585
breakpoints ..... 2586
call stack ..... 2590
call tree ..... 2590
cross-reference list ..... 2587
devices ..... 2582
element properties, visualization ..... 3396
memory view ..... 2592
modules application composer ..... 2583
POUs ..... 2583
standard menu bar ..... 2582
view indentation guides ..... 2567
virtual mode
command ..... 2643
virtual system testing command ..... 2644
virus protection ..... 1341
visualization ..... 53
alarm management ..... 2909
calling with parameter transfer ..... 2951
capture user input event ..... 2895
change variable values ..... 2884
design with elements ..... 2872
display histogram ..... 2862
display variable values ..... 2883
displaying data arrays ..... 2917
executing as integrated ..... 2976
folder containing image pool ..... 2777, 3387
folder containing text list ..... 2777, 3387
font ..... 3407
gradient editor ..... 3369
grouping elements ..... 3347
language ..... 3407
memory size ..... 3401
multiply element ..... 2919, 3350
object ..... 3393
object properties ..... 3388
operate with gestures ..... 2887
placing an element in the background ..... 3349
project settings ..... 2776, 3387
recipes ..... 2939
reference ..... 2941
refrigerator controller ..... 2855
run ..... 2973
switch, input action ..... 3373
text list ..... 2906
text, tooltip ..... 2906
trace wizard ..... 3358
ungrouping elements ..... 3348
variable values in tables, example ..... 2864, 2918
web browser, example ..... 2865
Visualization ..... 2852
Assignment of the visualizations to the display variants ..... 3402
visualization editor configuration ..... 3384
visualization element ..... 2987
add via command ..... 3364
animation ..... 2912
configuration with interface property . . 2777, 3387
configure size and position ..... 2874
configuring an offset ..... 2912
configuring while rotating ..... 2913
design in color ..... 2876
element list ..... 3342
enter static text ..... 1894
in visualization toolbox view ..... 3394
many of the same type ..... 2919
multiple template ..... 2919
repository ..... 3361
select in visualization toolbox ..... 2873
user input ..... 2886
view assignment in visualization toolbox ..... 3368
visualization manager ..... 3398
activate multitouch ..... 3401
settings ..... 3398
user management ..... 3400
visualization styles editor ..... 3399
visualization profile ..... 3385
project setting ..... 2777, 3388
repository ..... 3361
version ..... 2779, 3385
visualization style ..... 2979
copy ..... 3749
create ..... 2984, 3749
determine the appearance ..... 2979
edit ..... 2980
install ..... 2986
manage in repository ..... 2985
manage repository ..... 2986
option ..... 3382
preview in the visualization manager ..... 3399
preview of installed styles ..... 3364
repository and contents ..... 3364
selection in the visualization manager ..... 3399
switch ..... 2980
uninstall ..... 2986
update version ..... 2980
version ..... 2780, 3386
visualization style editor ..... 3750
open ..... 2983
open from development system ..... 2983
visualization toolbox ..... 3394
VisualizationManager ..... 53
voltage sag ..... 1480
VUM_ChangePassword user management dialog, visualization . . . . . 3400
VUM_Loginuser management dialog, visualization . . . . . 3400
VUM_UserManagementuser management dialog, visualization . . . . . 3400
W
W ..... 2231
size prefix ..... 2231
wall mounting accessory ..... 1324, 1331
warm start ..... 2634
warning disable, pragma ..... 2315
warning notices ..... 10, 12
warning restore, pragma ..... 2315
watch
add all forces to watchlist ..... 2585
open view ..... 2584
watch all forces ..... 1989
watch list ..... 2002
watch all forces ..... 2584
Watchdog ..... 4425
watchlist ..... 2584
add variable ..... 2743
web browser ..... 3262, 3687
visualization element ..... 3262, 3687
visualization, example ..... 2865
webvisu ..... 2974
calling a page in the browser ..... 2976
execute ..... 2974
WebVisu
object ..... 3409
security ..... 2042
WHILE ..... 2058
whitespace ..... 2566
show in text editor ..... 2566
window
auto hide ..... 2671
dock ..... 2671
float ..... 2671
reset layout ..... 2670
window <n> ..... 2673
windows ..... 2672
hide ..... 1807
layout ..... 1806
move ..... 1806
resize ..... 1806
show ..... 1807
toggle ..... 1807
Windows Certificate Store ..... 1820
Windows server 1594, 1647 ..... 1799
wink ..... 2640
word
addressing mode ..... 2231
WORD ..... 2235
convert ..... 2159
WORD_TO__XWORD ..... 2159
WORD_TO__UXINT ..... 2159
WORD_TO_XINT ..... 2159
WORD_TO_BIT ..... 2159
WORD_TO_BOOL ..... 2159
WORD_TO_BYTE ..... 2159
WORD_TO_DATE ..... 2159
WORD_TO_DINT ..... 2159
WORD_TO_DT ..... 2159
WORD_TO_DWORD ..... 2159
WORD_TO_INT ..... 2159
WORD_TO_LDATE ..... 2159
WORD_TO_LDT ..... 2159
WORD_TO_LINT ..... 2159
WORD_TO_LREAL ..... 2159
WORD_TO_LTIME ..... 2159
WORD_TO_LTOD ..... 2159
WORD_TO_LWORD ..... 2159
WORD_TO_REAL ..... 2159
WORD_TO_SINT ..... 2159
WORD_TO_STRING ..... 2159
WORD_TO_TIME ..... 2159
WORD_TO_TOD ..... 2159
WORD_TO_UDINT ..... 2159
WORD_TO_UINT ..... 2159
WORD_TO_ULINT ..... 2159
WORD_TO_USINT ..... 2159
WORD_TO_WSTRING ..... 2159
worksheet ..... 2685
write ..... 1987
write protection ..... 1824
project ..... 1823
writing values
command ..... 2649
prepare value ..... 2749
WSTRING ..... 2242
convert ..... 2175
index access ..... 2244
WSTRING_TO___UXINT ..... 2175
WSTRING_TO UXWORD ..... 2175
WSTRING_TO
XINT ..... 2175
WSTRING TO BIT ..... 2175
WSTRING_TO_BOOL ..... 2175
WSTRING_TO_BYTE ..... 2175
WSTRING_TO_DATE ..... 2175
WSTRING_TO_DINT ..... 2175
WSTRING_TO_DT ..... 2175
WSTRING_TO_DWORD ..... 2175
WSTRING_TO_INT ..... 2175
WSTRING_TO_LDATE ..... 2175
WSTRING_TO_LDT ..... 2175
WSTRING_TO_LINT ..... 2175
WSTRING_TO_LREAL ..... 2175
WSTRING_TO_LTIME ..... 2175
WSTRING_TO_LTOD ..... 2175
WSTRING_TO_LWORD ..... 2175
WSTRING_TO_REAL ..... 2175
WSTRING_TO_STRING ..... 2175
WSTRING_TO_TIME ..... 2175
WSTRING_TO_TOD ..... 2175
WSTRING_TO_UDINT ..... 2175
WSTRING_TO_UINT ..... 2175
WSTRING_TO_ULINT ..... 2175
WSTRING_TO_USINT ..... 2175
WSTRING_TO_WORD ..... 2175
X
X. ..... 2231
size prefix ..... 2231
XADD ..... 2213
XOR ..... 2140
XORN ..... 2087
XSIZEOF ..... 2138
Z
zoom
factor ..... 2048
zoom in/out graphical editor ..... 2048
new.abb.com/p/c
© Copyright 2021-2024 ABB.
We reserve all rights in this document and in the information contained therein. Reproduction, use or disclosure to third parties without express authority is strictly forbidden.


[^0]:    As a specialist for AC500 V2 CPU range, new to AC500 V3 CPU range

[^1]:    Complete reset of the CPU

    To do a complete reset of the CPU thereby erasing the application from the RAM and flash EEPROM do the following.

[^2]:    Complete reset of the CPU

    To do a complete reset of the CPU thereby erasing the application from the RAM and flash EEPROM do the following.

[^3]:    Engineering Provides access from the programming system to an external project database in which the interface program source code of one or several automation projects is managed. Optionally, a version control system can be used in order to ensure data consistency of the program code for several different users and projects.

[^4]:    1 MC5141 memory card
    2 MC5102 micro memory card
    3 Micro memory card adapter
    4 Memory card slot

[^5]:    *) The value is the sampling time on the option board. The complete conversion cycle time is also related to the CPU cycle time.

[^6]:    Remarks:
    ${ }^{11}$ : The values are for information only and cannot be fulfilled altogether. The available resources are limited at the end by the maximal downloadable application size for each CPU.

[^7]:    *) Modules in lifecycle Classic are available from stock but not recommended for planning and commissioning of new installations.

[^8]:    5 LEDs for state display
    22 rotary switches for address setting
    3 Label
    42 serial interfaces: COM1 and COM2
    ${ }_{x}^{*}+{ }_{*}^{*}$ Sign for XC version

[^9]:    ${ }^{1}$ ) When inputs are used with 24 VAC , external surge limiting filters are required.

[^10]:    ${ }^{1}$ ) with CS31 and addresses less than 70, the value is increased by 1

[^11]:    ${ }^{1}$ ) Per group in case of group fuse protection. For each channel in case of channel-by-channel fuse protection. The maximum current per group must not be exceeded.

[^12]:    ${ }^{1}$ ) Per group in case of group fuse protection. For each channel in case of channel-by-channel fuse protection. The maximum current per group must not be exceeded.

[^13]:    ${ }^{1}$ ) With CS31 and addresses smaller than 70 and FBP, the value is increased by 1

[^14]:    ${ }^{1}$ ) With CS31 and addresses less than 70 and FBP, the value is increased by 1
    ${ }^{2}$ ) Not with FBP

[^15]:    Connection of sensors with frequency outputs

[^16]:    1 Din rail 15 mm
    2 Din rail 7.5 mm

[^17]:    ${ }^{1}$ ) With a faulty ID, the module reports a "parameter error" and does not perform cyclic process data transmission
    ${ }^{2}$ ) For a description of the counter operating modes, please refer to the fast counter section
    ${ }^{4}$ Chapter 5.2.4.3.2.9 "Fast counter" on page 606.

[^18]:    *) The parameters behaviourAOatCommunicationFault and behaviourDOatCommunicationFault are only analyzed if the Failsafe-mode is ON.

[^19]:    *) The parameter behaviourDOatCommunicationFault is only analyzed if the Failsafe-mode is ON.

[^20]:    *) Due to the direct connection to the output, the demagnetizing varistor is also effective at the input (see figure) above. This is why the difference between UPx and the input signal may not exceed the clamp voltage of the varistor. The varistor limits the voltage to approx. 36 V . Following this, the input voltage must range from $-12 \mathrm{~V} \ldots+30 \mathrm{~V}$ when $\mathrm{UPx}=24 \mathrm{~V}$ and from $-6 \mathrm{~V} . .+30 \mathrm{~V}$ when $\mathrm{UPx}=30 \mathrm{~V}$.

[^21]:    Remarks:

[^22]:    \& Chapter 6.8.2.12 "Fast counters in AC500 devices" on page 4536

[^23]:    *) Priorization with the aid of VLAN-ID including priority level

[^24]:    *) Priorization with the aid of VLAN-ID including priority level

[^25]:    « Chapter 6.8.2.12 "Fast counters in AC500 devices" on page 4536

[^26]:    *) Priorization with the aid of VLAN-ID including priority level

[^27]:    Legal Notice | © WIBU-SYSTEMS AG | CodeMeter License Central WebDepot v19.05.205.500.ws | 2021-05-28 13:14:29 (UTC)

[^28]:    Remove user accounts further inquiry! An error message appears if you try to delete all users from a group. At least one entry must remain.

[^29]:    I/Os

    Mapping of the Double-click on the desired "PROFINET-IO-Device" module object in the device tree to show current I/O mappings connected to this module.

[^30]:    ${ }^{1}$ ) Note that columns "Port State", "Frame Error Counters", "Physical Layer Error Counters" and "Link Lost Counters" show "LLD: Timeout", if this state is NOT CONNECTED, because this information is not accessible.
    ${ }^{2}$ ) Note that this column contains "LLD: Timeout", if slave state is NOT CONNECTED.

[^31]:    Firmware $\quad$ As of Automation Builder 2.2.1, the IP Configuration Tool can be used to perform firmware update updates for CI52x-MODTCP devices.

[^32]:    Alternative branch

    The horizontal line before and after the branch is a single line.
    In an alternative branch, the branch lines must begin and end with transitions. The branch lines can contain additional branches.

[^33]:    Monitoring a scalar variable

[^34]:    Subtracting pointers

    The result of the difference between two pointers is a value of type DWORD, even on 64-bit platforms when the pointers are 64-bit pointers.

[^35]:    Status bar The status bar which is always visible notifies you about the data source type and the most important communication settings. When the communication is established by means of the data source type CODESYS Symbolic, the name of the data source type, the connection type, and the network name of the remote device are displayed. When the communication is established by means of data source type CODESYS ApplicationV3, then the name of the data source type, the location of the remote project, and the instance name of the remote application.
    Example:
    CODESYS Symbolic (CODESYS V3): PLC_Name

[^36]:    Example of the "<device name> I/O Mapping" tab for a CAN bus slave:

[^37]:    Calling a function In ST, you can use the call of a function as an operand in expressions. In SFC, you can use a function call only within step actions or transitions.

[^38]:    Special methods of a function block

[^39]:    NOTICE!
    An implicit monitoring function call does not occur for THIS pointer and SUPER pointer.

[^40]:    Input field
    'Single scaling
    (factor)'
    The input variable is multiplied by the factor.

[^41]:    Diagram pre- Requirement: A diagram is selected in the tree view "Presentation (Diagrams)" view

[^42]:    Predefined con- Use this option to employ existing connections from an EDS file. The data that can be changed nection (EDS are defined in the EDS file.
    file)

[^43]:    Element property 'Absolute movement'

    The properties contain IEC variables for controlling the position of the element dynamically. The reference point is the upper left corner of the element. In runtime mode, the entire element is moved.

[^44]:    Element property 'Access
    rights'

[^45]:    Element property 'Position'

    The position defines the location and size of the element in the visualization window. These are based on the Cartesian coordinate system. The origin is located at the upper left corner of the window. The positive horizontal $x$-axis runs to the right. The positive vertical $y$-axis runs downwards.

[^46]:    Element property 'References'

[^47]:    Element property 'Access
    rights'

[^48]:    Element prop-
    erty 'Display
    type'

[^49]:    Element prop- The variables control the element behavior dynamically. erty 'State variables'

[^50]:    ${ }^{1}$ ) Slot input in the block can be ignored. Similar to ETH input of the ModMast blocks.
    ${ }^{2}$ ) Error generated only in the primary PLC, to reset ACK input to be used.
    ${ }^{3}$ ) No runtime error in function block. Module generates S-Err and ZP Blinks.
    ${ }^{4}$ ) No runtime error in function block. Module generates S-Err.
    ${ }^{5}$ ) Runtimer Error bit2 gets reset when the PLC is switched over and error won't be available in any of the PLC regardless of its Primary status.

[^51]:    "Dynamic limits"

    Users can update the maximum limits here. Some parameters depend on the drive settings and needs to be set correctly to get the desired result.

    The user can set the "Maximum application velocity" to a desired value to limit the maximum application speed.

[^52]:    You can undo the changes that the "Extract function" command made in your project by positioning the cursor in the device tree and clicking "Edit $\rightarrow$ Undo".

[^53]:    "Filter" $\quad$ Input field for strings to be searched for

